第一章

非線性方程式的解法

1.1 前言

在工程分析方面常會遇到解非線性方程式之問題。如在柱之挫曲問題上須求下列方程式之根。

$$\tan xL - xL = 0 \tag{1.1}$$

大部分非線性方程式均無法以直接解析方法求解，一般皆以反覆試算過程求解。本章將介紹幾種常用方法並附以符傳程式以供參考。

1.2 直接代入法

直接代入法為反覆試算法中之最基本型式，大部分試算法皆可視為直接代入法，故予最先討論。該法係將原來一般非線性方程式(1.2)，改寫成式(1.3)之型式。

$$f(x) = 0 \tag{1.2}$$

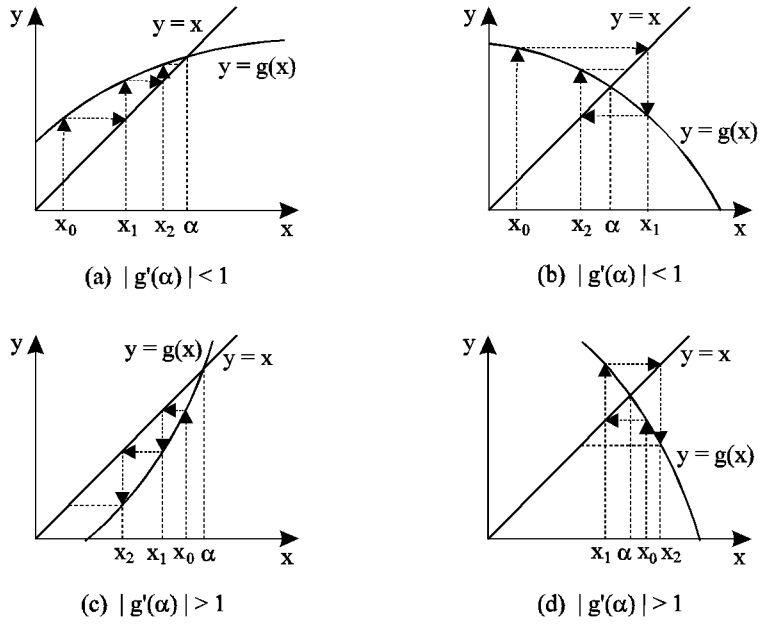
$$x = g(x) \tag{1.3}$$

若$x = \alpha$為方程式(1.2)之一根，則$x = \alpha$亦為式(1.3)之一根，即$f(\alpha) = 0,\ \alpha = g(\alpha)$。此$\alpha$值即為所欲求解之值，當然解題之前無法預知，不過我們可以循一定途徑，由一假設之值開始可逐步逼近所要之α值。首先假設$x = x_o$為起始值，代入式(1.3)等號右邊算得$g(x_o)$之值，若$g(x_o) \neq x_o$則x_o

非為所求之根，令其為x_1，即$x_1 = g(x_o)$。再以$x = x_1$代入式(1.3)等號右邊算得$x_2 = g(x_1)$。如此重覆以下式計算，可連續算得$x_1, x_2, ..., x_n$之值。

$$x_{i+1} = g(x_i), \qquad i = 0, 1, ..., n-1 \tag{1.4}$$

若上述重覆計算最後可得x_n等於或近似等於x_{n-1}，或$|x_n - x_{n-1}| < \epsilon$（其中$\epsilon$為一很小值，可用以控制精度），則$x_n$即等於或近似等於所欲求得之$\alpha$，亦即$x_n$可收斂至所求之根$\alpha$。上述重覆運算並不一定均能收斂至所求之根。參考[圖一]各情形，可知只有在$|\frac{dg(x)}{dx}|_{x=\alpha} < 1$時，數列$x_1, x_2, ..., x_n$才會收斂。



圖一 直接代入法

現舉一簡單實例說明直接代入法的應用，欲求$\sqrt{3}$之值，相當於求下列方程式之根。

$$f(x) = x^2 - 3 = 0 \tag{1.5}$$

上列方程式改寫成式(1.3)可有許多種不同型式，下列僅舉三種型式加以討論。

$$x = \frac{1}{2}(x + \frac{3}{x}) \tag{1.6}$$

$$x = x^2 + x - 3 \tag{1.7}$$

$$x = \frac{3}{x} \tag{1.8}$$

試從$x_o = 2$為起始值：由式(1.6)可得$x_1 = 1.75$，$x_2 = 1.73214$，$x_3 = 1.73205$，$x_4 = 1.73205$；由式(1.7)可得$x_1 = 3$，$x_2 = 9$，$x_3 = 87$，$x_4 = 7653$；由式(1.8)可得$x_1 = 1.5$，$x_2 = 2$，$x_3 = 1.5$，$x_4 = 2$。顯然可見，由式(1.6)可以很快求得$\sqrt{3}$之值，而由式(1.7)及式(1.8)則均無法收斂至$\sqrt{3}$。現計算三式之$g'(x)$分別為：$\frac{1}{2}(1 - \frac{3}{x^2})$，$2x + 1$與$-\frac{3}{x^2}$。三式之$g'(\sqrt{3})$分別為：0，4.4641與$-1$。因此可驗証該值之絕對值小於1者會收斂；否則即不收斂。

以下推導上述收斂條件之理論根據：將式(1.3)中之$g(x_i)$對$x = \alpha$點做泰勒(Taylor)級數展開如式(1.9)，其中$g(\alpha) = \alpha$，故得式(1.10)，或式(1.11)，其中$\delta_i = x_i - \alpha$。

$$x_{i+1} = g(\alpha) + g'(\alpha)(x_i - \alpha) + \cdots + \frac{1}{p!}g^{(p)}(\alpha)(x_i - \alpha)^p + \cdots \tag{1.9}$$

$$= \alpha + g'(\alpha)(x_i - \alpha) + \cdots + \frac{1}{p!}g^{(p)}(\alpha)(x_i - \alpha)^p + \cdots \tag{1.10}$$

$$\delta_{i+1} = g'(\alpha)\delta_i + \frac{1}{2}g''\delta_i^2 + \cdots + \frac{1}{p!}g^{(p)}(\alpha)\delta_i^p + \cdots \tag{1.11}$$

若

$$g'(\alpha) = g''(\alpha) = \cdots = g^{(p-1)}(\alpha) = 0, \qquad g^{(p)}(\alpha) \neq 0 \tag{1.12}$$

則

$$\delta_{i+1} \doteq \frac{1}{p!}g^{(p)}(\alpha)\delta_i^p \tag{1.13}$$

因$\frac{1}{p!}g^{(p)}(\alpha)$為一固定之值，$\delta_{i+1}$之大小即正比於$\delta_i^p$而遞減，一般稱此直接代入法之收斂階數(convergence order)為p。常見之直接代入法之收斂階數為1或2，不太容易有$p > 2$之方法，下節將介紹之牛頓–瑞福生法(Newton–Raphson method)之收斂階數為2。當收斂階數為1時，直接代入法是否會收斂，自然就取決於$|g'(\alpha)|$是否小於1了。

直接代入法之符傳程式十分簡單，下列為一通用之直接代入法符傳主程式，該程式首先讀入一起始值$X = x_o$，$EPS = \epsilon$及$N = n$。程式內計算$g(x_i)$係呼叫使用者自寫函數$G(X)$為之。依序利用式(1.4)計算$x_1, x_2, ...$至x_n為止而停止運算，其間若發覺$|x_{i+1} - x_i| < \epsilon$，則不再運算至$x_n$，並印出$x_i$及$x_{i+1}$做為所求之解。主程式後所附自寫函數係以式(1.6)為例寫成，可做為書寫其他函數之參考。

```
************************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      READ(*,'(2F10.2,I5)') X,EPS,N
      DO 20 I=1,N
      XOLD=X
      X=G(X)
      IF(ABS(X-XOLD).LE.EPS) GO TO 40
   20 CONTINUE
      WRITE(*,'(/'' NOT CONVERGE IN'',I4,'' ITERATIONS WITHIN'',1PE10.3
     *          /'' THE LAST TWO SUCCESSIVE X VALUES ARE'',2E13.6
     *          /'' TRY ANOTHER TYPE OF G(X) FUNCTION.'')') N,EPS,XOLD,X
      STOP
C**   ------------------------------------------------------------    **
   40 WRITE(*,'(/'' THE SEQUENCE CONVERGES IN'',I4
     *          ,'' ITERATIONS WITHIN'',1PE10.3
     *          /'' THE LAST TWO SUCCESSIVE X VALUES ARE'',2E13.6)')
     *          I,EPS,XOLD,X
      STOP
      END
************************************************************************
      FUNCTION G(X)
      IMPLICIT REAL*8 (A-H,O-Z)
      G=0.5*(X+3.0/X)
      RETURN
      END
************************************************************************
```

1.3 牛頓–瑞福生法

牛頓–瑞福生法（Newton-Raphson method）解方程式(1.2)之根，相當於在直接代入法中選擇

$$g(x) = x - \frac{f(x)}{f'(x)} \tag{1.14}$$

亦即以下式做重覆運算以求解。

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{1.15}$$

式(1.15)之推導：將$f(x_{i+1})$對$x = x_i$點做泰勒級數展開如式(1.16)，略去高次項，並令$f(x_{i+1}) = 0$，解x_{i+1}即得式(1.15)。因為用以求x_{i+1}使$f(x_{i+1}) = 0$之方程式為略去高次項之近似式，故所得之x_{i+1}，並不能真的使$f(x_{i+1})$為零，但應該是比x_i更好的近似值，因此須做多次重覆運算。

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \cdots \tag{1.16}$$

牛頓-瑞福生法一般收斂很快，因為函數$g(x)$之導函數為：

$$g'(x) = 1 - \frac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f'(x)]^2} \tag{1.17}$$

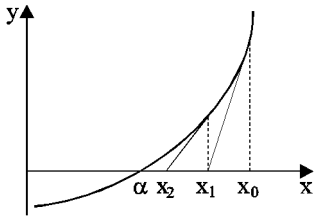
當$x = \alpha$時，因$f(\alpha) = 0$，故若$f'(\alpha)$不等於0，則$g'(\alpha) = 0$，故式(1.15)在根$x = \alpha$之附近必然會收斂，其收斂階數為2。[圖二]說明牛頓–瑞福生法之反覆運算過程及收斂情形。

前節所列符傳主程式亦可用來以牛頓–瑞福生法求式(1.1)之根。只要自寫一函數$G(x)$用以計算$x - f(x)/f'(x)$之函數值即可。以式(1.1)之非線性方程式為例。

$$g(x) = x - \frac{\tan xL - xL}{(\sec^2 xL - 1)L} = \frac{x}{\sin^2 xL} - \frac{\cos xL}{L \sin xL} \tag{1.18}$$

其符傳函數可寫成如下，其後為輸入資料及計算結果。

```
************************************
      FUNCTION G(X)
      IMPLICIT REAL*8 (A-L,O-Z)
      DATA L/10.0/
      XL=X*L
      CS=COS(XL)
      SS=SIN(XL)
      G=X/(SS*SS)-CS/(L*SS)
      RETURN
      END
************************************
```



圖二 牛頓–瑞福生法

```
      0.43   0.00001   10

 THE SEQUENCE CONVERGES IN   8 ITERATIONS WITHIN 1.000E-05
 THE LAST TWO SUCCESSIVE X VALUES ARE 4.493471E-01 4.493409E-01
```

1.4 半區間法

若方程式(1.2)僅有一單根介於x_a與x_b之間，且函數$f(x)$在x_a與x_b之間連續，則$f(x_a)$與$f(x_b)$必為異號值。令$x_c = (x_a + x_b)/2$，並計算$f(x_c)$，再根據$f(x_c)$之正負，做下列之判斷：

(1)若$f(x_c)$與$f(x_b)$同號，根必介於x_a與x_c之間。可改令x_b等於x_c，並重覆上述運算。

(2)若$f(x_c)$與$f(x_a)$同號，根必介於x_b與x_c之間。可改令x_a等於x_c，並重覆上述運算。

不管$f(x_c)$與何者同號，根之可能範圍即縮小一半。若前述步驟共進行n次，則根之誤差將小於$|x_a - x_b| * 2^{-n}$，因此可以根據x_a與x_b之差值，適當選擇n值而達到所要求之精度。

若根之範圍不能確定在某一區間時，可從一指定之最小值x_{min}開始，以δx為一區間，搜尋每一區間是否有根存在。意即首先判斷x_{min}與$x_{min}+\delta x$之間，$f(x)$是否變號，如未變號，則判斷$x_{min}+\delta x$與$x_{min}+2\delta x$之間，$f(x)$是否變號，依此類推，直至某一區間$f(x)$變號為止。再以前述半區間法求得較精確之根。如欲求較多之根，可用已求得之根之右側值為x_{min}，繼續搜求較多之根。

δx之決定應注意兩項原則：一為δx愈大則搜求速度快，但須做較多次之半區間判斷；二為δx須小於最靠近之兩根之差值，否則在搜尋過程中，可能會漏掉所要之根。

[表一]即為以半區間法求根之一通用副程式，該副程式能從$XMIN$開始以DX為一區間搜尋至有根之區間後改以半區間法求更精確之根。所得根之誤差小於$DX*2^{-M}$，如果搜尋至$XMAX$仍無根時，則回到呼叫程式。

［表一］ 半區間法副程式

```
************************************************************************
      SUBROUTINE SEARCH(XX,FX,F,XMIN,XMAX,DX,ICUT,FLMT)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ================================================================    **
      FA= 1.0
      FB=-1.0
      XX=XMIN
      IR=0
C**   +--------------------------------------------------------------+    **
C**   | Compute function value                                       |    **
C**   +--------------------------------------------------------------+    **
   30 FX=F(XX)
      IR=IR+1
      IF(FX.EQ.0.0) RETURN
C*    WRITE(*,'('' IR,FX,XX'',I5,1P2E15.6)') IR,FX,XX
C**   +--------------------------------------------------------------+    **
C**   | Return on discontinuous point                                |    **
C**   +--------------------------------------------------------------+    **
      IF(DABS(FX).GE.FLMT) RETURN
C**   +--------------------------------------------------------------+    **
C**   | Set limits for next new root                                 |    **
C**   +--------------------------------------------------------------+    **
      IF(FX.LT.0.0) THEN
        XA=XX
        FA=FX
      ELSE
        XB=XX
        FB=FX
      ENDIF
C**   +--------------------------------------------------------------+    **
C**   | Search interval for changing sign                            |    **
C**   +--------------------------------------------------------------+    **
      IF(FA*FB.GT.0.0) THEN
```

```
        IF(XX.GT.XMAX) THEN
          WRITE(*,'('' NO ROOT BETWEEN'',1P2E15.6)') XMIN,XMAX
          RETURN
        ENDIF
        XX=XX+DX
        IE=IR+ICUT
        GO TO 30
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Get new root by half interval method                   |    **
C**   +--------------------------------------------------------+    **
      XX=0.5*(XA+XB)
C**   +--------------------------------------------------------+    **
C**   | Get new root by false position method                  |    **
C**   +--------------------------------------------------------+    **
C  75 XX=XA-FA*(XB-XA)/(FB-FA)
C**   +--------------------------------------------------------+    **
C**   | Check the iteration numbers                            |    **
C**   +--------------------------------------------------------+    **
      IF(IR.LE.IE) GO TO 30
      WRITE(*,*) ' THE NUMBER OF ITERATIONS EXCEEDS',ICUT
      RETURN
      END
*************************************************************************
```

如欲求方程式(1.1)介於1.0至100.0間之10個根，可用[表二]之主程式及對應之函數程式，配合[表一]之副程式求得。

[表二] 半區間法主程式

```
*************************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL F
      XMAX=100.0
      XMIN=1.0
      DX=0.1
      ICUT=20
      FLMT=1000.
      DO 30 I=1,10
      CALL SEARCH(X,FX,F,XMIN,XMAX,DX,ICUT,FLMT)
      IF(ABS(FX).GE.FLMT)
     *WRITE(*,'('' FOLLOWING MAY BE A DISCONTINUOUS POINT'')')
      WRITE(*,'('' I=''I3,'', X(I)=''1PE13.6,'', F(X)=''E13.6)') I,X,FX
      XMIN=X+DX/10.0
      IF(X.GT.XMAX) STOP
   30 CONTINUE
      STOP
      END
*************************************************************************
      FUNCTION F(X)
      IMPLICIT REAL*8 (A-L,O-Z)
      DATA L/1.0/
      XL=X*L
      F=SIN(XL)/COS(XL)-XL
      IF(ABS(F).LE.0.00001) F=0.0
      RETURN
      END
*************************************************************************
```

[表三]為上列程式之輸出結果，注意其中第1,3,5,7,9等五個x所對應之$f(x)$分別為$2066.7,\ -2528.5,\ -4769.0,\ 4952.2,\ 3032.9$皆相當大，可知並非$f(x)=0$的根。實際上$f(x)$在這五個$x$值處為不連續，$x$值兩側之$f(x)$，一邊趨於正無窮大，一邊趨於負無窮大，因此在利用半區間法時，應校核最後算得之$f(x)$以判斷所得之x是否為所要之根。另請注意在F函數程式中之$IF(ABS(F).LE.0.00001)\ \ F=0.0$指令，可用以按$|f(x)|<0.00001$之條件結束反覆之試算。

[表三] 半區間法輸出結果

```
FOLLOWING MAY BE A DISCONTINUOUS POINT
I=  1, X(I)= 1.570313E+00, F(X)= 2.065321E+03
I=  2, X(I)= 4.493409E+00, F(X)= 0.000000E+00
FOLLOWING MAY BE A DISCONTINUOUS POINT
I=  3, X(I)= 4.712784E+00, F(X)=-2.535118E+03
I=  4, X(I)= 7.725252E+00, F(X)= 0.000000E+00
FOLLOWING MAY BE A DISCONTINUOUS POINT
I=  5, X(I)= 7.854002E+00, F(X)=-4.923385E+04
I=  6, X(I)= 1.090412E+01, F(X)= 0.000000E+00
FOLLOWING MAY BE A DISCONTINUOUS POINT
I=  7, X(I)= 1.099537E+01, F(X)= 4.926082E+03
I=  8, X(I)= 1.406619E+01, F(X)= 0.000000E+00
FOLLOWING MAY BE A DISCONTINUOUS POINT
I=  9, X(I)= 1.413713E+01, F(X)= 2.811500E+04
I= 10, X(I)= 1.722076E+01, F(X)= 0.000000E+00
```

1.5 割線法與假位法

於前節所述之半區間法計算新x_{i+1}係取區間之中點，但如新x_{i+1}取$(x_a,f(x_a))$與$(x_b,f(x_b))$兩點之連線與x軸之交點之x_{i+1}值，則稱為假位法（false position method）。（見[表一]之附註指令75）。又如新x_{i+1}取最近所得之兩點$(x_{i-1},f(x_{i-1}))$，$(x_i,f(x_i))$連線與x軸之交點之x_{i+1}值，則稱為割線法（secant method）。割線法之好處為求根速度快，但缺點為不可靠，因其不像假位法或半區間法永遠用使$f(x)$為異號的兩個點，因此，當附近之$f'(x)$接近零時，所得新x_{i+1}可能會遠離$x_{i-1},\ x_i$等值，而不能趨近附近之根。

[表四]程式係兼用割線法與假位法求根。正常運算以割線法求新x_{i+1}，但亦如同半區間法或假位法保留一個最小區間，該區間兩端之函數為異號值。當割線法所得之新x_{i+1}落於區間內時則予採用，反之，當割線法所得之新x_{i+1}落於區間外時，則自動改用區間兩端點以假位法計算新x_{i+1}，則該x_{i+1}必落於區間內而保證x_{i+1}趨近於區間內之根。

此法具有割線法速度快與假位法穩定可靠之好處，因此特別推薦採用。其唯一缺點為只適用於求算單一函數之實根。

［表四］ 割線法與假位法合成副程式

```
************************************************************************
      SUBROUTINE SEARCH(XX,FX,F,XMIN,XMAX,DX,ICUT,FLMT)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C*I   XX   = Argument of function                                     **
C*I   FX   = Function value of XX, F(XX)                              **
C*F   F(X) = Function to calculate F(X)                               **
C*I   XMIN,XMAX,DX = Search limits and increment for root             **
C*I   ICUT = Max number of iterations                                 **
C*I   FLMT = Limited function values of F(XX)                         **
C**   ============================================================    **
      FN= 1.0E30
      FP=-1.0E30
      XX=XMIN
      IR=0
C**   +----------------------------------------------------------+    **
C**   | Compute function value                                   |    **
C**   +----------------------------------------------------------+    **
   30 FX=F(XX)
      IR=IR+1
      IF(FX.EQ.0.0) RETURN
C*    WRITE(*,'('' IR,FX,XX'',I5,1P2E15.6)') IR,FX,XX
C**   +----------------------------------------------------------+    **
C**   | Return on discontinuous point                            |    **
C**   +----------------------------------------------------------+    **
      IF(DABS(FX).GE.FLMT) RETURN
C**   +----------------------------------------------------------+    **
C**   | Push down old values, and put the new one on top         |    **
C**   +----------------------------------------------------------+    **
      X1=X2
      X2=XX
      F1=F2
      F2=FX
C**   +----------------------------------------------------------+    **
C**   | Set limits for next new root                             |    **
C**   +----------------------------------------------------------+    **
      IF(FX.LT.0.0) THEN
        XN=XX
        IF(FX.LT.FN) IE=IE-1
        FN=FX
      ELSE
        XP=XX
        IF(FX.GT.FP) IE=IE-1
        FP=FX
      ENDIF
C**   +----------------------------------------------------------+    **
C**   | Search interval for changing sign                        |    **
C**   +----------------------------------------------------------+    **
      IF(FN*FP.GT.0.0) THEN
        IF(XX.GT.XMAX) THEN
          WRITE(*,'('' NO ROOT BETWEEN'',1P2E15.6)') XMIN,XMAX
          RETURN
```

```
        ENDIF
        XX=XX+DX
        IE=IR+ICUT
        GO TO 30
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Get new root by secant method                          |    **
C**   +--------------------------------------------------------+    **
      IF(F1.NE.F2) THEN
        XX=X1-F1*(X1-X2)/(F1-F2)
C**     +------------------------------------------------------+    **
C**     | Check if the new root goes outside the limits        |    **
C**     +------------------------------------------------------+    **
        IF((XX-XP)*(XX-XN).GT.0.0) THEN
C**       +----------------------------------------------------+    **
C**       | YES: Get new root by false position method         |    **
C**       +----------------------------------------------------+    **
          XX=XP-FP*(XP-XN)/(FP-FN)
        ENDIF
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Check the iteration numbers                            |    **
C**   +--------------------------------------------------------+    **
      IF(IR.LE.IE) GO TO 30
      WRITE(*,*) ' THE NUMBER OF ITERATIONS EXCEEDS',ICUT
      RETURN
      END
**************************************************************************
```

1.6 重根之處理

當$f(x)=0$有重根時，可令$u(x)=f(x)/f'(x)$，再改求$u(x)=0$之根。由下列關係可證明$u(x)=0$之根均為$f(x)=0$之根，且均為單根。
設$f(x)=0$之根α為q重根($q\geq 1$，即以下推導亦適用於單根)，則

$$f(x) = (x-\alpha)^q h(x), \qquad h(\alpha)\neq 0 \tag{1.19}$$

$$f'(x) = (x-\alpha)^{q-1}[q\,h(x)+(x-\alpha)h'(x)] \tag{1.20}$$

故

$$u(x) = (x-\alpha)k(x), \qquad k(\alpha)\neq 0 \tag{1.21}$$

上式中

$$k(x) = h(x)/[q\,h(x)+(x-\alpha)h'(x)] \tag{1.22}$$

$$k(\alpha) = h(\alpha)/[q\,h(\alpha)] = 1/q \neq 0 \tag{1.23}$$

故$u(x)=0$有α之根且為單根。

1.7 黑箱作業程式

前面所提供之求解程式均採用一般慣用之方式：即函數之計算由求解程式內部直接呼叫使用者提供之函數程式。此種方式在函數程式需要用到很多參數時，會有許多不便之處。[表五]所提供之求解程式$XZERO$，將不再直接呼叫任何副程式或函數程式，所有數值均由使用者於呼叫$XZERO$時透過呼叫參數提供。對使用者而言，此求解程式就像一個黑箱子，此程式可根據歷次所得資料（如x與$f(x)$值），運用其所採用方法，回覆一個更接近$f(x)=0$之根之x值。為了使同一個程式同時可以求解很多不同的方程式，其所需保存的資料均放置於使用者所提供之暫存數列$BUF(11)$中，並由使用者自行保存。

利用[表五]副程式，[表一]至[表四]等副程式，即可改為如[表六]之副程式。另外在以後數節之求解多元方程式之程式中，所有涉及一元方程式之求解均呼叫此程式。如果這些程式不呼叫此程式，而呼叫以一般慣用方式寫成之程式，則必顯得相當複雜。

[表五] 一元方程式求根副程式

```
************************************************************************
      FUNCTION XZERO(X,F,DX,ISTEP,B)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(11),B(11)
      EQUIVALENCE (A(1),X0),(A(2),X1),(A(3),X2)
     *           ,(A(4),F0),(A(5),F1),(A(6),F2)
     * ,(A(7),XP),(A(8),XN),(A(9),FP),(A(10),FN),(A(11),ST)
C**   ============================================================    **
C*I   X     = Argument of function                                    **
C*I   F     = Function value of X, F(X)                               **
C*O   XZERO = A better guess for F(XZERO)=0                           **
C*I   DX    = In case no better formula to guess XZERO,               **
C**           This function will return :                             **
C**           Either XZERO = X + DX, if DX .NE. 0                     **
C**           Or     XZERO = X + F,  if DX .EQ. 0                     **
C**                  this will simulate XZERO = G(X)                  **
C**                  as in the direct substitution method             **
C**                  if set F(X) = G(X) - X                           **
C*I   ISTEP = 1 : For the first call of a new function                **
C**         > 1 : Otherwise                                           **
C**                                                                   **
C**   For ISTEP = 1 :                                                 **
C**   XZERO = X + DX  or  XZERO = X + F                               **
C**                                                                   **
C**   For ISTEP > 1 :                                                 **
C**   XZERO by secant method or                                       **
C**         by false position method (if applicable)                  **
C**         if XZERO by secant method goes outside (XP,XN)            **
C**   ============================================================    **
```

```
      IF (ISTEP.EQ.1) THEN
        DO 10 I=1,11
   10   A(I)=0.0
        FP=-1.0
        FN=+1.0
      ELSE
        DO 20 I=1,11
   20   A(I)=B(I)
      ENDIF
      ST=ST+1
C**   +------------------------------------------------------------+    **
C**   | Save the most newly 3 points                               |    **
C**   +------------------------------------------------------------+    **
      X0=X1
      X1=X2
      X2=X
      F0=F1
      F1=F2
      F2=F
C**   +------------------------------------------------------------+    **
C**   | Find new points (XP,XN) that bracket the root              |    **
C**   +------------------------------------------------------------+    **
      IF (F.LT.0.0) THEN
        XN=X
        FN=F
      ELSE
        XP=X
        FP=F
      ENDIF
      IGET=0
      XX=0.0
C**   +------------------------------------------------------------+    **
C**   | For slow convergence due to multiple roots                 |    **
C**   | Then use secant method on U(X)=F(X)/F'(X)=0                |    **
C**   | After 13 iterations (13 in statement below, >=3)           |    **
C**   +------------------------------------------------------------+    **
      IF (ST.GE.13) THEN
        IF ((F2-F1).NE.0.0.AND.(F1-F0).NE.0.0) THEN
          U1=F1*(X1-X0)/(F1-F0)
          U2=F2*(X2-X1)/(F2-F1)
          IF ((U2-U1).NE.0.0) THEN
            XX=X2-U2*(X2-X1)/(U2-U1)
            IGET=4
          ENDIF
        ENDIF
      ENDIF
C**   +------------------------------------------------------------+    **
C**   | If we have at least 2 functions and no XX get              |    **
C**   | Then use secant method on F(X)=0                           |    **
C**   +------------------------------------------------------------+    **
      IF (ST.GE.2.AND.IGET.EQ.0) THEN
        IF ((F2-F1).NE.0.0) THEN
          XX=X2-F2*(X2-X1)/(F2-F1)
          IGET=3
        ENDIF
      ENDIF
C**   +------------------------------------------------------------+    **
C**   | If root is bracketed by (XP,XN)                            |    **
C**   | And if no XX get or XX goes outside (XP,XN)                |    **
C**   | Then use false position method by (XP,XN)                  |    **
```

```
C**   +--------------------------------------------------------+    **
      IF (FP.GE.0.0.AND.FN.LT.0.0) THEN
        IF (IGET.EQ.0.OR.(XX-XP)*(XX-XN).GT.0.0) THEN
          XX=XP-FP*(XP-XN)/(FP-FN)
          IGET=2
        ENDIF
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | If secant method & false position method cannot be used: |    **
C**   | If DX.NE.0 : Arbitrary set XX=X+DX                     |    **
C**   |    Else    : Set XX=X+F(X)                             |    **
C**   |              To simulate direct substitution method    |    **
C**   |              If set F(X)=G(X)-X in calling program      |    **
C**   +--------------------------------------------------------+    **
      IF (IGET.EQ.0) THEN
        IF (DX.NE.0.0) THEN
          XX=X+DX
        ELSE
          XX=X+F
        ENDIF
        IGET=1
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | For debug                                              |    **
C**   +--------------------------------------------------------+    **
      X0=IGET
      F0=XX
C**   ----------------------------------------------------------    **
      DO 30 I=1,11
   30 B(I)=A(I)
      XZERO=XX
      RETURN
      END
*************************************************************************
```

［表六］ 呼叫XZERO之求根副程式

```
*************************************************************************
      SUBROUTINE FROOT(XX,FX,F,XINI,DX,EPS,ITMAX,FLMT)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION BUF(11)
C**   ============================================================    **
C**   Find root of F(XX)=0 from the initial guess XINI                **
C**   ============================================================    **
                       NSTP=ITMAX
      IF(NSTP.EQ.0)    NSTP=12
                       ERRX=EPS
      IF(ERRX.EQ.0.0) ERRX=0.000001
      IF(XINI.NE.0.0) ERRX=ERRX*ABS(XINI)
                       FLMY=FLMT
      IF(FLMY.EQ.0.0) FLMY=1.0E30
C**   ----------------------------------------------------------    **
      XN=XINI
      DO 20 ISTEP=1,NSTP
        XX=XN
        FX=F(XX)
        XN=XZERO(XX,FX,DX,ISTEP,BUF)
        WRITE(*,'('' FROOT''/(1X,1P6E13.6))') BUF
        IF(DABS(XN-XX).LE.ERRX) RETURN
```

```
        IF(DABS(FX).GT.FLMY) RETURN
   20 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE GROOT(XN,XX,G,XINI,EPS,ITMAX)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION BUF(11)
      ZERO=0.0
C**   ============================================================    **
C**   Find root of XX=G(XX) from the initial guess XINI               **
C**   ============================================================    **
                      NSTP=ITMAX
      IF(NSTP.EQ.0)   NSTP=12
                      ERRX=EPS
      IF(ERRX.EQ.0.0) ERRX=0.000001
      IF(XINI.NE.0.0) ERRX=ERRX*ABS(XINI)
C**   ------------------------------------------------------------    **
      XN=XINI
      DO 20 ISTEP=1,NSTP
        XX=XN
        XN=G(XX)
        XN=XZERO(XX,XN-XX,ZERO,ISTEP,BUF)
        WRITE(*,'('' DIRSUB''/(1X,1P6E13.6))') BUF
        IF(DABS(XN-XX).LE.ERRX) RETURN
   20 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE SROOT(XX,FX,F,XINI,XFIN,DX,EPS,ITMAX,FLMT)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION BUF(11)
C**   ============================================================    **
C**   Search root of F(XX)=0 from XINI to XFIN step DX                **
C**   ============================================================    **
                      NSTP=ITMAX
      IF(NSTP.EQ.0)   NSTP=12
                      ERRX=EPS
      IF(ERRX.EQ.0.0) ERRX=0.000001
      IF(XINI.NE.0.0) ERRX=ERRX*ABS(XINI)
                      FLMY=FLMT
      IF(FLMY.EQ.0.0) FLMY=1.0E30
                      DXX =(XFIN-XINI)/16
      IF(DXX .NE.0.0) DXX =SIGN(ABS(DX),XFIN-XINI)
C**   ------------------------------------------------------------    **
      XN=XINI
      FX=1.0
      DO 20 ISTEP=1,NSTP
   10   XX=XN
        FO=FX
        FX=F(XX)
C**     +----------------------------------------------------+    **
C**     | Search interval with root                          |    **
C**     +----------------------------------------------------+    **
        IF(ISTEP.EQ.2) THEN
          IF((XFIN-XX)*(XINI-XX).GT.0.0) THEN
            WRITE(*,'('' NO ROOT BETWEEN'',1P2E15.6)') XINI,XFIN
            RETURN
```

```
          ENDIF
          IF(FX*FO.GT.0.0) THEN
            XN=XZERO(XX,FX,DXX,1,BUF)
            WRITE(*,'('' SROOT''/(1X,1P6E13.6))') BUF
            GO TO 10
          ENDIF
        ENDIF
C**     ------------------------------------------------------------    **
        XN=XZERO(XX,FX,DXX,ISTEP,BUF)
        WRITE(*,'('' SROOT''/(1X,1P6E13.6))') BUF
        IF(DABS(XN-XX).LE.ERRX) RETURN
        IF(DABS(FX).GT.FLMY) RETURN
   20 CONTINUE
      RETURN
      END
*************************************************************************
```

1.8 多元聯立非線性方程式之解法

前面各節所述方法僅涉及一個未知數即一元之情形，這些方法並非皆可適用於多元聯立之情形。其中直接代入法與牛頓-瑞福生法可以直接延伸使用於多元。 割線法亦可使用於多元之情形，但其理論基礎不像一元那麼單純，因此不擬在本章介紹。可惜的是，具可靠優點的假位法（與半區間法）就目前所知仍無法適用於多元。不過一元的假位法與割線法仍然可以間接使用於本章將介紹之遞迴運算法與反覆試算法。該二法均將N元聯立方程式視為N個一元方程式求解。以下將以三個未知數為例說明各種解法。

1.9 多元直接代入法

直接代入法在多元情形亦為反覆試算法之最基本型式，該法係將原來一般非線性方程式(1.24)，改寫成式(1.25)之型式。

$$\begin{aligned} f_1(x,y,z) &= 0 \\ f_2(x,y,z) &= 0 \\ f_3(x,y,z) &= 0 \end{aligned} \tag{1.24}$$

$$\begin{aligned} x &= g_1(x,y,z) \\ y &= g_2(x,y,z) \\ z &= g_3(x,y,z) \end{aligned} \tag{1.25}$$

若$(x,y,z)=(\alpha,\beta,\gamma)$為方程式$(1.24)$之一根，則其亦為式$(1.25)$之一根，即若$f_1(\alpha,\beta,\gamma)=0,\ f_2(\alpha,\beta,\gamma)=0,\ f_3(\alpha,\beta,\gamma)=0$，則$\alpha=g_1(\alpha,\beta,\gamma),\ \beta=g_2(\alpha,\beta,\gamma),\ \gamma=g_3(\alpha,\beta,\gamma)$。此$(\alpha,\beta,\gamma)$值即為所欲求解之值，當然解題之前無法預知，不過我們可以循一定途徑，由一假設之值開始可逐步逼近所要之(α,β,γ)值。首先假設$(x,y,z)=(x_o,y_o,z_o)$為起始值，代入式(1.25)等號右邊算得$(g_1(x_o,y_o,z_o),g_2(x_o,y_o,z_o),g_3(x_o,y_o,z_o))$之值，若$(x_o,y_o,z_o)\neq(g_1(x_o,y_o,z_o),g_2(x_o,y_o,z_o),g_3(x_o,y_o,z_o))$，則$(x_o,y_o,z_o)$非為所求之根，令其為$(x_1,y_1,z_1)$，即$x_1=g_1(x_o,y_o,z_o),y_1=g_2(x_o,y_o,z_o),z_1=g_3(x_o,y_o,z_o)$，再以$(x,y,z)=(x_1,y_1,z_1)$代入式$(1.25)$等號右邊算得$(x_2,y_2,z_2)=(g_1(x_1,y_1,z_1),g_2(x_1,y_1,z_1),g_3(x_1,y_1,z_1))$，如此重覆以下式計算，可連續算得$(x_1,y_1,z_1),(x_2,y_2,z_2),...,(x_n,y_n,z_n)$之值如下：

$$\begin{aligned} x_{i+1} &= g_1(x_i,y_i,z_i) \\ y_{i+1} &= g_2(x_i,y_i,z_i), \qquad i=0,1,...,n-1 \\ z_{i+1} &= g_3(x_i,y_i,z_i) \end{aligned} \tag{1.26}$$

若上述重覆計算最後可得(x_n,y_n,z_n)等於或近似等於$(x_{n-1},y_{n-1},z_{n-1})$，或$|x_n-x_{n-1}|<\epsilon$，$|y_n-y_{n-1}|<\epsilon$且$|z_n-z_{n-1}|<\epsilon$（其中$\epsilon$為一很小值，可用以控制精度），則$(x_n,y_n,z_n)$即等於或近似等於所欲求得之$(\alpha,\beta,\gamma)$，亦即$(x_n,y_n,z_n)$可收斂至所求之根$(\alpha,\beta,\gamma)$。上述重覆運算並不一定均能收斂至所求之根。只有在$|J(\frac{g_1,g_2,g_3}{x,y,z})|_{(x,y,z)=(\alpha,\beta,\gamma)}<1$時，數列$(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots,(x_n,y_n,z_n)$才會收斂。其中$J$為賈柯比矩陣(Jacobian)。

1.10 多元牛頓-瑞福生法

解聯立非線性方程式之牛頓-瑞福生法，可由下列泰勒展開式（已略去高次項）令$f_j(x_{i+1},y_{i+1},z_{i+1})=0,\ j=1,2,3$，即得可用以解$(x_{i+1},y_{i+1},z_{i+1})$之聯立線性方程式。

$$\begin{Bmatrix} f_1(x_{i+1},y_{i+1},z_{i+1}) \\ f_2(x_{i+1},y_{i+1},z_{i+1}) \\ f_3(x_{i+1},y_{i+1},z_{i+1}) \end{Bmatrix} = \begin{Bmatrix} f_1(x_i,y_i,z_i) \\ f_2(x_i,y_i,z_i) \\ f_3(x_i,y_i,z_i) \end{Bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix}_i \begin{Bmatrix} x_{i+1}-x_i \\ y_{i+1}-y_i \\ z_{i+1}-z_i \end{Bmatrix} \tag{1.27}$$

牛頓-瑞福生法在根的附近通常收斂很快，其收斂階數為2。但亦可能不收斂。其程式如[表七]所示，該程式須呼叫JAC副程式與FUN函

數程式以計算式(1.27)中之矩陣(稱賈柯比(Jacobian)矩陣)與各函數在(x_i, y_i, z_i)點之值。解線性聯立方程式之副程式$LUFPDC$與$LUFPSB$請參考第三章。在二個或更多未知數之情形，並無類似假位法或半區間法，可以由函數值之正負以確定根在某一範圍之方法，因此也就不容易找到能夠保証會收斂且收斂速度快的方法了。

［表七］ 牛頓-瑞福生法求根副程式

```
************************************************************************
      SUBROUTINE NROOTN(N,FX,XX,X0,ERR,D,LI,LJ,JAC,FUN)
C**   ============================================================    **
      REAL*8  X0(N),ERR(N)
      REAL*8  FX(N),XX(N),D(N,N)
      INTEGER LI(N),LJ(N),N,I,ITER,NITER/500/
C**   ============================================================    **
C*I   N        = Number of variables                                  **
C*O   FX(N)    = Function values                                      **
C*O   XX(N)    = Root find by this subroutine                         **
C*I   X0(N)    = Initial values for the first try                     **
C*I   ERR(N)   = Tolerance of X(N)                                    **
C*W   D(N,N)   = Jacobian matrix                                      **
C*W   LI(N)    = Row permutation index                                **
C*W   LJ(N)    = Column permutation index                             **
C*S   JAC(X,D) = Subroutine to calculate D(N,N) for given X(N)        **
C*F   FUN(X,I) = Function   to calculate F_I(X) for given X(N) & I    **
C**   ============================================================    **
      DO 10 I=1,N
   10 XX(I)=X0(I)
      DO 80 ITER=1,NITER
C**   +----------------------------------------------------------+    **
C**   | Compute FX(N),D(N,N)                                     |    **
C**   +----------------------------------------------------------+    **
      DO 40 I=1,N
   40 FX(I)=FUN(XX,I)
      CALL JAC(XX,D,N)
C**   ------------------------------------------------------------    **
C*    DO 45 I=1,N
C* 45 WRITE(*,'(I4,1P5E15.6)') I,(D(I,J),J=1,N)
      WRITE(*,'(I4,1P5E15.6)') ITER,(FX(I),I=1,N)
C**   +----------------------------------------------------------+    **
C**   | Solve -DX from  D(N,N)*DX(N) = -FX(N)                    |    **
C**   | Get XX(N).new = XX(N).old + DX                           |    **
C**   +----------------------------------------------------------+    **
      CALL LUFPDC(D,LI,LJ,N,N)
      CALL LUFPSB(D,FX,LI,LJ,N,N)
      DO 50 I=1,N
   50 XX(I)=XX(I)-FX(I)
C**   ------------------------------------------------------------    **
C*    DO 55 I=1,N
C* 55 WRITE(*,'(I4,1P5E15.6)') I,(D(I,J),J=1,N)
      WRITE(*,'(I4,1P5E15.6)') ITER,(FX(I),I=1,N)
C**   +----------------------------------------------------------+    **
C**   | Convergence check                                        |    **
C**   +----------------------------------------------------------+    **
      DO 70 I=1,N
      IF(DABS(FX(I)).GT.ERR(I)) GO TO 80
```

```
   70 CONTINUE
      RETURN
   80 CONTINUE
      WRITE(*,*) ' THE NUMBER OF ITERATIONS EXCEEDS',NITER
      RETURN
      END
************************************************************************
```

1.11 遞迴運算法

對於一元非線性方程式，由於可以配合割線法與假位法而成為一快速而可靠的方法，因此以下提供一種完全以解一元非線性方程式的方式來解多元聯立非線性方程式的方法。這個方法具有前述一元方法的可靠性的優點，但未知數愈多則效率會變差，因此建議使用在未知數少於10之情形為宜。如果解一元問題平均需試算M次，則解N元的問題約需試算M^N次。以$M = 4.642$為例：如$N = 3$，則$M^N = 100$；如$N = 6$，則$M^N = 10000$；如$N = 9$，則$M^N = 10^6$。

遞迴運算法之運算過程為：由下列一元方程式求x

$$F_1(x) = f_1(x, y(x), z(x, y(x))) = 0 \tag{1.28}$$

式中$y(x)$須滿足下列一元方程式（y為未知，x為已知參數）：

$$F_2(y; x) = f_2(x, y(x), z(x, y(x))) = 0 \tag{1.29}$$

而$z(x, y(x))$須滿足下列一元方程式（z為未知，x與y為已知參數）：

$$F_3(z; x, y) = f_3(x, y(x), z(x, y(x))) = 0 \tag{1.30}$$

注意上式中：$F_1(x) = 0$僅x為未知數；$F_2(y; x) = 0$僅y為未知數，x視為已知之參數；$F_3(z; x, y) = 0$僅z為未知數，x與y均視為已知之參數。在計算$F_1(x)$之值時，須先求得滿足$F_2(y; x) = 0$之y及滿足$F_3(z; x, y) = 0$之z，同樣在計算$F_2(y; x)$之值時，亦須先求得滿足$F_3(z; x, y) = 0$之z。

以$N = 3$為例，其程式可如[表八]所示。對於任意N元聯立方程式，其程式可如[表九]所示。請仔細比較[表八]與[表九]，[表九]雖然比[表八]簡短許多，但卻可以做不限圈數的迴圈，即[表八]僅做固定三個迴圈；而[表九]可做任意N個迴圈。如果採用遞迴(呼叫)方式(recursive method)處理，程式流程會較清楚，但目前大部分符傳編譯器仍未允許採

用遞迴呼叫方式，故[表九]之程式並未採用遞迴呼叫方式，但卻已經引用了遞迴的觀念，因此該程式或可做為處理遞迴的一個範例。

［表八］ 遞迴運算法之三元方程式求根副程式

```
*************************************************************************
      SUBROUTINE FROOTN(N,FX,XX,XN,X0,DX,ERR,FLM,STEP,BUF,FUN)
C**   ============================================================    **
      REAL*8  X0(N),DX(N),ERR(N),FLM(N)
      REAL*8  FX(N),XX(N),XN(N),BUF(11,N)
      INTEGER STEP(N),N,I,NSTEP/12/
C**   ============================================================    **
      IF(N.NE.3) STOP 'FOR N=3 ONLY'
      DO 5 I=1,3
    5 XN(I)=X0(I)
C**   ------------------------------------ LOOP 1 for F_1(XX(1))=0    **
      STEP(1)=1
   10 XX(1)=XN(1)
C**   ------------------------------------ LOOP 2 for F_2(XX(2))=0    **
      STEP(2)=1
   20 XX(2)=XN(2)
C**   ------------------------------------ LOOP 3 for F_3(XX(3))=0    **
      STEP(3)=1
   30 XX(3)=XN(3)
      FX(3)=FUN(XX,3)
      XN(3)=XZERO(XX(3),FX(3),DX(3),STEP(3),BUF(1,3))
      IF(ABS(XN(3)-XX(3)).LE.ERR(3)) GO TO 110
      IF(ABS(FX(3)).GT.FLM(3)) GO TO 310
      STEP(3)=STEP(3)+1
      IF(STEP(3).LE.NSTEP) GO TO 30
C**   ------------------------------------------- End of LOOP 3    **
  110 FX(2)=FUN(XX,2)
      XN(2)=XZERO(XX(2),FX(2),DX(2),STEP(2),BUF(1,2))
      IF(ABS(XN(2)-XX(2)).LE.ERR(2)) GO TO 210
      IF(ABS(FX(2)).GT.FLM(2)) GO TO 310
      STEP(2)=STEP(2)+1
      IF(STEP(2).LE.NSTEP) GO TO 20
C**   ------------------------------------------- End of LOOP 2    **
  210 FX(1)=FUN(XX,1)
      XN(1)=XZERO(XX(1),FX(1),DX(1),STEP(1),BUF(1,1))
      IF(ABS(XN(1)-XX(1)).LE.ERR(1)) GO TO 310
      IF(ABS(FX(1)).GT.FLM(1)) GO TO 310
      STEP(1)=STEP(1)+1
      IF(STEP(1).LE.NSTEP) GO TO 10
C**   ------------------------------------------- End of LOOP 1    **
  310 RETURN
      END
*************************************************************************
```

［表九］ 遞迴運算法之求根副程式

```
*************************************************************************
      SUBROUTINE FROOTN(N,FX,XX,XN,X0,DX,ERR,FLM,STEP,BUF,FUN)
C**   ============================================================    **
      REAL*8  X0(N),DX(N),ERR(N),FLM(N)
      REAL*8  FX(N),XX(N),XN(N),BUF(11,N)
      INTEGER STEP(N),N,I,NSTEP/12/
C**   ============================================================    **
```

```
C*I   N         = Number of variables                                  **
C*O   FX(N)     = Function values                                      **
C*O   XX(N)     = Root find by this subroutine                         **
C*W   XN(N)     = New values of X(N) during iteration                  **
C*I   X0(N)     = Initial values for the first try                     **
C*I   DX(N)     = Increment values for the second try                  **
C*I   ERR(N)    = Tolerance of XX(N)                                   **
C*I   FLM(N)    = Limited function values of FX(N)                     **
C*W   STEP(N)   = Current iteration step for each variable             **
C*W   BUF(11,N) = Buffers                                              **
C*F   FUN(X,I)  = Function to calculate F_I(X) for given X(N) & I      **
C**   ========================================================         **
      DO 10 I=1,N
   10 XN(I)=X0(I)
C**   ------------------------------------------------------------     **
      I=0
   50 I=I+1
C**   +----------------------------------------------------------+     **
C**   | Enter I-th LOOP : DO STEP(I)=1,NSTEP                     |     **
C**   +----------------------------------------------------------+     **
      WRITE(*,'(I3)') I
      STEP(I)=1
   60 XX(I)=XN(I)
      IF(I.LT.N) GO TO 50
C**   ------------------------------------------------------------     **
   70 FX(I)=FUN(XX,I)
      XN(I)=XZERO(XX(I),FX(I),DX(I),STEP(I),BUF(1,I))
C     WRITE(*,'('' I,XX,FX'',I3,1P2E15.6)') I,XX(I),FX(I)
      IF(DABS(XN(I)-XX(I)).LE.ERR(I)) GO TO 80
      IF(DABS(FX(I)).GT.FLM(I)) GO TO 90
      STEP(I)=STEP(I)+1
      IF(STEP(I).LE.NSTEP) GO TO 60
C**   +----------------------------------------------------------+     **
C**   | Leave I-th LOOP                                          |     **
C**   +----------------------------------------------------------+     **
   80 I=I-1
      WRITE(*,'(3I3,1PE15.6)') I,STEP(I),STEP(I+1),FX(I+1)
      NT=NT+STEP(I+1)
      IF(I.GE.1) GO TO 70
   90 CONTINUE
      WRITE(*,'('' NUMBER OF FUNCTION EVALUATIONS :'',I9)') NT
      RETURN
      END
*************************************************************************
```

1.12 反覆試算法

對於某些特定問題，如非線性函數經過特別安排的情形，則可能可以用下列方法求解：

$$\begin{aligned} F_1(x) &= f_1(x;y,z) = 0 \\ F_2(y) &= f_2(y;x,z) = 0 \\ F_3(z) &= f_3(z;x,y) = 0 \end{aligned} \tag{1.31}$$

上列三個方程式，每個方程式都僅視其中一個變數為未知數，其餘變數均為已知，但其餘變數並非固定值，係由其它方程式決定，因此必須由下式以反覆運算方式求解。

$$\begin{aligned} F_1(x_{i+1}) &= f_1(x_{i+1}; y_i, z_i) = 0 \\ F_2(y_{i+1}) &= f_2(y_{i+1}; x_{i+1}, z_i) = 0 \\ F_3(z_{i+1}) &= f_3(z_{i+1}; x_{i+1}, y_{i+1}) = 0 \end{aligned} \tag{1.32}$$

這個方法基本上與解聯立線性方程式之高斯-賽德(Gauss-Siedel)反覆試算法類似。 該法之試算次數不會像前一方法一樣隨著未知數的增加而成冪次方的方式增加。甚至可能不隨未知數的增加而增加試算次數。但該法之缺點為不一定會收斂，也就是不一定可以用該法求解。前面所提到的特別安排，最簡單也是最起碼的是決定用那一個方程式解那一個未知數。例如決定用那一個式子當做式(1.24)中之$F_1(x)$用以解未知數x等。$f_1(x,y,z)=0$選為求解x之原則為f_1對x之偏微分在根附近的值的絕對值要大於對其他未知數之偏微分值的絕對值。如能大於其他偏微分值之絕對值之和，且每一個未知數均如此，則該法即會收斂。

在[圖一]中之直接代入法，可視為二元聯立方程式，即二個未知數(x,y)須滿足聯立式$y-x=0$及$y-g(x)=0$，再利用本節方法求解。若$g'(\alpha)<1$：則以$F_1(x_{i+1})=y_i-x_{i+1}=0$，$F_2(y_{i+1})=y_{i+1}-g(x_{i+1})=0$求解會收斂，此種求解過程相當於直接代入法。反之，若$g'(\alpha)>1$：則可改以$F_1(x_{i+1})=y_i-g(x_{i+1})=0$，$F_2(y_{i+1})=y_{i+1}-x_{i+1}=0$求解仍然會收斂，其收斂"路線"則正好與圖示箭頭方向相反。不過此時由已知之y_i以$y_i-g(x_{i+1})=0$解x_{i+1}須利用其它方法求解，已經不屬於直接代入法。

[表十]為前述方法的程式，此程式之流程比[表九]簡單，且不必用遞迴方式處理，但須有一反覆運算的迴圈DO 80。

[表十] 反覆試算法之求根副程式

```
************************************************************************
      SUBROUTINE GSITER(N,FX,XX,XN,X0,DX,ERR,FLM,XI,BUF,FUN)
C**   ============================================================    **
      REAL*8  X0(N),DX(N),ERR(N),FLM(N)
      REAL*8  FX(N),XX(N),XN(N),XI(N),BUF(11)
      INTEGER STEP,N,I,NSTEP/12/,ITER,NITER/500/
C**   ============================================================    **
C*I   N         = Number of variables                                  **
C*O   FX(N)     = Function values                                      **
C*O   XX(N)     = Root find by this subroutine                         **
C*W   XN(N)     = New values of XX(N) during equation solving          **
```

```
C*I   X0(N)     = Initial values for the first try                    **
C*I   DX(N)     = Increment values for the second try                 **
C*I   ERR(N)    = Tolerance of XX(N)                                  **
C*I   FLM(N)    = Limited function values of FX(N)                    **
C*W   XI(N)     = Initial values of XX(N) during iteration            **
C*W   BUF(11)   = Buffers                                             **
C*F   FUN(X,I)  = Function to calculate F_I(X) for given X(N) & I     **
C**   ================================================================**
      DO 10 I=1,N
      XX(I)=X0(I)
   10 XN(I)=X0(I)
C**   ----------------------------------------------------------------**
      DO 80 ITER=1,NITER
C**   ----------------------------------------------------------------**
      DO 60 I=1,N
      XI(I)=XN(I)
C**   +--------------------------------------------------------------+**
C**   | Find XX(I) such that FX(I)=F_I(XX)=0                          |**
C**   +--------------------------------------------------------------+**
      DO 50 STEP=1,NSTEP
      XX(I)=XN(I)
      FX(I)=FUN(XX,I)
      XN(I)=XZERO(XX(I),FX(I),DX(I),STEP,BUF)
      IF(DABS(XN(I)-XX(I)).LE.ERR(I)) GO TO 60
      IF(DABS(FX(I)).GT.FLM(I)) GO TO 90
   50 CONTINUE
C* 55 WRITE(*,'(3I4,1P2E15.6)') ITER,I,STEP,XX(I),FX(I)
   60 CONTINUE
C**   +--------------------------------------------------------------+**
C**   | Convergence check                                            |**
C**   +--------------------------------------------------------------+**
      DO 70 I=1,N
      IF(DABS(XX(I)-XI(I)).GT.ERR(I)) GO TO 80
   70 CONTINUE
      GO TO 90
C**   ----------------------------------------------------------------**
   80 CONTINUE
      WRITE(*,*) ' THE NUMBER OF ITERATIONS EXCEEDS',NITER
   90 RETURN
      END
*************************************************************************
```

1.13 牛頓–瑞福生法之幾何意義

為求簡化，本節僅以二個未知數為例說明，理論上可適用於N個未知數之情形。將$f_1(x,y)=0$及$f_2(x,y)=0$視為在(x,y)之二維平面上之二條曲線，則該二條曲線之交點(α,β)即為聯立式之根。現令$z=f_1(x,y)$與$z=f_2(x,y)$，則其可視為在(x,y,z)之三維空間內的二個曲面。這二個曲面與$z=0$之平面之二條交線即為前述在(x,y)之二維平面上之二條曲線。基本上，若沿著$z=f_1(x,y)$與$z=f_2(x,y)$二曲面在三維空間內的交線搜尋，則在$z=0$處應可找到聯立方程式的根(α,β)。

要在空間內找到某一點落在二曲面的交線上本來並不是很容易。但是如果將曲面對z方向比例調整為$z = s_1 f_1(x,y)$與$z = s_2 f_2(x,y)$，則對任意(x_i, y_i)座標，令$z_1 = s_1 f_1(x_i, y_i)$，$z_2 = s_2 f_2(x_i, y_i)$，並取$s_1 = 1/f_1(x_i, y_i)$，$s_2 = 1/f_2(x_i, y_i)$使$z_1 = z_2 = 1$，則$(x_i, y_i, 1)$即落在二曲面之交線上。

如欲自$(x_i, y_i, 1)$沿著曲面之交線搜尋，則搜尋方向(dx, dy, dz)應為交線的切線方向。該切線因同時落在二曲面上，故該切線必須與二曲面之法線方向$(s_1 \partial f_1/\partial x,\ s_1 \partial f_1/\partial y,\ -1)$與$(s_2 \partial f_2/\partial x,\ s_2 \partial f_2/\partial y,\ -1)$垂直，即$(dx, dy, dz)$由下式求得：

$$\begin{bmatrix} s_1 \frac{\partial f_1}{\partial x} & s_1 \frac{\partial f_1}{\partial y} & -1 \\ s_2 \frac{\partial f_2}{\partial x} & s_2 \frac{\partial f_2}{\partial y} & -1 \end{bmatrix}_i \begin{Bmatrix} dx \\ dy \\ dz \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{1.33}$$

上式分別除以s_1與s_2，亦即乘以$f_1(x_i, y_i)$與$f_2(x_i, y_i)$，並將dz項移至等號右邊，即得

$$\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}_i \begin{Bmatrix} dx \\ dy \end{Bmatrix} = \begin{Bmatrix} f_1(x_i, y_i) \\ f_2(x_i, y_i) \end{Bmatrix} dz \tag{1.34}$$

此式所得之(dx, dy)即與牛頓–瑞福生法之(dx, dy)方向相同。令上式中$dz = -1$，則與牛頓–瑞福生法之運算式相同。

經由上述對牛頓–瑞福生法賦與一項幾何說明後，應該對該法有較深一層的認識，而能在試算過程中做更合理的掌握。例如(dx, dy)的方向決定後，可調整步長大小使其不至於太遠離曲線的交線。或者亦可根據$Err = \sum_{j=1}^{2} s_j^2 f_j^2(x_i + \lambda dx, y_i + \lambda dy)$之最小值決定步長$\lambda$。在沿線搜尋過程中可能會找到$z$為局部極小值之點，這些點的存在，通常就是造成牛頓–瑞福生法不會收斂的主要原因：試算點經常會在該點附近跳動而不收斂。此時之處理方式可以為：(1)放棄該點另找起始點試算；(2)沿著交線繼續尋找根的位置，但如照前述Err之最小值決定步長，則須暫時改為尋找Err之最大值，之後再回復為尋找Err之最小值。

1.14 弧長控制法

弧長控制法(Arc length control)為在結構幾何非線性分析中常用之方法，結構幾何非線性分析 主要在計算結構承受一組變化荷重$\{P\} = \{{}_1p,\ {}_2p, \ldots, {}_Np\}$作用時對應變位$\{U\} = \{{}_1u,\ {}_2u, \ldots, {}_Nu\}$之變化情形。以勁

度法分析時，荷重與變位向量之階數N為結構變位之自由度（數目）。設$\{U\}$與$\{P\}$滿足下列之力平衡關係，式中$\{F\}=\{{}_1f,\,{}_2f,\ldots,{}_Nf\}$。

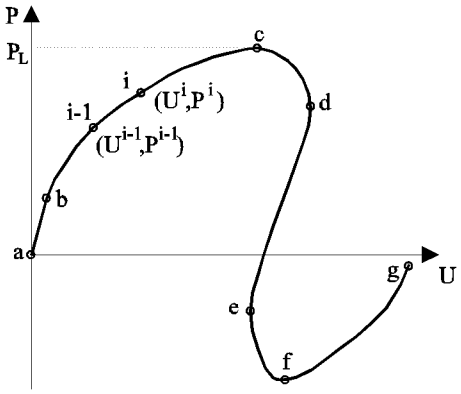
$$\{F(\{U\},\{P\})\}=\{0\} \tag{1.35}$$

若採用牛頓法，則於第n次試算時，上式以泰勒展開式取一階近似為

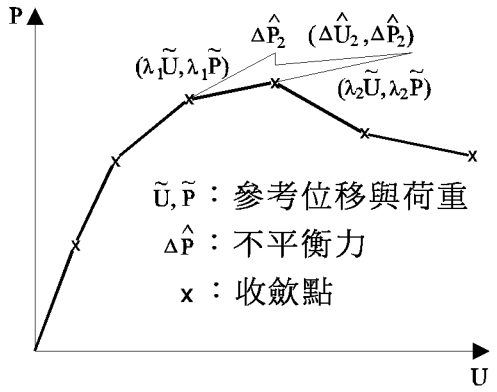
$${}_lf(\{U_{n-1}\},\{P_{n-1}\})+\sum_j\frac{\partial_l f}{\partial_j u}\Delta_j u_n+\sum_j\frac{\partial_l f}{\partial_j p}\Delta_j p_n=0 \tag{1.36}$$

式(1.35)中之$\{P\}$為作用在自由度之外力，在隨後之討論視為已知（但亦可考慮荷重隨作用位置改變如跟隨荷重 (follow load)），但非固定值，因此式(1.36)中之$\frac{\partial_l f}{\partial_j p}=-\delta_{lj}$。令$[K_{n-1}]$為$\left[\frac{\partial_l f}{\partial_j u}\right]$在$\{U_{n-1}\},\ \{P_{n-1}\}$時之值，稱為切線勁度。令$\{\Delta\hat{P}_n\}=-\{F_{n-1}\}=-\{F(\{U_{n-1}\},\{P_{n-1}\})\}$，為第$n-1$次試算之不平衡力。並以符號$\{\Delta\tilde{P}_n\}$表示上式中之$\{\Delta_1p_n,\Delta_2p_n,\ldots,\Delta_Np_n\}$，則上式可寫成

$$[K_{n-1}]\{\Delta U_n\}=\{\Delta\hat{P}_n\}+\{\Delta\tilde{P}_n\} \tag{1.37}$$



圖三 荷重變位曲線



圖四 收斂路徑

荷重與變位關係一般可繪成荷重變位曲線 (Load–displacement curve)。該曲線通常有下列數項特性（參考圖三）：

(1)設a–b之間$[K]$為常數，若$\{P\}$保持一定之比例，即可寫成$\{P\}=\lambda\{\check{P}\}$，則$\{U\}=\lambda\{\check{U}\}$，故a–b為一段直線。因此僅需解一次下列聯立式即可。

$$[K]\{\check{U}\}=\{\check{P}\} \tag{1.38}$$

(2)b–c 之間 $[K]$ 為 $\{U\}, \{P\}$ 之非線性函數。此段之計算，通常將曲線分成若干小段，並假設 $\{U^{i-1}\}, \{P^{i-1}\}$ 已經求出，再用反覆試算方式求出 $\{U^i\}, \{P^i\}$ 。若令

$$\{U^i\} = \{U^{i-1}\} + \{\Delta U\}\,, \quad \{P^i\} = \{P^{i-1}\} + \{\Delta \tilde{P}\} \tag{1.39}$$

則 $\{\Delta U\}$ 與 $\{\Delta \tilde{P}\}$ 稱為小段之增量。此增量如以牛頓法求解，則第 n 次試算之結果為

$$\{\Delta U\} = \sum_{k=1}^{n-1}\{\Delta U_k\} + \{\Delta U_n\}\,, \quad \{\Delta \tilde{P}\} = \sum_{k=1}^{n-1}\{\Delta \tilde{P}_k\} + \{\Delta \tilde{P}_n\} \tag{1.40}$$

其中 $\{\Delta U_n\}$ 由式 (1.37) 計算，式中有關之 $\{U_{n-1}\}$ 與 $\{P_{n-1}\}$ 為

$$\{U_{n-1}\} = \{U^{i-1}\} + \sum_{k=1}^{n-1}\{\Delta U_k\}\,, \quad \{P_{n-1}\} = \{P^{i-1}\} + \sum_{k=1}^{n-1}\{\Delta \tilde{P}_k\} \tag{1.41}$$

當 $n \to \infty$ 時，若 $\{\Delta U_n\} \to 0$ ，則可收斂求得 $\{\Delta U\}$ 。

上述做法(以 c 點附近而言)，在 $P \le P_L$ 時，才可能收斂求得 $\{\Delta U\}$ ；但是在 $P > P_L$ 時，因無對應之解 $\{\Delta U\}$ ，故上述之反覆試算無法收斂。解決之法為：i 點之位置不以 $\{\Delta \tilde{P}\}$ 之值決定，而改為由 $i-1$ 至 i 點之弧長為固定值決定，即所謂弧長控制。以 $\{\Delta \tilde{P}\}$ 之值決定者稱為荷重控制。雖然亦可以 $\{\Delta U\}$ 之某一分量為固定值做控制，稱為變位控制，但其在 d 點附近亦會造成類似問題而無收斂之解。因此弧長控制應為最理想之方法。

以弧長控制步長時，$\{\Delta \tilde{P}\}$ 不再為固定值，故以一荷重率 λ 或 λ_n 表示荷重大小為 $\{\Delta \tilde{P}\} = \lambda\{\hat{P}\}$ 或 $\{\Delta \tilde{P}_n\} = \lambda_n\{\hat{P}\}$ 。而弧長之平方可定義為

$$\{\Delta U\}^T\{\Delta U\} + \{\Delta \tilde{P}\}^T\{\Delta \tilde{P}\} = \{\Delta U\}^T\{\Delta U\} + \lambda^2\{\hat{P}\}^T\{\hat{P}\} \tag{1.42}$$

但該弧長之定義會受變位與荷重所用之單位之影響，因二者因次不同。若將 $\Delta_j u,\ \Delta_j p$ 除以其對應之初始值 $\Delta_j u^1,\ \Delta_j p^1$ ，則為無因次之相對值，即不受所用單位之影響。另外各相對值亦可分別再乘以權值 $\sqrt{\Delta_j u^1 \Delta_j p^1}$ ，而得 $\Delta_j \bar{u} = \sqrt{\Delta_j u^1 \Delta_j p^1}(\Delta_j u/\Delta_j u^1)$ 與 $\Delta_j \bar{p} = \sqrt{\Delta_j u^1 \Delta_j p^1}(\Delta_j p/\Delta_j p^1)$ 。注意權值之平方為該自由度之力對結構所作之功之 2 倍，如此即可考慮各分量之重要性。因此弧長之平方即定義為

$$\sum[(\Delta_j \bar{u})^2 + (\Delta_j \bar{p})^2] = \sum(\Delta_j u^1 \Delta_j p^1)\Big[(\frac{\Delta_j u}{\Delta_j u^1})^2 + (\frac{\Delta_j p}{\Delta_j p^1})^2\Big] \tag{1.43}$$

在以後之說明中仍採用式(1.42)之符號，以上式之弧長定義之作法相同。

通常$[K]$矩陣為對稱，如將λ直接做為未知數，將使矩陣變成不對稱，因此採用Batoz與Dhatt之做法將$\{\Delta U_n\}$分成二部分：$\{\Delta\widehat{U}_n\}$與$\{\Delta\widetilde{U}_n\}=\lambda_n\{\widetilde{U}_n\}$分別為對應於$\{\Delta\widehat{P}_n\}$與$\{\Delta\widetilde{P}_n\}=\lambda_n\{\widetilde{P}\}$之解，即

$$\{\Delta U_n\}=\{\Delta\widehat{U}_n\}+\{\Delta\widetilde{U}_n\}=\{\Delta\widehat{U}_n\}+\lambda_n\{\Delta\widetilde{U}_n\} \tag{1.44}$$

$$[K_{n-1}]\{\Delta\widehat{U}_n\}=\{\Delta\widehat{P}_n\}\,,\quad [K_{n-1}]\{\widetilde{U}_n\}=\{\widetilde{P}\} \tag{1.45}$$

因此式(1.40)可寫成

$$\{\Delta U\} \;=\sum_{k=1}^{n-1}\{\Delta U_k\}+\{\Delta U_n\}=\sum_{k=1}^{n-1}\{\Delta U_k\}+(\{\Delta\widehat{U}_n\}+\lambda_n\{\widetilde{U}_n\}) \tag{1.46}$$

$$\{\Delta\widetilde{P}\} \;=\sum_{k=1}^{n-1}\{\Delta\widetilde{P}_k\}+\{\Delta\widetilde{P}_n\}=(\sum_{k=1}^{n-1}\{\Delta P_k\}+\{\Delta\widehat{P}_n\})+\lambda_n\{\widetilde{P}\} \tag{1.47}$$

上式中之$\sum_{k=1}^{n-1}\{\Delta P_k\}=\sum_{k=1}^{n-1}\{\Delta\widetilde{P}_k\}-\{\Delta\widehat{P}_n\}=\sum_{k=1}^{n-1}\{\Delta\widetilde{P}_k\}+\{F(\{U_{n-1}\},\{P_{n-1}\})\}$為用以平衡由$\sum_{k=1}^{n-1}\{\Delta U_k\}$之變位所增加之內力。將式(1.47)代入式(1.42)，令其為固定值L^2，可得

$$||\sum_{k=1}^{n-1}\{\Delta U_k\}+\{\Delta\widehat{U}_n\}+\lambda_n\{\widetilde{U}_n\}||_2^2+||\sum_{k=1}^{n-1}\{\Delta\widetilde{P}_k\}+\lambda_n\{\widetilde{P}\}||_2^2=L^2 \tag{1.48}$$

當$n=1$時：$\{\Delta\widehat{P}_1\}=\{\Delta\widehat{U}_1\}=\{0\}$，因此得$\lambda_1$為

$$\lambda_1=\sqrt{L^2/(||\{\widetilde{U}_1\}||_2^2+||\{\widetilde{P}\}||_2^2)} \tag{1.49}$$

當$n>1$時，式(1.48)展開可得

$$A\lambda_n^2+2B\lambda_n+C=0 \tag{1.50}$$

式中

$$A \;=\; ||\{\widetilde{U}_n\}||_2^2+||\{\widetilde{P}\}||_2^2 \tag{1.51}$$

$$B \;=\; (\sum_{k=1}^{n-1}\{\Delta U_k\}+\{\Delta\widehat{U}_n\})^T\{\widetilde{U}_n\}+(\sum_{k=1}^{n-1}\{\Delta\widetilde{P}_k\})^T\{\widetilde{P}\} \tag{1.52}$$

$$C \;=\; ||(\sum_{k=1}^{n-1}\{\Delta U_k\}+\{\Delta\widehat{U}_n\}||_2^2+||\{\Delta\widehat{U}_n\}||_2^2-L^2 \tag{1.53}$$

式(1.50)之解通常有二根，即有二組$(\{\Delta U\},\{\Delta\tilde{P}\})$之解，由此所得之狀態向量$\{\{U^i\},\{P^i\}\}$與前一點$\{\{U^{i-1}\},\{P^{i-1}\}\}$間之弧長相等，正確之根應為使下列之值為正且為較大者，以使狀態向量之變化較平緩。圖四為弧長控制之收斂情形。

$$\{\Delta U^{i-1}\}^T\{\Delta U^i\}+\{\Delta\tilde{P}^{i-1}\}^T\{\Delta\tilde{P}^i\} \tag{1.54}$$

注意當$[K_{n-1}]$為奇異矩陣(或接近)時，因$||\{\tilde{U}_n\}||_2^2$與$||\{\hat{U}_n\}||_2^2$均很大，故

$$A\approx||\{\tilde{U}_n\}||_2^2\,,\quad B\approx\{\Delta\hat{U}_n\}^T\{\tilde{U}_n\}\,,\quad C\approx||\{\Delta\hat{U}_n\}||_2^2 \tag{1.55}$$

則由Cauchy-Schwarz不等式可得$B^2\le AC$，因此可知式(1.50)無實根之解。下節將對此問題略作討探，並說明處理方法。

1.15 奇異勁度矩陣之處理

本節將探討切線勁度矩陣$[K_{n-1}]$為奇異矩陣之情形。以下說明為簡化符號將省略下標$n-1$。現將荷重向量$\{P\}$與對應變位向量$\{U\}$分為二組：$\{P\}=\{P_1,P_2\}$，$\{U\}=\{U_1,U_2\}$。本節之$\{U\}$代表上節之$\{\Delta\hat{U}\}$或$\{\Delta\tilde{U}\}$；$\{P\}$則對應代表上節之$\{\Delta\hat{P}\}$或$\{\Delta\tilde{P}\}$。因此荷重向量與變位向量之關係可寫成

$$\begin{bmatrix}K_{11}&K_{12}\\K_{21}&K_{22}\end{bmatrix}\begin{Bmatrix}U_1\\U_2\end{Bmatrix}=\begin{Bmatrix}P_1\\P_2\end{Bmatrix} \tag{1.56}$$

若K_{11}非奇異且等於$L_{11}D_{11}L_{11}^T$，其中L_{11}為下三角矩陣，D_{11}為對角矩陣，則以L_{11}及D_{11}做前進代入運算，可得

$$\begin{bmatrix}L_{11}^T&D_{11}^{-1}L_{11}^{-1}K_{12}\\0&K_{22}-K_{21}K_{11}^{-1}K_{12}\end{bmatrix}\begin{Bmatrix}U_1\\U_2\end{Bmatrix}=\begin{Bmatrix}D_{11}^{-1}L_{11}^{-1}P_1\\P_2-K_{21}K_{11}^{-1}P_1\end{Bmatrix} \tag{1.57}$$

上式可用下式之新符號簡化寫成

$$\begin{bmatrix}L_{11}^T&\bar{K}_{12}\\0&\bar{K}_{22}\end{bmatrix}\begin{Bmatrix}U_1\\U_2\end{Bmatrix}=\begin{Bmatrix}\bar{P}_1\\\bar{P}_2\end{Bmatrix} \tag{1.58}$$

(1)若$\bar{K}_{22}$非奇異，則可解出$U_2=\bar{K}_{22}^{-1}\bar{P}_2$，再用$L_{11}^T$做反向代入運算，而得

$$\begin{aligned}\begin{Bmatrix}U_1\\U_2\end{Bmatrix}&=\begin{Bmatrix}L_{11}^{-T}\bar{P}_1-L_{11}^{-T}\bar{K}_{12}\bar{K}_{22}^{-1}\bar{P}_2\\\bar{K}_{22}^{-1}\bar{P}_2\end{Bmatrix}\\&=\begin{Bmatrix}K_{11}^{-1}P_1\\0\end{Bmatrix}+\begin{bmatrix}-K_{11}^{-1}K_{12}\\I_{22}\end{bmatrix}\{\bar{K}_{22}^{-1}\bar{P}_2\}\end{aligned} \tag{1.59}$$

若將荷重向量分成二部分：$\{P\}=\{P'\}+\{P''\}$，如下式

$$\left\{\begin{array}{c} P_1 \\ P_2 \end{array}\right\}=\left\{\begin{array}{c} P_1 \\ K_{21}K_{11}^{-1}P_1 \end{array}\right\}+\left\{\begin{array}{c} 0 \\ P_2-K_{21}K_{11}^{-1}P_1 \end{array}\right\} \tag{1.60}$$

則其對應之二部分變位：$\{U\}=\{U'\}+\{U''\}$，即如式(1.59)所示。

(2)若$\bar{K}_{22}=0$，或當$\bar{K}_{22}$接近奇異時且式(1.50)中之$B^2-AC<0$，則可能有下述二種情況：(2a)$P''\neq 0$；(2b)$P''=0$。茲分別討論如下

(2a)當$P''\neq 0$(即$\bar{P}_2\neq 0$)時，變位向量U''將趨近於無窮大，此時之U'與其相比則太小而可忽略。但為求得有限之變位向量，必須令$\bar{P}_2=0$或非常小之值而將$\bar{K}_{22}^{-1}\bar{P}_2$改為任意設定之有限變位向量$\{U_2\}$，以求得$U=U''$之值如下

$$\{U''\}=\left[\begin{array}{c} -K_{11}^{-1}K_{12} \\ I_{22} \end{array}\right]\{U_2\} \tag{1.61}$$

(2b)當$P''=0$(即$\bar{P}_2=0$)時，可按下列方式求得$U=U'$：

$$\{U'\}=\left\{\begin{array}{c} K_{11}^{-1}P_1 \\ 0 \end{array}\right\} \tag{1.62}$$

上式之解一般稱為分叉解，通常為不想求得的解。因此為了不讓此解出現而故意加入一些所謂的不完美荷重以使$\bar{P}_2\neq 0$而成為前述(2a)之情形，因此而可避免分叉解出現。但由上述分析可知：若出現$\bar{K}_{22}=0$之情形即用式(1.61)求解即可不必借助不完美荷重之方式解決。

當切線勁度矩陣為奇異時，可考慮不在此階段平衡前階段之不平衡力，而留待下階段再平衡。即此階段可設$\{\Delta\widehat{P}_n\}=\{0\}$與$\{\Delta\widetilde{P}_n\}=\{0\}$，因此$\{\Delta\widehat{U}_n\}=\{0\}$，而$\{\Delta\widetilde{U}_n\}=\{\Delta\widetilde{U}_n''\}$係由式(1.61)求得。因此式(1.48)等簡化為

$$||\sum_{k=1}^{n-1}\{\Delta U_k\}+\lambda_n\{\widetilde{U}_n\}||_2^2+||\sum_{k=1}^{n-1}\{\Delta P_k\}||_2^2=L^2 \tag{1.63}$$

$$A = ||\{\widetilde{U}_n\}||_2^2 \tag{1.64}$$

$$B = (\sum_{k=1}^{n-1}\{\Delta U_k\})^T\{\widetilde{U}_n\} \tag{1.65}$$

$$C = ||\sum_{k=1}^{n-1}\{\Delta U_k\}||_2^2-L^2 \tag{1.66}$$

對於非奇異矩陣與奇異矩陣之分解與求解請參考第三章與第四章。

習題

1. 求方程式$x^3 - 2x - 5 = 0$之實根。

2. 求方程式$\cosh x \cos x + 1 = 0$之最小根。

3. 在$XZERO$函數程式中，新的XX點亦可利用通過$(X0, F0)$, $(X1, F1)$, $(X2, F2)$三點之二次曲線與x軸的交點求得，試修改$XZERO$加入此項功能。

4. 在牛頓–瑞福生法中，若$f'(x)$不易求得，則式(1.15)中之$f'(x_i)$可以用下列差分近似之，一般稱此法為割線法(secant method)：

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

   試按上述方法寫一符傳程式。

5. 在多元牛頓–瑞福生法中，若賈柯比矩陣不易直接求得，可類似上題方式以差分近似之。試寫一JAC副程式，以呼叫FUN算得之函數之差分計算賈柯比矩陣。

參考文獻

1. Lin, T.W., Yang, Y.B. and Shiau, H.T., "A Work Weighted State Vector Control Method for Geometrically Nonlinear Analysis," Computers and Structures, Vol.46, No.4, pp.689–694, 1993.

第二章

數值積分法

2.1 前言

許多數學積分式很難，甚至無法以解析方法求得其確切積分解，因此必須以數值方法計算。數值積分法一般並不陌生，最簡單的數值積分法為常用之梯形面積法，其次為辛蒲生法(Simpson's rule)，但程式中則以倫伯格積分法(Romberg table)最常被採用。倫伯格積分法主要以梯形面積法為基礎，連續利用外插值法以獲得較準確的積分值。另外一種數值積分法為高斯積分法(Gaussian quadrature)，該法利用正交函數之特性選用特別的積分點，而可以少用約一半的積分點。本章亦將探討奇異值之積分與無限區間之積分之一些處理方法。最後，並提供一種能夠自動調配積分點分布之積分法及副程式。

2.2 梯形面積法

如欲計算下列積分I

$$I = \int_a^b f(x)\,dx \tag{2.1}$$

可將$a,\ b$間分為n等分，則每一等分之間隔為$h = \frac{b-a}{n}$。將每一間隔內之面積視為梯形，並將該n個梯形面積相加，可得積分I之近似值T_n為

$$T_n = \frac{h}{2}[f(a) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(b)] \tag{2.2}$$

其中$x_i = a + ih$，$i = 0, 1, ..., n$。

2.3 辛蒲生法

計算式(2.1)積分時，如$a,\ b$間仍分為n等分，每一等分之間隔仍為$h=\frac{b-a}{n}$，但將每一單元之面積視為拋物線下之面積，而該拋物線須與原函數曲線在區間二端點及中點等三處相交。然後將該n個單元面積相加，可得積分I之近似值S_n為

$$S_n=\frac{h}{6}[f(a)+4f(x_{1/2})+2f(x_1)+4f(x_{3/2})+\cdots \\ +2f(x_{n-1})+4f(x_{n-1/2})+f(x_n)] \qquad (2.3)$$

其中$x_i=a+ih$，$i=0,\frac{1}{2},1,\frac{3}{2},...,n-1,n-\frac{1}{2},n$。

2.4 倫伯格積分法

由數學推導可證明

$$I=\int_a^b f(x)\,dx$$
$$=\frac{h}{2}[f(a)+2\sum_{i=1}^{n-1}f(a+ih_N)+f(b)]+C_2^{(1)}h^2+C_4^{(1)}h^4+C_6^{(1)}h^6+\cdots \qquad (2.4)$$

其中$h=\frac{b-a}{n}$，$C_2^{(1)},\ C_4^{(1)},\ C_6^{(1)},\ldots$等為僅與$f(x)$及其導函數有關而與$h$無關之係數，且式(2.4)中不含$h$之奇次冪項之係數。

如果取$n=2^{N-1}$，N為任意正整數。令$h_N=\frac{b-a}{n}=\frac{b-a}{2^{N-1}}$，及

$$A_N^{(1)} = \frac{h_N}{2}[f(a)+2\sum_{i=1}^{n-1}f(a+ih_N)+f(b)] \qquad (2.5)$$

則式(2.4)可寫成

$$I = A_N^{(1)}+C_2^{(1)}h_N^2+C_4^{(1)}h_N^4+C_6^{(1)}h_N^6+\cdots \qquad (2.6)$$

注意$A_N^{(1)}$等於梯形面積法分為n等分算得之面積T_n。當h_N很小時，由式(2.6)知$A_N^{(1)}$之誤差約略與h_N^2成正比，因此稱$A_N^{(1)}$或T_n之積分階次為2。

同理，如果取$n=2^N$，則$h_{N+1}=\frac{b-a}{2^N}=\frac{h_N}{2}$，及

$$A_{N+1}^{(1)} = \frac{h_{N+1}}{2}[f(a)+2\sum_{i=1}^{n-1}f(a+ih_{N+1})+f(b)] \qquad (2.7)$$

$$I = A_{N+1}^{(1)}+C_2^{(1)}h_{N+1}^2+C_4^{(1)}h_{N+1}^4+C_6^{(1)}h_{N+1}^6+\cdots \qquad (2.8)$$

或

$$I = A_{N+1}^{(1)} + \frac{C_2^{(1)}}{4}h_N^2 + \frac{C_4^{(1)}}{4^2}h_N^4 + \frac{C_6^{(1)}}{4^3}h_N^6 + \cdots \quad (2.9)$$

由式(2.9)乘以4，再減去式(2.6)可消去含係數$C_2^{(1)}$之項，並解得I值如下式：

$$I = \frac{4A_{N+1}^{(1)} - A_N^{(1)}}{4-1} + \frac{(4^{-1}-1)C_4^{(1)}}{4-1}h_N^4 + \frac{(4^{-2}-1)C_6^{(1)}}{4-1}h_N^6 + \cdots \quad (2.10)$$

令

$$A_N^{(2)} = \frac{4A_{N+1}^{(1)} - A_N^{(1)}}{4-1}, \quad C_4^{(2)} = \frac{(4^{-1}-1)C_4^{(1)}}{4-1}, \quad C_6^{(2)} = \frac{(4^{-2}-1)C_6^{(1)}}{4-1}, \cdots$$

則式(2.10)可簡化為

$$I = A_N^{(2)} + C_4^{(2)}h_N^4 + C_6^{(2)}h_N^6 + C_8^{(2)}h_N^8 + \cdots \quad (2.11)$$

同理，可得

$$I = A_{N+1}^{(2)} + C_4^{(2)}h_{N+1}^4 + C_6^{(2)}h_{N+1}^6 + C_8^{(2)}h_{N+1}^8 + \cdots \quad (2.12)$$

則積分I之近似值$A_N^{(2)}$之積分階數為4。由式(2.12)乘以4^2，再減去式(2.11)可消去$C_4^{(2)}$之項，並解得I值如下式：

$$I = \frac{4^2A_{N+1}^{(2)} - A_N^{(2)}}{4^2-1} + \frac{(4^{-1}-1)C_6^{(2)}}{4^2-1}h_N^6 + \frac{(4^{-2}-1)C_8^{(2)}}{4^2-1}h_N^8 + \cdots \quad (2.13)$$

令

$$A_N^{(3)} = \frac{4^2A_{N+1}^{(2)} - A_N^{(2)}}{4^2-1}, \quad C_6^{(3)} = \frac{(4^{-1}-1)C_6^{(2)}}{4^2-1}, \quad C_8^{(3)} = \frac{(4^{-2}-1)C_8^{(2)}}{4^2-1}, \cdots$$

則式(2.13)可簡化為

$$I = A_N^{(3)} + C_6^{(3)}h_N^6 + C_8^{(3)}h_N^8 + \cdots \quad (2.14)$$

由式(2.6)推導至式(2.11)以至式(2.14)之過程，可看出計算$A_N^{(m+1)}$之通式為

$$A_N^{(m+1)} = \frac{4^mA_{N+1}^{(m)} - A_N^{(m)}}{4^m-1} \quad (2.15)$$

將各$A_N^{(m)}$值列成[表一]，稱倫伯格積分表。

[表一]

$$
\begin{array}{cccccc}
A_1^{(1)} & & & & & \\
A_2^{(1)} & A_1^{(2)} & & & & \\
A_3^{(1)} & A_2^{(2)} & A_1^{(3)} & & & \\
A_4^{(1)} & A_3^{(2)} & A_2^{(3)} & A_1^{(4)} & & \\
A_5^{(1)} & A_4^{(2)} & A_3^{(3)} & A_2^{(4)} & A_1^{(5)} & \\
- & - & - & - & - & -
\end{array}
$$

[表二]

$$
\begin{array}{cccccc}
A(1) & & & & & \\
A(2) & A(1) & & & & \\
A(3) & A(2) & A(1) & & & \\
A(4) & A(3) & A(2) & A(1) & & \\
A(5) & A(4) & A(3) & A(2) & A(1) & \\
- & - & - & - & - & -
\end{array}
$$

計算上表時，有下列幾點應予注意：

1. 先由第一列算起，再算第二列，第三列，...。
2. 每列由左向右之次序計算。
3. 每列之最右近似值$A_1^{(m)}$為該列積分階數$(2m)$最高者；故為最精確值。比較前後兩列之最精確值，可判斷積分值之精確度是否足夠，以決定是否需再繼續算次一列。
4. 每列最左邊數值$A_N^{(1)}$之計算可利用前列最左邊值$A_{N-1}^{(1)}$，而僅需計算前列計算點間之中間點之函數值。

$$
\begin{aligned}
A_N^{(1)} &= \frac{h_N}{2}[f(a)+2\sum_{i=1}^{n-1} f(a+ih_N)+f(b)] \\
&= \frac{1}{2}\frac{h_{N-1}}{2}\{[f(a)+2\sum_{i=2,4}^{n-2} f(a+ih_N)+f(b)]+2\sum_{i=1,3}^{n-1} f(a+ih_N)\} \\
&= \frac{1}{2}\frac{h_{N-1}}{2}\{[f(a)+2\sum_{j=1}^{n/2-1} f(a+jh_{N-1})+f(b)]+2\sum_{i=1,3}^{n-1} f(a+ih_N)\} \\
&= \frac{1}{2}[A_{N-1}^{(1)}+h_{N-1}\sum_{i=1,3}^{n-1} f(a+ih_N)] \qquad (2.16)
\end{aligned}
$$

5. $A_N^{(2)}$等於將$a,\ b$間分成$n=\frac{b-a}{n}$等分，每一等分用辛蒲生法所得之積分值S_n。以$N=1$為例：

$$A_1^{(2)} = \frac{4A_2^{(1)}-A_1^{(1)}}{3}$$

$$= \frac{4}{3}\frac{(b-a)}{4}[f(a)+2f(\frac{a+b}{2})+f(b)] - \frac{1}{3}\frac{(b-a)}{2}[f(a)+f(b)]$$
$$= \frac{(b-a)}{6}[f(a)+4f(\frac{a+b}{2})+f(b)] = S_1$$

6. 程式計算時，如欲節省儲存該表資料之記憶位置，可僅保留最後一列數值，而將其存於一向量A中，如[表二]所示，將$A_N^{(m)}$存於$A(N)$中。例如，已算完第四列而要計算第五列時，可按下述方式進行：

   (a) 由$A_4^{(1)}$（存於$A(4)$）利用式(2.16)計算$A_5^{(1)}$，而存於$A(5)$。

   (b) 由$A_5^{(1)}$及$A_4^{(1)}$（分別存於$A(5)$及$A(4)$）利用式(2.15)計算$A_4^{(2)}$，而存於$A(4)$（因此取代了原存之$A_4^{(1)}$）。

   (c) 由$A_4^{(2)}$及$A_3^{(2)}$（分別存於$A(4)$及$A(3)$）利用式(2.15)計算$A_3^{(3)}$，而存於$A(3)$（因此取代了原存之$A_3^{(2)}$）。

   (d) 由$A_3^{(3)}$及$A_2^{(3)}$利用式(2.15)計算$A_2^{(4)}$，而存於$A(2)$。

   (e) 最後由$A_2^{(4)}$及$A_1^{(4)}$計算$A_1^{(5)}$而存於$A(1)$，結束第五列之計算。

2.5 倫伯格積分法之副程式

[表三]程式係利用前節所述方法及過程計算下列積分：

$$\text{AREA} = \int_{\text{XA}}^{\text{XB}} F(x)\,dx \tag{2.17}$$

該程式中，若指令之前三個字母為'C*1'者，係做為保留及印出倫伯格積分表之用，須要時可將其去掉。

[表四]示一主程式用以呼叫副程式$ROMBRG$計算下列二個積分，並印出倫伯格積分表，如[表五]之輸出結果。

$$\int_0^1 \frac{dx}{\sqrt{1+x^2}}$$
$$\int_{-1}^1 x\,\sin x\,dx$$

[表三] 副程式ROMBRG

```
************************************************************************
      SUBROUTINE ROMBRG(XA,XB,F,AREA,AOLD,EPS)
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
C**   ============================================================    **
C**   AREA = Int_XA^XB F(X) dX
C**   ------------------------------------------------------------    **
      DIMENSION A(20)
C*1   DIMENSION T(20,20)
C**   ============================================================    **
      H=XB-XA
      A(1)=0.5*(F(XA)+F(XB))*H
      AREA=A(1)
C*1   T(1,1)=A(1)
C*1   WRITE(*,'(1X,1PE13.6)') T(1,1)
C**   +----------------------------------------------------------+    **
C**   | COMPUTE T^{(1)}_N                                        |    **
C**   +----------------------------------------------------------+    **
      JJ=1
      DO 30 N=2,20
      AOLD=AREA
      AN=0.0
      X=XA+H*0.5
      DO 15 J=1,JJ
      AN=AN+F(X)
   15 X=X+H
      A(N)=0.5*(A(N-1)+H*AN)
C*1   T(N,1)=A(N)
C**   +----------------------------------------------------------+    **
C**   | COMPUTE T^{(m)}_N                                        |    **
C**   +----------------------------------------------------------+    **
      R=1.0
      DO 20 I=N-1,1,-1
      R=R*4.
      A(I)=A(I+1)+(A(I+1)-A(I))/(R-1.)
C*1   T(N,N+1-I)=A(I)
   20 CONTINUE
C**
C*1   WRITE(*,'(1X,1P10E13.6)') (T(N,M),M=1,N)
C**
      AREA=A(1)
      IF(DABS(AOLD-AREA).LE.EPS*DABS(AREA)) RETURN
      H=H*0.5
   30 JJ=JJ+JJ
      RETURN
      END
*************************************************************************
```

［表四］ 積分主程式

```
*************************************************************************
C**   PROGRAM FOR ROMBERG INTEGRATION
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL FA,FB
C**   ============================================================    **
      EPS=0.000001
C**   ------------------------------------------------------------    **
      XA=0.0
      XB=1.0
      CALL ROMBRG(XA,XB,FA,AREA,AOLD,EPS)
      WRITE(*,'('' AREA ='',1PE13.6/)') AREA
C**   ------------------------------------------------------------    **
      XA=-1.0
```

```
      XB= 1.0
      CALL ROMBRG(XA,XB,FB,AREA,AOLD,EPS)
      WRITE(*,'('' AREA ='',1PE13.6/)') AREA
C**   ------------------------------------------------------------    **
      STOP
      END
**********************************************************************
      FUNCTION FA(X)
      IMPLICIT REAL*8 (A-H,O-Z)
      FA=1.0/SQRT(1.0+X*X)
      RETURN
      END
**********************************************************************
      FUNCTION FB(X)
      IMPLICIT REAL*8 (A-H,O-Z)
      FB=X*SIN(X)
      RETURN
      END
**********************************************************************
```

［表五］ 積分結果

```
8.535534E-01
8.739903E-01 8.808026E-01
8.795308E-01 8.813776E-01 8.814159E-01
8.809131E-01 8.813739E-01 8.813737E-01 8.813730E-01
8.812585E-01 8.813736E-01 8.813736E-01 8.813736E-01 8.813736E-01
AREA = 8.813736E-01

1.682942E+00
8.414710E-01 5.609807E-01
6.604483E-01 6.001074E-01 6.027158E-01
6.167642E-01 6.022028E-01 6.023425E-01 6.023366E-01
6.059378E-01 6.023290E-01 6.023374E-01 6.023374E-01 6.023374E-01
6.032371E-01 6.023368E-01 6.023374E-01 6.023374E-01 6.023374E-01
6.023374E-01
AREA = 6.023374E-01
```

2.6 高斯積分法

前面各節所介紹之積分法均以等間距的函數值乘一係數後相加而得函數之積分值，即

$$I = \sum_{i=0}^{n} W_i f_i = \sum_{i=0}^{n} W_i f(x_i) \tag{2.18}$$

式中$f_i = f(x_i)$，$x_i = a + ih$，係數W_i稱為權值。如梯形面積法之權值中間各點為h，二端點為$h/2$。事實上，求積分所用函數值之x_i不必為等間距。理論上，如果函數$f(x)$為n次多項式，對一組任意選擇之$(x_0, x_1, \ldots, x_n)$，均可由式(2.21)求得一組對應之權值$(W_0, W_1, \ldots, W_n)$，使式(2.20)算得之

積分為正確值。

$$I = \int_a^b w(x)f(x)\,dx \tag{2.19}$$

$$\doteq \sum_{i=0}^{n} W_i f(x_i) \tag{2.20}$$

式中

$$W_i = \int_a^b w(x)L_i(x)\,dx \tag{2.21}$$

式中$L_i(x)$為拉格蘭治(Lagrange)多項式：

$$L_i(x) = \frac{1}{\pi'(x_i)}\frac{\pi(x)}{x - x_i} \tag{2.22}$$

式中

$$\pi(x) = (x - x_0)(x - x_1)(x - x_2)\cdots(x - x_n) = \Pi_{i=0}^{n}(x - x_i) \tag{2.23}$$

令$p(x)$係由n次多項式之$L_i(x)$乘以$f(x_i)$後相加而得：

$$p(x) = \sum_{i=0}^{n} f(x_i)L_i(x) \tag{2.24}$$

注意式中之$p(x)$亦為n次多項式。利用拉格蘭治多項式之下列性質，可知$p(x_i) = f(x_i)$。但因通過$(x_0, x_1, x_2, \ldots, x_n)$之$n$次曲線為唯一，故若$f(x)$為$n$次多項式，則$f(x) = p(x)$，代入式(2.19)可得式(2.20)及式(2.21)。亦即由此可証知式(2.20)所得之值，對於n次多項式之$f(x)$而言，正好等於式(2.19)之積分。

$$\begin{aligned} L_i(x_j) &= 0, \qquad i \neq j \\ L_i(x_i) &= 1, \qquad i = 0, 1, \cdots, n \end{aligned} \tag{2.25}$$

注意式(2.19)之積分式中多了一項函數$w(x)$，稱為權函數 (weighting function)。以前各節之積分式可視為本節積分式中之$w(x) = 1$之特例，因此本節所述方法同樣亦適用於式(2.1)之積分式。

本節所要介紹的高斯積分法，可以用$n + 1$個點的函數值，以式(2.20)正確求得式(2.19)中之$f(x)$為$2n + 1$次多項式時之積分值。不過此時之x_i不能任意選取。以下將詳細說明x_i與權函數$w(x)$及對應積分範圍$[a, b]$間之關係。

首先介紹一種適用於數值積分之正交函數 (orthogonal function)：對於一特定權函數$w(x)$於一特定區間$[a,b]$，若存在一組n次多項式$p_n(x),\ n=0,1,\ldots$，滿足下列關係，則稱$p_n(x)$為對應於權函數$w(x)$及區間$[a,b]$之正交函數。

$$\int_a^b w(x)p_n(x)p_m(x)\,dx=0, \qquad n\neq m \tag{2.26}$$

$$\int_a^b w(x)p_n(x)p_n(x)\,dx=P_n\neq 0 \tag{2.27}$$

利用上述之正交函數，可以將$2n+1$次多項式$f_{2n+1}(x)$寫成：

$$f_{2n+1}(x) = p_{n+1}(x)g_n(x)+h_n(x) \tag{2.28}$$

$$= p_{n+1}(x)\sum_{i=0}^{n}\beta_i p_i(x)+h_n(x) \tag{2.29}$$

式中$g_n(x)$與$h_n(x)$均為n次多項式。而$g_n(x)$可化為$\sum_{i=0}^{n}\beta_i p_i(x)$，$\beta_i$為常係數，如式(2.29)所示。將式(2.29)代入式(2.19)可得：

$$I = \int_a^b w(x)f_{2n+1}(x)\,dx$$

$$= \sum_{i=0}^{n}\beta_i\int_a^b w(x)p_{n+1}(x)p_i(x)\,dx+\int_a^b w(x)h_n(x)\,dx \tag{2.30}$$

$$= \int_a^b w(x)h_n(x)\,dx \tag{2.31}$$

式(2.31)之結果利用了式(2.26)之關係（$i<(n+1)$故$i\neq(n+1)$）。將式(2.28)代入式(2.20)可得式(2.32)。注意當$p_{n+1}(x_i)=0$，即當x_i為多項式$p_{n+1}(x)$之根時，可得式(2.33)。

$$I \doteq \sum_{i=0}^{n}W_i f_{2n+1}(x_i)$$

$$= \sum_{i=0}^{n}W_i p_{n+1}(x_i)g_n(x_i)+\sum_{i=0}^{n}W_i h_n(x_i) \tag{2.32}$$

$$= \sum_{i=0}^{n}W_i h_n(x_i) \tag{2.33}$$

根據前述對式(2.20)之論証，可知式(2.33)之值會正好等於式(2.31)之積分，因此可得

$$I=\int_a^b w(x)f_{2n+1}(x)\,dx=\sum_{i=0}^{n}W_i f_{2n+1}(x_i) \tag{2.34}$$

亦即僅用$n+1$個特殊點x_i之函數值，即可算得$2n+1$次多項式$f_{2n+1}(x)$之正確積分值，如式(2.34)所示。這種數值積分方法稱為高斯積分法。注意$n+1$次正交函數$p_{n+1}(x)$即為式(2.23)中之函數$\pi(x)$。

2.7 常用正交函數

[表六]列出一些常用之正交函數之多項式，積分區間與權函數。

[表六] 正交函數

polynomial	$[a,b]$	$w(x)$
Legendre	$[-1,1]$	1
Jacobi	$[0,1]$	x^k
Legendre	$[a,b]$	$\sqrt{b-x}$
Legendre	$[a,b]$	$\frac{1}{\sqrt{b-x}}$
Chebyshev	$[-1,1]$	$\frac{1}{\sqrt{1-x^2}}$
Chebyshev	$[a,b]$	$\frac{1}{\sqrt{x-a}\sqrt{x-b}}$
Laguerre	$[0,\infty]$	e^{-x}
Hermite	$[-\infty,\infty]$	e^{-x^2}
Special	$[0,1]$	$\ln(x)$

[表七]僅列出較常用之*Legendre*多項式$P_n(x)$，*Chebyshev*多項式$T_n(x)$，*Laguerre*多項式$L_n(x)$，*Hermite*多項式$H_n(x)$。各多項式之零次函數均為1，即$P_o(x)=T_o(x)=L_o(x)=H_o(x)=1$，未列於表中。一般正交函數均有對應之遞迴關係存在，因此亦列在各該多項式之後。

[表七] 常用正交函數

$$P_1(x)=x$$
$$P_2(x)=(3x^2-1)/2$$
$$P_3(x)=(5x^3-3x)/2$$
$$P_4(x)=(35x^4-30x^2+3)/8$$
$$P_n(x)=\frac{2n-1}{n}xP_{n-1}(x)-\frac{n-1}{n}P_{n-2}(x)$$

$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$
$$T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1$$
$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$$

$$L_1(x) = -x + 1$$
$$L_2(x) = x^2 - 4x + 2$$
$$L_3(x) = -x^3 + 9x^2 - 18x + 6$$
$$L_4(x) = x^4 - 16x^3 + 72x^2 - 96x + 24$$
$$L_n(x) = (2n - 1 - x)L_{n-1}(x) - (n-1)^2 L_{n-2}(x)$$

$$H_1(x) = 2x$$
$$H_2(x) = 4x^2 - 2$$
$$H_3(x) = 8x^3 - 12x$$
$$H_4(x) = 16x^4 - 48x^2 + 12$$
$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x)$$

［表八］ 正交函數之根及權值

polynomial	$n+1$	x_i	W_i
Legendre	2	$\pm\sqrt{1/3}$	1
	3	$0, \pm\sqrt{3/5}$	$8/9, 5/9$
	4	±0.33998 10435 84856	0.65214 51548 62546
		±0.86113 63115 94053	0.34785 48451 37454
	5	0	0.56888 88888 88889
		±0.53846 93101 05683	0.47862 86704 99366
		±0.90617 98459 38664	0.23692 68850 56189
	6	±0.23861 91860 83197	0.46791 39345 72691
		±0.66120 93864 66265	0.36076 15730 48139
		±0.93246 95142 03152	0.17132 44923 79170
Chebyshev	$n+1$	$\cos\frac{(2i+1)\pi}{2n+2}$	$\frac{\pi}{n+1}$

［表八］ 正交函數之根及權值（續）

polynomial	$n+1$	x_i	W_i
Laguerre	2	$2-\sqrt{2}$	0.85355 33905 93
		$2+\sqrt{2}$	0.14644 66094 07
	3	0.41577 45567 83	0.71109 30099 29
		2.29428 03602 79	0.27851 77335 69
		6.28994 50829 37	0.10389 25650 16$E-1$
	4	0.32254 76896 19	0.60315 41043 42
		1.74576 11011 58	0.35741 86924 38
		4.53662 02969 21	0.38887 90851 50$E-1$
		9.39507 09123 01	0.53929 47055 61$E-3$
	5	0.26356 03197 18	0.52175 56105 83
		1.41340 30591 07	0.39866 68110 83
		3.59642 57710 41	0.75942 44968 17$E-1$
		7.08581 00058 59	0.36117 58679 92$E-2$
		12.64080 08442 76	0.23369 97238 58$E-4$
	6	0.22284 66041 79	0.45896 46739 50
		1.18893 21016 73	0.41700 08307 72
		2.99273 63260 59	0.11337 33820 74
		5.77514 35691 05	0.10399 19745 31$E-1$
		9.83746 74183 83	0.26101 72028 15$E-3$
		15.98287 39806 02	0.89854 79064 30$E-6$
Hermite	2	$\pm\sqrt{1/2}$	0.88622 69254 528
	3	0	1.18163 59006 04
		$\pm\sqrt{3/2}$	0.29540 89751 509
	4	±0.52464 76232 75290	0.80491 40900 055
		±1.65068 01238 85785	0.08131 28354 4725
	5	0	0.94530 87204 829
		±0.95857 24646 13819	0.39361 93231 522
		±2.02018 28704 56086	0.01995 32420 5905

對於各種n值之$(x_0, x_1, \ldots, x_n)$與$(W_0, W_1, \ldots, W_n)$，一般可由如參考

文獻[1,2]等數學函數手冊查得。[表八]亦僅列示較常用多項式之部分x_i與W_i值以供參考。

利用高斯積分法時必須有相同的區間及權函數。(1)要使區間相同，一般可做適當之變數轉換，如以下範例之式(2.36)與式(2.39)。(2)要使權函數相同，一般可將原積分函數除以權函數做為式(2.19)之$f(x)$，如以下範例之式(2.40)。

$$I = \int_a^b F(z)\,dz = (\frac{b-a}{2})\int_{-1}^{1} f(x)\,dx \tag{2.35}$$

$$z = (\frac{b-a}{2})x + (\frac{b+a}{2}), \qquad x = (\frac{2}{b-a})z - (\frac{b+a}{b-a}) \tag{2.36}$$

$$f(x) = F((\frac{b-a}{2})x + (\frac{b+a}{2})) \tag{2.37}$$

$$I = \int_a^\infty F(z)\,dz = \int_0^\infty e^{-x} f(x)\,dx \tag{2.38}$$

$$z = x + a, \qquad x = z - a \tag{2.39}$$

$$f(x) = e^x F(x+a) \tag{2.40}$$

以下之變數轉換中r值之決定以能使$g(z) = z^{-r+1}G(z)$在積分區間變化不大或接近常數為原則，如此則$f(x) = g(z)$亦將變化不大或接近常數。雖然r值之大小係取決於函數$G(z)$之特性；但s值則可任意選定。為求簡便，可取$s = 1$，則轉換後之積分式可用倫伯格積分法計算。式(2.42)與式(2.43)之變數轉換關係可由$z^{r-1}\,dz = x^{s-1}\,dx$之積分求得。

$$\int_{za}^{zb} G(z)\,dz = \int_{za}^{zb} z^{r-1} g(z)\,dz = \int_{xa}^{xb} x^{s-1} f(x)\,dx = \int_{xa}^{xb} F(x)\,dx \tag{2.41}$$

$$z = (\frac{r}{s})^{1/r} x^{s/r}, \qquad x = (\frac{s}{r})^{1/s} z^{r/s}, \qquad r \neq 0 \text{ and } s \neq 0 \tag{2.42}$$

$$z = \exp(x^s/s), \qquad x = [s\ln(z)]^{1/s}, \qquad r = 0 \text{ and } s \neq 0 \tag{2.43}$$

$$F(x) = x^{s-1} f(x) = x^{s-1} z^{1-r} G(z), \qquad f(x) = g(z) = z^{1-r} G(z) \tag{2.44}$$

[表九]為採用上述變數轉換之積分法之一範例程式，所用積分函數為式(2.64)。其中副程式$ROMR$配合函數副程式F可做式(2.42)當$1 > r > 0$且$s > r$時或式(2.43)當$s = 1$時之變數轉換之積分，但須注意積分區間應符合$zb > za > 0$之條件。其中呼叫之副程式$ROMQ$請參考[表十四]。在[表九]程式中先將無限區間$[0,\infty]$之積分經式(2.65)轉成有限區間$[0,1]$之積分，但轉換後之函數$1/2\sqrt{-\ln x}$在$x = 1$為奇異點，由下式之級數關係可知

函數在$x=1$附近之變化類似$1/\sqrt{1-x}$，因此將積分區間分為二段$[0,1/e]$與$[1/e,1]$。後段用$v=1-x$轉換變數後，再用式(2.42)以$r=1/2$及$s=1$做轉換即可用很少的積分點求得積分值。前段之積分雖然在$x=0$附近沒有奇異點，但其變化並不易看出，經由試誤方式可知當r值在0.24至0.35之間時，可用較少的積分點求得積分值，因此該範例程式即選$r=1/3$。亦請注意後段$[1/e,1]$之積分亦可直接由原函數e^{-z^2}計算區間$[0,1]$之積分。

$$\ln(1\pm x) \;=\; \pm x-\frac{x^2}{2}\pm\frac{x^3}{3}-\frac{x^4}{4}\pm\frac{x^5}{5}+\cdots \qquad (0<1\pm x\le 2)$$

［表九］ 變數轉換法之積分程式範例

```
************************************************************************
      PROGRAM ROMQMN
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/ROMQ00/ LTMAX,LSMAX,LYMIN,IDEBG,KFUNC,KSUBC,KLVLM,KDIMM
      COMMON R,S,R1,S1,RS,SR
      EXTERNAL F0,F1,G0
C**   ============================================================    **
C**   Find the integral of exp(-Y**2) from YA to YB.                   **
C**   The change of variable Y=sqrt(-ln(Z)) transforms                **
C**     the integral into (1/2)sqrt(-ln(Z))                           **
C**     from ZB=exp(-YB**2) to ZA=exp(-YA**2).                        **
C**   Then divide the interval into [ZB,1/e] and [1/e,ZA].            **
C**   For Z:[ZB,1/e] change variable to X by Z^{1/3-1}dZ=dX           **
C**   For Z:[1/e,ZA] change variable to V by Z=1-V                    **
C**   and V:[1-ZA,1-1/e] change var. to X by V^{1/2-1}dV=dX           **
C**   ============================================================    **
   10 READ (*,'(3F10.0,4I5)') YA,YB,EPS,LTMAX,LSMAX,LYMIN,IDEBG
      IF(YA.EQ.0.0)  YA= 1.0D-6
      IF(YB.EQ.0.0)  YB= 1.0D+6
      IF(EPS.EQ.0.0) EPS=1.0D-7
      IF(LTMAX.EQ.0) LTMAX=41
      IF(LSMAX.EQ.0) LSMAX=7
      IF(LYMIN.EQ.0) LYMIN=1
C**   +-------------------------------------------------------------+    **
C**   | AREA = Series expansions + Asymptotic expansion             |    **
C**   +-------------------------------------------------------------+    **
      YA2=YA**2
      YB2=YB**2
      AREA=DSQRT(3.14159265358979D0)/2-YA
     *    *(1-YA2/(3*1)+YA2**2/(5*2*1)-YA2**3/(7*3*2*1))
      IF(YB.LT.7.0D0) AREA=AREA-EXP(-YB2)/(2*YB)
     *    *(1-1/(2*YB2)+1*3/(2*YB2)**2-1*3*5/(2*YB2)**3)
      WRITE(*,'(1X,1P3E11.4,E22.14)') YA,YB,DEXP(-1.0D0),AREA
C**   +-------------------------------------------------------------+    **
C**   | Integration by the change of variable:                      |    **
C**   | Y = sqrt(-ln(Z)),  Z = exp(-Y**2)                           |    **
C**   | Z:[ZA,ZB] divided into Z:[ZA,1/e] and Z:[1/e,ZB]            |    **
C**   +-------------------------------------------------------------+    **
      ZA=DEXP(-YA**2)
      ZB=DEXP(-YB**2)
      ZC=DEXP(-1.0D0)
```

```
      S=1.0D0
C**   +--------------------------------------------------------+    **
C**   | For Z:[ZB,1/e] change variable by Z^{R-1}dZ=dX          |    **
C**   |                F0(X) = G0(Z)*Z**(1-R)                  |    **
C**   |                G0(Z) = sqrt(-ln(Z))                    |    **
C**   +--------------------------------------------------------+    **
      R=1.0D0/3.0D0
      CALL ROMR (ZB,ZC,F0,ARA0,ARB0,EPS)
      IF(IDEBG.LE.0) WRITE(*,'(2X,4I3,4I5,1PE22.14,E12.4)')
     *   LTMAX,LSMAX,LYMIN,IDEBG,KFUNC,KSUBC,KLVLM,KDIMM
     *   ,ARA0,ARB0-ARA0
C**   +--------------------------------------------------------+    **
C**   | For Z:[1/e,ZA] change variable by Z=1-V to V            |    **
C**   | V:[1-ZA,1-1/e] change variable by V^{R-1}dV=dX          |    **
C**   |                F1(X) = G1(V)*V**(1-R)                  |    **
C**   |                G1(V) = G0(1-V) = sqrt(-ln(Z))          |    **
C**   +--------------------------------------------------------+    **
      R=1.0D0/2.0D0
      CALL ROMR (1.0D0-ZA,1.0D0-ZC,F1,ARA1,ARB1,EPS)
      IF(IDEBG.LE.0) WRITE(*,'(2X,4I3,4I5,1PE22.14,E12.4)')
     *   LTMAX,LSMAX,LYMIN,IDEBG,KFUNC,KSUBC,KLVLM,KDIMM
     *   ,ARA1,ARB1-ARA1
C**   ---------------------------------------------------------    **
      WRITE(*,'(1X,1P3E11.4,E22.14,E12.4)')
     *   ZB,1-ZA,EPS,ARA0+ARA1,AREA-ARA0-ARA1
C**   +--------------------------------------------------------+    **
C**   | ARAA = value by Romberg integration tree               |    **
C**   +--------------------------------------------------------+    **
      CALL ROMQ (ZB,ZA,G0,ARAA,ARBB,EPS)
      IF(IDEBG.LE.0) WRITE(*,'(2X,4I3,4I5,1PE22.14,E12.4)')
     *   LTMAX,LSMAX,LYMIN,IDEBG,KFUNC,KSUBC,KLVLM,KDIMM
     *   ,ARAA,ARBB-ARAA
C**   ---------------------------------------------------------    **
      WRITE(*,'(1X,1P3E11.4,E22.14,2E12.4)')
     *   ZB,ZA,EPS,ARAA,ARBB-ARAA,AREA-ARAA
      GO TO 10
      END
************************************************************************
      SUBROUTINE ROMR(ZA,ZB,F,ARAA,ARBB,EPS)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON R,S,R1,S1,RS,SR
      EXTERNAL F
C**   ========================================================    **
C**      ZB          ZB              XB              XB           **
C**     /           /      R-1      /      S-1      /             **
C**    | G(Z) dZ = | g(Z) Z   dZ = | f(X) X   dX = | F(X) dX      **
C**   /           /               /               /              **
C**   ZA          ZA              XA              XA             **
C**                                        1-R                   **
C**                   f(X) = g(Z) = G(Z) Z                       **
C**                                                              **
C**                S-1        1-R S-1            1/R-1 S/R-1     **
C**   F(X) = f(X) X   = G(Z) Z   X   = G(Z) (R/S)     X          **
C**                                                              **
C**                      R-1      S-1                          **
C**                     Z   dZ = X   dX                          **
C**    --------------------------------------------------------    **
C**    IF R .NE. 0 (.AND. S .NE. 0 ) :                           **
C**                                                              **
```

```
C**                1/R  S/R                      1/S  R/S              **
C**       Z = (R/S)    X       and     X = (S/R)    Z                  **
C**     ------------------------------------------------------------    **
C**     IF R .EQ. 0 (.AND. S .NE. 0 ) :                                 **
C**                                                                     **
C**                         S                              1/S         **
C**       Z = EXP((1/S) X )     and     X = (S LOG(Z))                  **
C**     ============================================================    **
C**     This program requires the conditions (0) and ((1) or (2))       **
C**     (0) ZA > 0 and ZB > 0                                           **
C**     (1) 1 > R > 0 and S > R (S = 1 is the best)                     **
C**     (2)     R = 0 and S = 1                                         **
C**     ============================================================    **
        IF(R.NE.0.0D0) THEN
           R1=1.D0/R
           S1=1.D0/S
           RS=R*S1
           SR=S*R1
           R1=RS**R1
           S1=SR**S1
           XA=S1*ZA**RS
           XB=S1*ZB**RS
        ELSE
           XA=DLOG(ZA)
           XB=DLOG(ZB)
        ENDIF
        CALL ROMQ (XA,XB,F,ARAA,ARBB,EPS)
        RETURN
        END
*************************************************************************
        FUNCTION F0(X)
C**     ============================================================    **
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON R,S,R1,S1,RS,SR
C**     ============================================================    **
        IF(R.NE.0.0D0) THEN
           Z=R1*X**SR
           IF(S.EQ.1.0D0) THEN
              F0=G0(Z)*Z**(1-R)
           ELSE
              F0=G0(Z)*R1*SR*X**(SR-1)
           ENDIF
        ELSE
           Z=DEXP(X)
           F0=G0(Z)*Z
        ENDIF
        RETURN
        END
*************************************************************************
        FUNCTION F1(X)
C**     ============================================================    **
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON R,S,R1,S1,RS,SR
C**     ============================================================    **
        IF(R.NE.0.0D0) THEN
           Z=R1*X**SR
           IF(S.EQ.1.0D0) THEN
              F1=G1(Z)*Z**(1-R)
           ELSE
              F1=G1(Z)*R1*SR*X**(SR-1)
```

```
         ENDIF
      ELSE
         Z=DEXP(X)
         F1=G1(Z)*Z
      ENDIF
      RETURN
      END
*************************************************************************
      FUNCTION G0(Z)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
      IF(Z.LE.0.0D0) THEN
         G0=0.0D0
      ELSE IF(Z.GE.1.0D0) THEN
         G0=1.0D7
      ELSE
         G0=0.5D0/DSQRT(-DLOG(Z))
      ENDIF
      RETURN
      END
*************************************************************************
      FUNCTION G1(Z)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C**   G1=G0(1.0D0-Z)
      IF(Z.LE.0.0D0) THEN
         G1=1.0D30
      ELSE IF(Z.GE.1.0D0) THEN
         G1=0.0D0
      ELSE IF(Z.LE.1.D-3) THEN
         G1=0.5D0/DSQRT(Z+Z**2/2+Z**3/3+Z**4/4+Z**5/5)
      ELSE
         G1=0.5D0/DSQRT(-DLOG(1.0D0-Z))
      ENDIF
      RETURN
      END
*************************************************************************
```

2.8 奇異值之積分

考慮下列積分式，若$r \leq 0$，則積分值為正或負無窮大，一般視為不可積分。若$r > 0$，則積分值為有限之值，即稱該積分式為可積分。若$1 > r > 0$，則雖然積分式可積分，但積分函數在$x = a$之值為無窮大，一般稱$x = a$為函數之奇異點，若數值積分法須用到奇異點（$x = a$）之函數值時，例如梯形面積法或倫伯格積分法，則該積分法不能直接適用。

$$\int_a^b (x-a)^{r-1}\,dx = \frac{1}{r}(x-a)^r\Big|_a^b, \qquad r \neq 0 \tag{2.45}$$

$$= \ln(x-a)\Big|_a^b, \qquad r = 0 \tag{2.46}$$

以下說明幾種處理奇異點積分之方法：

(1)廣義高斯積分法：對於一些與[表六]函數相關之積分可利用對應之高斯積分法。例如式(2.41)中之$r=\frac{1}{2}$之情形可用$Chebyshev$多項式之高斯積分法。

(2)變數轉換法：對於已知r值之情形，可利用式(2.42)或式(2.43)之變數轉換，若取$s=1$，則轉換後之積分函數$f(x)$就不再有$x=0$之奇異點，因此即可採用倫伯格積分法或標準狹義高斯積分法（即$[a,b]=[-1,1],\ w(x)=1$）（可參考[表九]之副程式）。採用變數轉換法，除須知r之值外，亦須知奇異點之位置，以便先將變數平移，使奇異點在新變數為零之點，然後再利用式(2.42)或式(2.43)做變數轉換。

(3)部分解析積分法：考慮式(2.42)中$z^{r-1}g(z)$之積分，將$g(z)$用泰勒級數展開得$g(z)=g(0)+g'(0)\,z+\frac{1}{2}g''(0)\,z^2+\cdots+\frac{1}{(p-1)!}g^{(p-1)}(0)\,z^{p-1}+\frac{1}{p!}g^{(p)}(0)\,z^p+\cdots$，若$g(0)=g'(0)=\cdots=g^{(p-1)}(0)=0$，$g^{(p)}(0)\neq 0$，則$z^{r-1}g(z)=\frac{1}{p!}g^{(p)}(0)\,z^{p+r-1}+z^{r-1}(g(z)-\frac{1}{p!}g^{(p)}(0)\,z^p)$。若$p+r>0$，則其第一項可直接由解析法計算積分值（參考式(2.45)）；第二項則不再有$z=0$之奇異點（因該項中z之最低次項為$\frac{1}{(p+1)!}g^{(p+1)}(0)\,z^{p+r}$，其值在$z=0$處為零），而可用倫伯格積分法或標準狹義高斯積分法計算。實用上，由解析法計算之第一項可包含更多高次項（第二項亦須扣除對應項）。例如當$p+r$不為整數而小於1時，最好第一項多含（第二項多扣）$\frac{1}{(p+1)!}g^{(p+1)}(0)\,z^{p+r}$，可使第二項之數值積分計算更精確或用最少的積分點。此法須求函數$g(z)$之微分為其缺點，幸好最常見的情形為$p=0$即$g(0)\neq 0$，此時即可不必求函數$g(z)$之微分。部分解析積分法也可用於函數值接近無窮大之點處之積分，如$\int_{-a}^{b}\frac{f(x)\,dx}{x^2+c^2}$中$c$值很小之情形。

(4)奇異值正負抵消法：在積分式$\int_a^b\frac{f(x)\,dx}{x}$中假設$f(0)\neq 0$。如果$a=0,\ b>0$或$a<0,\ b=0$則其積分值為正或負無窮大而為不可積分。但若$a<0,\ b>0$則$x=0$兩邊之函數值分別為正或負的無窮大值，因二者可以互相抵消，故其積分值為有限。現以$|a|>|b|$為例說明如何求此積分值：將積分區間$[a,b]$分成$[a,-b]$與$[-b,b]$二段，第一段不含奇異點，可直接做數值積分。以下說明含奇異點之第二段積分之處理方式：將區間$[-b,b]$再分成$[-b,0]$及$[0,b]$，令$x=-y$將前段積分變數改為y可得$\int_{-b}^{0}\frac{f(x)\,dx}{x}=\int_b^0\frac{-f(-y)\,dy}{-y}=-\int_0^b\frac{f(-y)\,dy}{y}$。式中$y$可改為$x$而為$-\int_0^b\frac{f(-x)\,dx}{x}$，再與後段合併即得式(2.47)。此式因$[f(0)-f(-0)]=[f(0)-f(0)]=0$，若$f'(0)$有限，則$x=0$即不再是奇異點，而可用一般數值積分法求積分值。注意式(2.47)中之分母若

為x^p時亦可做類似之處理。

$$\int_{-b}^{b} \frac{f(x)\,dx}{x} = \int_{0}^{b} \frac{[f(x)-f(-x)]\,dx}{x} \tag{2.47}$$

以下以式(2.48)之積分為例探討變數轉換的運用及好處：式(2.48)左式之積分可直接用積分區間為$[a,b]$之$Chebyshev$多項式高斯積分；或先經式(2.48)右式之變數轉換後再用積分區間為$[-1,1]$之$Chebyshev$多項式高斯積分；如果進一步再經式(2.49)及式(2.50)之變數轉換後，即可採用狹義高斯積分。請特別注意：式(2.48)左式與右式之積分只有在其積分區間為式中所示者方可利用其所對應之高斯積分。事實上，大部分廣義高斯積分之積分區間均受限於權函數而無法更改。但是經轉換為式(2.50)之積分後，其積分區間可不限於$[-1,1]$，即相當於式(2.48)中之區間不必局限於$[a,b]$或$[-1,1]$。這是因為狹義高斯積分之權函數為$w(x)=1$，沒有區間特性在內，且不受變數轉換的影響。表面上雖然其區間須為$[-1,1]$，但任何區間均可經線性（即平移及比例縮放）的變數轉換而轉為$[-1,1]$。因此當積分區間不能與廣義高斯積分之區間一致時，就不能直接使用此廣義高斯積分，而須先經變數轉換後再利用狹義高斯積分了。式(2.50)之積分除可用狹義高斯積分外，亦可利用倫伯格積分法，或其它便於取得之套裝程式。式(2.48)左式之積分也可用式(2.42)之變數轉換。該變數轉換以考慮一個奇異點之特性為主，因此將積分區間分為二部分，如式(2.51)所示。二部分之分母部分之函數顯示積分函數在奇異點附近之變化特性，分子部分之函數值（相對於分母部分之函數值）在奇異點附近則變化不大。二部分之積分變數先分別平移其原點至奇異點後，即可照式(2.42)按$r=\frac{1}{2}$之情形做變數轉換。

變數轉換能將奇異函數轉成非奇異函數之原理，可以簡單地由幾何關係說明：即新變數之間距與舊變數之間距之比例若與舊函數值之大小成正比，則新函數值之大小即可接近常數，因此也較容易做數值積分。

$$\int_{a}^{b} \frac{f(z)\,dz}{\sqrt{z-a}\sqrt{b-z}} = \int_{-1}^{1} \frac{f((\frac{b+a}{2})+(\frac{b-a}{2})x)\,dx}{\sqrt{1-x^2}} \tag{2.48}$$

$$= \int_{0}^{\pi} f((\frac{b+a}{2})-(\frac{b-a}{2})\cos\theta)\,d\theta \tag{2.49}$$

$$= \frac{\pi}{2}\int_{-1}^{1} f((\frac{b+a}{2})-(\frac{b-a}{2})\cos(\frac{\pi}{2}+\frac{\pi}{2}y))\,dy \tag{2.50}$$

$$\int_{a}^{b} \frac{f(z)\,dz}{\sqrt{z-a}\sqrt{b-z}} = \int_{a}^{\frac{b+a}{2}} \frac{[f(z)/\sqrt{b-z}]\,dz}{\sqrt{z-a}} + \int_{\frac{b+a}{2}}^{b} \frac{[f(z)/\sqrt{z-a}]\,dz}{\sqrt{b-z}} \tag{2.51}$$

2.9 無限區間之積分

積分之上限及或下限如為正或負無窮大值，則亦無法以梯形面積法或倫伯格積分法做數值積分。以下說明幾種處理無限區間積分之方法：
(1)廣義高斯積分法：對於一些與[表六]函數相關之積分可利用對應之高斯積分法。例如式(2.38)之情形可用$Laguerre$多項式之高斯積分法。如下限亦為$-\infty$之情形可用$Hermite$多項式之高斯積分法。
(2)變數轉換法：利用變數轉換法可以將積分區間轉為有限區間。但有些函數經轉換後可能會有奇異點在區間之端點，此時須要再用前節所述之方法處理。有些函數可能須要分成二段或數段分別做數值積分，所述變數轉換法可用以處理含無限區間之最後一段積分。式(2.52)至式(2.65)為一些常用之變數轉換，注意轉換函數$x(z)$在積分區間必須為單調函數 (monotonic)。式(2.60)與式(2.62)之積分下限亦可改為0。[表九]以式(2.64)為例，包含數種變數轉換以及分段積分之運用，請直接參考該程式中之說明。

$$\int_0^\infty e^{-z} f(z)\,dz = \int_0^1 f(-\ln x)\,dx \tag{2.52}$$

$$z = -\ln x, \quad x = e^{-z}, \quad dz = -\frac{1}{x}\,dx \tag{2.53}$$

$$\int_a^\infty f(z)\,dz = \int_0^1 f(\frac{a+bx}{1-x})\frac{a+b}{(1-x)^2}\,dx \tag{2.54}$$

$$z = \frac{a+bx}{1-x}, \quad x = \frac{z-a}{z+b}, \quad dz = \frac{a+b}{(1-x)^2}\,dx \tag{2.55}$$

$$\int_a^\infty f(z)\,dz = \int_0^1 f(\frac{a+b-bx}{x})\frac{a+b}{x^2}\,dx \tag{2.56}$$

$$z = \frac{a+b-bx}{x}, \quad x = \frac{a+b}{z+b}, \quad dz = -\frac{a+b}{x^2}\,dx \tag{2.57}$$

$$\int_a^\infty f(z)\,dz = \int_{-1}^1 f(a+b\frac{1+x}{1-x})\frac{2b}{(1-x)^2}\,dx \tag{2.58}$$

$$z = a+b\frac{1+x}{1-x}, \quad x = \frac{z-a-b}{z-a+b}, \quad dz = \frac{2b}{(1-x)^2}\,dx \tag{2.59}$$

$$\int_{-\infty}^\infty f(z)\,dz = \int_{-1}^1 f(\frac{1}{2}\ln\frac{1+x}{1-x})\frac{1}{(1+x)(1-x)}\,dx \tag{2.60}$$

$$z = \frac{1}{2}\ln\frac{1+x}{1-x}, \quad x = \frac{e^z-e^{-z}}{e^z+e^{-z}}, \quad dz = \frac{1}{(1+x)(1-x)}\,dx \tag{2.61}$$

$$\int_{-\infty}^{\infty} f(z)\,dz = \int_{-\pi/2}^{\pi/2} f(\tan x)\frac{1}{\cos^2 x}\,dx \tag{2.62}$$

$$z = \tan x, \quad x = \tan^{-1} z, \quad dz = \frac{1}{\cos^2 x}\,dx \tag{2.63}$$

$$\int_{0}^{\infty} e^{-z^2} f(z)\,dz = \int_{0}^{1} f(\sqrt{-\ln x})\frac{1}{2\sqrt{-\ln x}}\,dx \tag{2.64}$$

$$z = \sqrt{-\ln x}, \quad x = e^{-z^2}, \quad dz = -\frac{1}{2x\sqrt{-\ln x}}\,dx \tag{2.65}$$

(3)部分解析積分法：將積分函數$g(z)$用$z \to \infty$之近似函數取代，或扣除可以直接由解析法計算其積分值之部分函數。近似函數通常亦可直接由解析法計算其積分值或較容易選用適當的變數轉換關係。通常扣除部分函數後之函數，當$z \to \infty$時，會較原函數更快速地趨近於零，因此使積分上限可用較小的有限值替代而能求得近似積分值。例如以下為一些有解析解之積分式，可用以扣除相關之積分函數。

$$\int_{0}^{\infty} \frac{\sin bz}{z}\,dz = (b/|b|)\frac{\pi}{2} \tag{2.66}$$

$$\int_{0}^{\infty} \frac{\cos bz}{z^2+1}\,dz = \frac{\pi}{2}e^{-|b|} \tag{2.67}$$

2.10 自動布點積分法－倫伯格積分樹

本節介紹一種能夠自動調配積分點分布之積分法，即積分函數變化較大之處採用較多即較密之積分點。此法基本上以倫伯格積分法為主要依據，因此將[表一]略加改變符號後（A改為T，並省略右上標之括號）重新列示於[表十]以方便後繼之說明。以下亦將以全積分區間分成8小段為例說明。在全區間分成8小段的情況下：若將全區間對分為二個小區間，則每一小區間分別分為4小段之積分值亦可由已知之8小段之函數值用倫伯格積分表求得。但此倫伯格積分表共有二個必須另外建立。同樣地，將全區間視為4小區間，每小區間用2小段之函數值之倫伯格積分表則有4個；將全區間視為8小區間，每小區間用1小段之函數值之倫伯格積分表則有8個。現將所述之所有倫伯格積分表列出，即得[表十一]。注意[表十一]中之符號較[表十]者多了左側二個上下標。上標I與下標i表示該積分表為將全區間分成I個小區間之第i小區間之積分表。

如果將[表十一]中有小括號()之值取出另成[表十二]，有大括號{}之值取出另成[表十三]，稱其為倫伯格積分樹。[表十二]之積分樹（稱舊積分樹）上各值為將全區間分為4小段時：${}_1^1T_1^3$等於全區間之積分值；${}_1^2T_1^2,\ {}_2^2T_1^2$等於全區間分成二小區間之二積分值；${}_1^4T_1^1,\cdots,{}_4^4T_1^1$等於全區間分成四小區間之四積分值。[表十三]之積分樹（稱新積分樹）上各值為將全區間分為8小段時：${}_1^1T_1^4$等於全區間之積分值；${}_1^2T_1^3,\ {}_2^2T_1^3$等於全區間分成二小區間之二積分值；${}_1^4T_1^2,\cdots,{}_4^4T_1^2$等於全區間分成四小區間之四積分值。${}_1^8T_1^1,\cdots,{}_8^8T_1^1$等於全區間分成八小區間之八積分值。

以下說明如何由舊積分樹計算新積分樹：由[表十一]之倫伯格積分表，大括號內之值$\{{}_i^IT_1^{m+1}\}$可以由小括號內之值$({}_i^IT_1^m)$與中括號內之值$[{}_i^IT_2^m]$按式(2.15)，即下式計算：

$$\{{}_i^IT_1^{m+1}\}=\frac{4^m[{}_i^IT_2^m]-({}_i^IT_1^m)}{4^m-1} \tag{2.68}$$

其中小括號內之值$({}_i^IT_1^m)$均存於舊積分樹上；而中括號內之值$[{}_i^IT_2^m]$可按下式由新積分樹上之二個下層新值計算：

$$[{}_i^IT_2^m]=\{{}_{2i-1}^{2I}T_1^m\}+\{{}_{2i}^{2I}T_1^m\} \tag{2.69}$$

總之，新積分樹之建立先由小段加倍開始，計算舊小段中點之函數值，再配合舊小段之函數值以計算新積分樹之最下層之值。然後由式(2.69)與式(2.68)計算上一層之值，依此層層向上計算至最上一層之單一值為止。

最後說明積分樹之用途：在建立新積分樹的過程中，由新舊積分值之差可知各積分值是否符合精度要求。若積分樹上已有積分值符合精度要求（如符合精度要求之積分值不只一個，則選用區間最大者），則對應之小區間（稱A區間）之積分值即已求得，該小區間之對偶小區間（稱B區間）則成為新的獨立待求積分區間，可繼續加倍細分小段，直至積分值求得為止，再合併A與B二區間之積分值，即可得其上一層之小區間之積分值。如該上一層區間非最上層之全區間，則繼續求其對偶區間之積分值再合併成上一層積分值，直至最上層之全區間之積分值求得為止。注意在求B區間之積分時，B區間內之小段繼續細分，但A區間內之小段則不再被細分，因此可以達到自動調配積分點之分布之功能。當然在求上述B區間之積分時仍然以積分樹求各小區間之積分值，因此形成遞迴的$(recursive)$運算方式。亦請注意B區間之積分樹不必全部重新建立，其舊積分樹可直接得自原有之積分樹。

［表十］ 倫伯格積分表－8小段

$$
\begin{array}{cccc}
T_1^1 & & & \\
T_2^1 & T_1^2 & & \\
T_3^1 & T_2^2 & T_1^3 & \\
T_4^1 & T_3^2 & T_2^3 & T_1^4
\end{array}
$$

［表十一］ 大小區間之十五個倫伯格積分表

$$
\begin{array}{lcccccccc}
 & {}_1^1T_1^1 & & & & & & & \\
 & {}_1^1T_2^1 & {}_1^1T_1^2 & & & & & & \\
1I4S\rightarrow & {}_1^1T_3^1 & {}_1^1T_2^2 & ({}_1^1T_1^3) & & & & & \\
1I8S\rightarrow & {}_1^1T_4^1 & {}_1^1T_3^2 & [{}_1^1T_2^3] & \{{}_1^1T_1^4\} & & & & \\
 & {}_1^2T_1^1 & & & & {}_2^2T_1^1 & & & \\
2I2S\rightarrow & {}_1^2T_2^1 & ({}_1^2T_1^2) & & & {}_2^2T_2^1 & ({}_2^2T_1^2) & & \\
2I4S\rightarrow & {}_1^2T_3^1 & [{}_1^2T_2^2] & \{{}_1^2T_1^3\} & & {}_2^2T_3^1 & [{}_2^2T_2^2] & \{{}_2^2T_1^3\} & \\
4I1S\rightarrow & ({}_1^4T_1^1) & & ({}_2^4T_1^1) & & ({}_3^4T_1^1) & & ({}_4^4T_1^1) & \\
4I2S\rightarrow & [{}_1^4T_2^1] & \{{}_1^4T_1^2\} & [{}_2^4T_2^1] & \{{}_2^4T_1^2\} & [{}_3^4T_2^1] & \{{}_3^4T_1^2\} & [{}_4^4T_2^1] & \{{}_4^4T_1^2\} \\
8I1S\rightarrow & \{{}_1^8T_1^1\} & \{{}_2^8T_1^1\} & \{{}_3^8T_1^1\} & \{{}_4^8T_1^1\} & \{{}_5^8T_1^1\} & \{{}_6^8T_1^1\} & \{{}_7^8T_1^1\} & \{{}_8^8T_1^1\}
\end{array}
$$

［表十二］ 舊倫伯格積分樹－4小段

$$
\begin{array}{lcccccccc}
1I4S\rightarrow & & & & ({}_1^1T_1^3) & & & & \\
2I2S\rightarrow & & & ({}_1^2T_1^2) & & & & ({}_2^2T_1^2) & \\
4I1S\rightarrow & ({}_1^4T_1^1) & & ({}_2^4T_1^1) & & ({}_3^4T_1^1) & & ({}_4^4T_1^1) &
\end{array}
$$

［表十三］ 新倫伯格積分樹－8小段

$$
\begin{array}{lcccccccc}
1I8S\rightarrow & & & & \{{}_1^1T_1^4\} & & & & \\
2I4S\rightarrow & & & \{{}_1^2T_1^3\} & & & & \{{}_2^2T_1^3\} & \\
4I2S\rightarrow & & \{{}_1^4T_1^2\} & & \{{}_2^4T_1^2\} & & \{{}_3^4T_1^2\} & & \{{}_4^4T_1^2\} \\
8I1S\rightarrow & \{{}_1^8T_1^1\} & \{{}_2^8T_1^1\} & \{{}_3^8T_1^1\} & \{{}_4^8T_1^1\} & \{{}_5^8T_1^1\} & \{{}_6^8T_1^1\} & \{{}_7^8T_1^1\} & \{{}_8^8T_1^1\}
\end{array}
$$

[表十四]為上述自動布點積分法之副程式$ROMQ$。該程式考慮當細分小段數達某一預設值$2^{LSMAX-1}$，而仍無積分值符合精度要求時，則將全區間視為二個獨立區間各別求積分值（即不用更高階之積分值）。此措施可以自動分割不同函數特性之區段。該程式之呼叫方式與$ROMBRG$副程式相同，不過需要另外由$COMMON/ROMQ00/$提供其中前四個數值；副程式執行後亦可由其中後四個數值取得積分點數等資料。其使用範例可參考[表九]之程式。

[表十四] 自動布點積分法副程式ROMQ

```
************************************************************************
      SUBROUTINE ROMQ(XA,XB,F,AREA,AREB,EPS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION YA(40),YC(40),YD(40),LS(40),LV(40),LX(40),LY(40)
      COMMON/ROMQ00/ LTMAX,LSMAX,LYMIN,IDEBG,KFUNC,KSUBC,KLVLM,KDIMM
      COMMON/ROMQ01/ AA(8704),BA(8704),MA(410)
      DATA NAMAX,NMMAX,NYMAX/8704,410,40/
C**   DATA LTMAX,LSMAX,LYMIN,IDEBG/41,7,1,0/
C**   ============================================================    **
C**          / XB                                                     **
C**   AREA = |   F(X) dX  SUCH THAT  |AREA-AREB| < |AREA|*EPS        **
C**        / XA                                                       **
C**                        AREA IS MORE ACCURATE THAN AREB            **
C**   ------------------------------------------------------------    **
C*I   LTMAX = 11-41   : MAX. LEVELS OF TOTAL LAYERS (31-41)           **
C*I   LSMAX = 3-13    : MAX. LEVELS IN A LAYER       (7-9)           **
C*I   LYMIN = 0-LSMAX : MIN. LAYERS WITH FULL LEVELS  (1)             **
C*I   IDEBG = 0-3     : INDEX FOR DEBUG OUTPUT                       **
C*O   KFUNC = NUMBER OF FUNCTION CALLS                               **
C*O   KSUBC = NUMBER OF SUB-ROMT CALLS                               **
C*O   KLVLM = NUMBER OF TOTAL LEVELS    <= LTMAX                     **
C*O   KDIMM = NUMBER OF USED DIMENSIONS <= NAMAX                     **
C**        <= 2**(LSMAX-1)*(LTMAX-LSMAX+2)    = 8704  (9,41,9)       **
C**        <= 2**(LSMAX-1)*(LAYER-1)+2**LSMAX = 8192 (13,1,13)       **
C**    2304 : (LTMAX,LSMAX,LYMIN) = (41, 7,1)                        **
C**             RECOMMEND FOR EPS >= 1.E-8                           **
C**    8704 : (LTMAX,LSMAX,LYMIN) = (41, 9,1)                        **
C**             RECOMMEND FOR EPS  < 1.E-8                           **
C**    8192 : (LTMAX,LSMAX,LYMIN) = (13,13,1)                        **
C**             FOR ONE LAYER PURE ROMBERG-TABLE                     **
C**   ============================================================    **
      LAY=1
      XOA=XA
      HOR=XB-XA
      LMM=0
      LOR=1
      LOS=1
      MA(1+LMM)=0
      MA(2+LMM)=1
      AA(1)=F(XA)
      AA(2)=F(XB)
      IF(IDEBG.GT.1) WRITE(*,'(1X,2F12.10,6I3)') XOA,HOR,0,LOR,1,1,LOS,1
      KFUNC=2
```

```
      KSUBC=0
      KLVLM=1
      KDIMM=2
C**   ------------------------------------------------------------    **
C**   CALL ROMT(XOA,HOR,LOR,LOS,LMM,LAY,F,EPS)
C**   AREA=AA(1)
C**   AREB=BA(1)
C**   RETURN
C**   END
C*************************************************************************
C**   SUBROUTINE ROMT (XOA,HOR,LOR,LOS,LMM,LAY,F,EPS)
C**   COMMON/ROMQ00/ LTMAX,LSMAX,LYMIN,IDEBG,KFUNC,KSUBC,KLVLM,KDIMM
C**   COMMON/ROMQ01/ AA(8704),BA(8704),MA(410)
C**   +------------------------------------------------------------+    **
C**   | The whole interval (LVV,IXX)=(1,1) is correct ?             |    **
C**   | Y:2000 : EXIT from ROMT (no more CALL ROMT)                |    **
C**   +------------------------------------------------------------+    **
 1000 IF(LOS.GE.2) THEN
        ESS=EPS/2.**(LOR-1)
        LSS=MA(1+LMM)
        IF(DABS(AA(1+LSS)-BA(1+LSS)).LE.DABS(AA(1))*ESS) GO TO 2000
      ENDIF
C**   +------------------------------------------------------------+    **
C**   | IF any sub-interval (LVV,IXX) is correct ?                  |    **
C**   | Y:2100: CALL ROMT for other sub-interv (LVV,IYY)            |    **
C**   +------------------------------------------------------------+    **
      IF(LOS.EQ.LSMAX.OR.LAY.GT.LYMIN) THEN
        NSS=1
C**     ESS=EPS/2.**(LOR-1)
        DO 1100 LVV=2,LOS-1
          ESS=ESS/2
          NSS=NSS*2
          LSS=MA(LVV+LMM)
          DO 1100 IXX=1,NSS
          IF(DABS(AA(IXX+LSS)-BA(IXX+LSS)).LE.DABS(AA(1))*ESS) GOTO 2100
 1100   CONTINUE
      ENDIF
C**   +------------------------------------------------------------+    **
C**   | IF Max. sub-Levels (LSMAX) has been reached ?               |    **
C**   | Y: CALL ROMT for both sub-interv (LVV,1) (LVV,2)            |    **
C**   +------------------------------------------------------------+    **
      IF(LOS.EQ.LSMAX) THEN
        LVV=2
        IXX=2
        IYY=1
        GO TO 2200
      ENDIF
C**   +------------------------------------------------------------+    **
C**   | DOUBLE the sub-sections                                     |    **
C**   +------------------------------------------------------------+    **
      NSS=2**LOS
      LSS=MA(LOS+1+LMM)
      IF(LSS+NSS.GT.NAMAX.OR.LMM+LOS.GE.NMMAX) GO TO 2000
      MA(LOS+2+LMM)=LSS+NSS
      NPP=NSS/2
      LPP=MA(LOS+LMM)
      LOS=LOS+1
      H1=HOR/NSS
      H2=H1*2
      H4=H2*2
```

```
      XX=XOA+HOR-H1
      KFUNC=KFUNC+NPP
      R4P=3
C**   +--------------------------------------------------------+    **
C**   | ADD one more Lower Level of this Layer                 |    **
C**   +--------------------------------------------------------+    **
      AA(NSS+1+LSS)=AA(NPP+1+LPP)
      DO 1400 NP=NPP,1,-1
      AA(2*NP-1+LSS)=AA(NP+LPP)
      AA(2*NP  +LSS)=F(XX)
      BA(NP+LPP)=H1*(AA(2*NP-1+LSS)+AA(2*NP+1+LSS))
      AA(NP+LPP)=(BA(NP+LPP)+H4*AA(2*NP+LSS))/R4P
 1400 XX=XX-H2
C**   +--------------------------------------------------------+    **
C**   | RESET Romberg-Tree of this Layer                       |    **
C**   +--------------------------------------------------------+    **
      DO 1600 LVV=LOS-2,1,-1
      R4P=R4P*4+3
      NPP=NPP/2
      LPP=MA(LVV  +LMM)
      LSS=MA(LVV+1+LMM)
      DO 1600 NP=NPP,1,-1
      BA(NP+LPP)=AA(NP+LPP)
      T2=AA(2*NP-1+LSS)+AA(2*NP+LSS)
 1600 AA(NP+LPP)=T2+(T2-BA(NP+LPP))/R4P
C**   +--------------------------------------------------------+    **
C**   | Total Level > LTMAX ?                                  |    **
C**   | N:1000 : Add one more Level of the tree                |    **
C**   | Y:4000 : EXIT from ROMT                                |    **
C**   +--------------------------------------------------------+    **
      IF(LOR+LOS.LE.LTMAX) GO TO 1000
 2000 GO TO 4000
C**   +--------------------------------------------------------+    **
C**   | SAVE the given sub-interval number (correct one)       |    **
C**   +--------------------------------------------------------+    **
 2100 IYY=IXX
C**   +--------------------------------------------------------+    **
C**   | At top Level || DIM > NMMAX || Layer > NYMAX ?         |    **
C**   | Y:4000 : EXIT from ROMT                                |    **
C**   | N:     : Integrate the other sub-interval              |    **
C**   +--------------------------------------------------------+    **
 2200 IF(LVV.EQ.1) GO TO 4000
      IF(LMM+LOS+LOS-LVV+2.GT.NMMAX.OR.LAY.GT.NYMAX)  GO TO 4000
      LSS=MA(LVV+LMM)
      ESS=EPS/2.**(LOR+LVV-2)
      IXX=((IXX+1)/2)*4-IXX-1
C**   +--------------------------------------------------------+    **
C**   | IF the other sub-interval correct ?                    |    **
C**   | N:3000 : CALL ROMT for the other sub-interval          |    **
C**   | Y:2500 : Correct or RETURN from ROMT                   |    **
C**   +--------------------------------------------------------+    **
      IF(DABS(AA(IXX+LSS)-BA(IXX+LSS)).GT.DABS(AA(1))*ESS) GO TO 3000
C**   +--------------------------------------------------------+    **
C**   | IF both sub-intervals finish ?                         |    **
C**   | N:2200 : for 1st sub-intv do 2nd sub-intv              |    **
C**   | Y:     : Combine two sub-interval & up one Level       |    **
C**   +--------------------------------------------------------+    **
 2500 IF(IXX.EQ.IYY) GO TO 2200
      LVV=LVV-1
      IPP=(IXX+1)/2
```

```
      LPP=MA(LVV  +LMM)
      LSS=MA(LVV+1+LMM)
      BA(IPP+LPP)=BA(IXX+LSS)+BA(IYY+LSS)
      AA(IPP+LPP)=AA(IXX+LSS)+AA(IYY+LSS)
      IF(IDEBG.GT.2) WRITE(*,'(25X,4I3,6X,1P2E12.4
     *   /31X,2I3,18X,1P2E12.4/34X,I3,18X,1P2E12.4)')
     *       -LAY,LOR,LVV,IPP,AA(IPP+LPP),BA(IPP+LPP)-AA(IPP+LPP)
     *             ,LVV+1,IXX,BA(IXX+LSS)-AA(IXX+LSS),AA(IXX+LSS)
     *                   ,IYY,BA(IYY+LSS)-AA(IYY+LSS),AA(IYY+LSS)
      IXX=IPP
      GO TO 2100
C**   +------------------------------------------------------------+    **
C**   | SETUP Romberg-Tree for next Layer FROM Old one             |    **
C**   | PUSH  XOA,(HOR,LOR),LOS,(LMM,LAY),LVV,IXX,IYY              |    **
C**   |                                    of this Layer           |    **
C**   | RESET XOA, HOR,LOR ,LOS, LMM,LAY  of next Layer            |    **
C**   | CALL  ROMT (XOA,HOR,LOR,LOS,LMM,LAY,F,EPS)                 |    **
C**   +------------------------------------------------------------+    **
 3000 NSF=IXX-1
      LNV=LOS+2
      DO 3200 LOV=LVV,LOS
      MA(LNV+LMM)=MA(LOV+LMM)+NSF
      NSF=NSF*2
 3200 LNV=LNV+1
      MA(LNV+LMM)=MA(LOS+1+LMM)
      IF(IDEBG.GT.1) WRITE(*,'(1X,2F12.10,6I3)')
     *         XOA,HOR,LAY,LOR,LVV,IXX,LOS,LOS-LVV+1
      LS(LAY)=LOS
      LV(LAY)=LVV
      LX(LAY)=IXX
      LY(LAY)=IYY
      YA(LAY)=XOA
      YC(LAY)=AA(1+MA(LOS  +LMM))
      YD(LAY)=AA(1+MA(LOS+1+LMM))
      LAY=LAY+1
      HOR=HOR/2**(LVV-1)
      XOA=XOA+HOR*(IXX-1)
      LMM=LMM+LOS+1
      LOR=LOR+LVV-1
      LOS=LOS-LVV+1
      GO TO 1000
C**   +------------------------------------------------------------+    **
C**   | POP XOA,HOR,(LOR),LOS,(LMM,LAY),LVV,IXX,IYY of this Layer  |    **
C**   | EXIT from Lower Level ROMT                                 |    **
C**   +------------------------------------------------------------+    **
 4000 KSUBC=KSUBC+1
      KLVLM=MAX0(KLVLM,LOR+LOS-1)
      KDIMM=MAX0(KDIMM,MA(LOS+1+LMM))
      IF(LAY.EQ.1) GO TO 4500
      LNS=LOS
      LAY=LAY-1
      IYY=LY(LAY)
      IXX=LX(LAY)
      LVV=LV(LAY)
      LOS=LS(LAY)
      LOR=LOR-LVV+1
      LMM=LMM-LOS-1
      HOR=HOR*2**(LVV-1)
      XOA=YA(LAY)
      AA(1+MA(LOS  +LMM))=YC(LAY)
```

```
      AA(1+MA(LOS+1+LMM))=YD(LAY)
      IF(IDEBG.GT.1) WRITE(*,'(1X,2F12.10,6I3,1P2E12.4)')
     *XOA,HOR,-LAY,LOR,LVV,IXX,LOS,LNS,AA(IXX+MA(LVV+LMM))
     *            ,BA(IXX+MA(LVV+LMM))-AA(IXX+MA(LVV+LMM))
      GO TO 2500
C**   +-------------------------------------------------------+    **
C**   | EXIT from Top Level ROMT                              |    **
C**   +-------------------------------------------------------+    **
 4500 AREA=AA(1)
      AREB=BA(1)
      IF(IDEBG.GT.0) WRITE(*,'(1X,4I3,4I5,1PE22.14,E12.4)')
     *LTMAX,LSMAX,LYMIN,IDEBG,KFUNC,KSUBC,KLVLM,KDIMM,AREA,AREB-AREA
      RETURN
      END
*************************************************************************
```

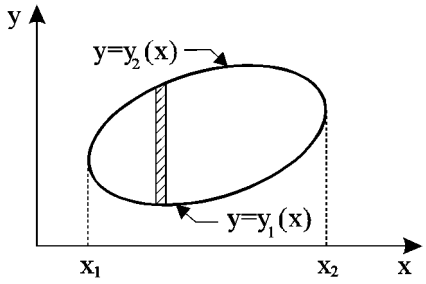
習題

1. 計算下列積分

$$A = \int_1^3 \frac{dx}{x}$$

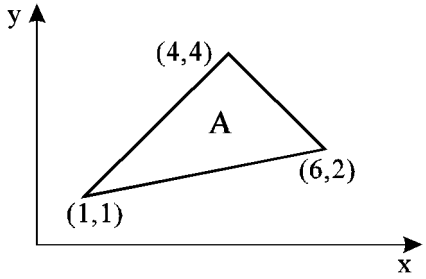
2. 試將[表二]之積分副程式$ROMBRG$修改為二個副程式，以計算雙重積分，其中一個副程式用以計算內層積分，另一副程式用以計算外層積分。外層積分函數為內層積分值，須呼叫內層積分副程式計算之。

$$V = \int_{x_1}^{x_2} [\int_{y_1(x)}^{y_2(x)} f(x,y)\,dy]\,dx$$



3. 寫三個函數程式F，$Y1$，$Y2$及一主程式以呼叫上題之雙重積分副程式計算下列積分。

$$V = \int\int_A \sin x \cos y\, dA$$



4. 試証明式(2.41)之關係。其中x與z滿足式(2.42)或式(2.43)。而$f(x)$、$g(z)$、$F(x)$與$G(z)$之間滿足式(2.44)。

參考文獻

1. *Handbook of Mathematical Functions,* Natl. Bur. Standards Appl. Math. Series, No.55, Washington, D.C., 1964.

2. *Tables of Functions and Zeros of Functions,* Natl. Bur. Standards Appl. Math. Series, No.37, Washington, D.C., 1954.

第三章

聯立線性方程式

3.1 前言

解聯立線性方程式為工程上經常會遇上之問題。聯立線性方程式之求解，理論上只要利用高斯消去法即可解決。一般聯立線性方程式之未知數之數目N均相當龐大，數萬個未知數之聯立式在有限元素分析法中可說是很平常的。求解聯立方程式之計算量與N^3成正比，聯立方程式之係數或矩陣元素之個數與N^2成正比，因此即使電腦之記憶容量再大，運算速度再快，對求解聯立方程式而言，將永遠無法滿足使用者的需求。因此如何更有效的儲存聯立方程式的係數或從事聯立方程式之運算，一直是被關心的問題。本章將探討係數矩陣能用有效方法儲存的帶狀矩陣之聯立方程式之解法，令帶狀矩陣之帶寬為M，則帶狀矩陣之元素個數與$N*M$成正比，計算量與$N*M^2$成正比，二者均遠比滿矩陣者小。當然，比帶狀矩陣更節省儲存位置且計算量更少的變寬帶矩陣之解法亦為本章探討對象。為充分利用帶狀矩陣或變寬帶矩陣之特性，聯立式之求解不能透過先求逆矩陣之方式，因逆矩陣通常不再為帶狀矩陣或變寬帶矩陣。本章之解法主要以所謂的矩陣分解法：將原係數矩陣分解為下三角矩陣與上三角矩陣之乘積，然後對聯立式右邊之常數向量用下三角矩陣做前進代入，此步驟相當於高斯消去法之消去過程，再用上三角矩陣做反向代入以求得方程式之解。上述之上三角矩陣實即高斯消去法所得之新係數矩陣，而下三角矩陣之元素亦均為高斯消去法之運算過程中所得之值。

當係數矩陣用帶狀矩陣或變寬帶矩陣儲存，則於分解過程中，若遇對角樞紐元素為零，須做二列或二行之對調時，則因其會改變原來之帶寬

而不易處理。對此問題則另於第四章提供一種有效而容易的處理方法。

3.2 聯立方程式之求解

下示一n階之聯立線性方程式

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array} \tag{3.1}$$

以矩陣符號改寫成

$$[A]\{X\} = \{B\} \tag{3.2}$$

其中

$$[A] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \qquad \{X\} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}, \qquad \{B\} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{Bmatrix}$$

式中$[A]$為$n \times n$之係數矩陣（行數與列數相等之矩陣亦稱方陣），$\{X\}$為未知向量，$\{B\}$為常數向量，均為$n \times 1$之矩陣（只有一行的矩陣亦稱行矩陣）。

上列方程式可先算出矩陣$[A]$之逆矩陣$[A]^{-1}$，再將式(3.2)兩邊前乘$[A]^{-1}$而得未知向量$\{X\}$之解為

$$\{X\} = [A]^{-1}\{B\} \tag{3.3}$$

但逆矩陣$[A]^{-1}$之計算常較費時，除非有必要，應儘量避免計算$[A]^{-1}$。一般較快速之解法為先做高斯消去法（詳細做法將述於次節）將方程式之未知數逐一從方程式中消去，最後將聯立式化成如下之型式

$$\begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & 1 & u_{23} & \cdots & u_{2n} \\ 0 & 0 & 1 & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{Bmatrix} \tag{3.4}$$

或

$$[U]\{X\} = \{Y\}$$

其中矩陣$[U]$僅對角線及其上方之元素不為零，對角線下方之其餘元素皆為零，非零元素所據範圍成一三角形故稱此矩陣為上三角矩陣。

由式(3.4)，從最後一個方程式可得$x_n = y_n$；將其代入第$n-1$個方程可得$x_{n-1} = y_{n-1} - u_{n-1,n}x_n$；再將$x_n$及$x_{n-1}$代入第$n-2$個方程式可得$x_{n-2} = y_{n-2} - u_{n-2,n-1}x_{n-1} - u_{n-2,n}x_n$；依此類推，可由第i個方程式算出$x_i$如下：

$$x_i = y_i - \sum_{j=i+1}^{n} u_{ij}x_j \tag{3.5}$$

上式須按$i = n, n-1, \cdots$至1之次序計算，因此稱為反向代入運算。

3.3 高斯消去法與矩陣分解法

以下將說明利用高斯消去法將式(3.2)化為式(3.4)之詳細過程，為了方便說明，以下面之4階聯立方程式為例分4個回合運算之。

$$\begin{bmatrix} 1 & 4 & 2 & 0 \\ 4 & 25 & 26 & 9 \\ 2 & 26 & 44 & 34 \\ 0 & 9 & 34 & 89 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 6 \\ 51 \\ 94 \\ 171 \end{Bmatrix} \tag{3.6}$$

第一回合：以第一列為樞紐列 (Pivoting row)，將第一列以後各列之第一個未知數x_1之係數a_{i1}消去，為了方便計算，先將樞紐列除以a_{11}(稱樞紐元素)使該樞紐元素化為1，欲消去其後各列中x_1之係數a_{i1}時，只要將該列係數減去a_{i1}乘樞紐列同行對應係數即得。以下為第一回合運算後之結果

$$\begin{array}{l} \overline{a}_{1j} = a_{1j}/a_{11} \\ \overline{a}_{2j} = a_{2j} - a_{21} \times \overline{a}_{1j} \\ \overline{a}_{3j} = a_{3j} - a_{31} \times \overline{a}_{1j} \\ \overline{a}_{4j} = a_{4j} - a_{41} \times \overline{a}_{1j} \end{array} \quad \begin{bmatrix} 1 & 4 & 2 & 0 \\ 0 & 9 & 18 & 9 \\ 0 & 18 & 40 & 34 \\ 0 & 9 & 34 & 89 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 6 \\ 27 \\ 82 \\ 171 \end{Bmatrix}$$

第二回合：以第二列為樞紐列，該列先除以對角線上樞紐元素$\overline{a}_{22}=9$再將樞紐列以後各列之x_2之係數$\overline{a}_{i2}$消去。以下為第二回合運算後之結果

$$\begin{array}{l}\overline{\overline{a}}_{2j}=\overline{a}_{2j}/\overline{a}_{22}\\ \overline{\overline{a}}_{3j}=\overline{a}_{3j}-\overline{a}_{32}\times\overline{\overline{a}}_{2j}\\ \overline{\overline{a}}_{4j}=\overline{a}_{4j}-\overline{a}_{42}\times\overline{\overline{a}}_{2j}\end{array}\quad\begin{bmatrix}1&4&2&0\\0&1&2&1\\0&0&4&16\\0&0&16&80\end{bmatrix}\begin{Bmatrix}x_1\\x_2\\x_3\\x_4\end{Bmatrix}=\begin{Bmatrix}6\\3\\28\\144\end{Bmatrix}$$

第三回合：以第三列為樞紐列，該列先除以對角線上樞紐元素$\overline{\overline{a}}_{33}=4$再將樞紐列以後各列之$x_3$之係數$\overline{\overline{a}}_{i3}$消去，以下為第三回合運算後之結果

$$\begin{array}{l}\overline{\overline{\overline{a}}}_{3j}=\overline{\overline{a}}_{3j}/\overline{\overline{a}}_{33}\\ \overline{\overline{\overline{a}}}_{4j}=\overline{\overline{a}}_{4j}-\overline{\overline{a}}_{43}\times\overline{\overline{\overline{a}}}_{3j}\end{array}\quad\begin{bmatrix}1&4&2&0\\0&1&2&1\\0&0&1&4\\0&0&0&16\end{bmatrix}\begin{Bmatrix}x_1\\x_2\\x_3\\x_4\end{Bmatrix}=\begin{Bmatrix}6\\3\\7\\32\end{Bmatrix}$$

第四回合：即最後一回合，只要將第四列除以$\overline{\overline{\overline{a}}}_{44}=16$即得下式而完成高斯消去法之運算。

$$\overline{\overline{\overline{\overline{a}}}}_{4j}=\overline{\overline{\overline{a}}}_{4j}/\overline{\overline{\overline{a}}}_{44}\quad\begin{bmatrix}1&4&2&0\\0&1&2&1\\0&0&1&4\\0&0&0&1\end{bmatrix}\begin{Bmatrix}x_1\\x_2\\x_3\\x_4\end{Bmatrix}=\begin{Bmatrix}6\\3\\7\\2\end{Bmatrix}$$

下式關係可以由直接計算證明

$$\begin{bmatrix}1&0&0&0\\4&9&0&0\\2&18&4&0\\0&9&16&16\end{bmatrix}\begin{bmatrix}1&4&2&0\\0&1&2&1\\0&0&1&4\\0&0&0&1\end{bmatrix}=\begin{bmatrix}1&4&2&0\\4&25&26&9\\2&26&44&34\\0&9&34&89\end{bmatrix}\tag{3.7}$$

或

$$[L][U]=[A]\tag{3.8}$$

式中之$[L]$為一下三角矩陣，其元素可分別由前述高斯消去過程中得到。如$[L]$矩陣之第一行為$[A]$矩陣第一行。$[L]$矩陣之第二行為$[A]$矩陣經第一回合消去後之$[\overline{A}]$矩陣之第二行對角線及其以下元素。$[L]$矩陣之第三行為

$[\overline{A}]$矩陣再經第二回合消去後之$[\overline{\overline{A}}]$矩陣之第三行對角線及其以下元素。餘類推。因此可證明式(3.7)之關係如下：$[\overline{A}]$係由$[A]$減$[L]$之第一行乘$[U]$之第一列而得；$[\overline{\overline{A}}]$係由$[\overline{A}]$減$[L]$之第二行乘$[U]$之第二列而得；$[\overline{\overline{\overline{A}}}]$係由$[\overline{\overline{A}}]$減$[L]$之第三行乘$[U]$之第三列而得；$[\overline{\overline{\overline{\overline{A}}}}]$減$[L]$之第四行乘$[U]$之第四列即得零矩陣；因此$[A]$等於$[L]$之第一至四行分別乘$[U]$之第一至四列之四個矩陣之和，此即等於$[L]$乘$[U]$之矩陣。注意如果將$[L]$各行以相反次序排列，即第一、四行對調，第二、三行對調；亦將$[U]$各列以相反次序排列；則二個矩陣均成右下角之三角矩陣，其乘積也會等於$[A]$。

式(3.8)中將矩陣$[A]$寫成下三角矩陣$[L]$與上三角矩陣$[U]$之乘積，稱為將$[A]$矩陣分解成$[L]$與$[U]$。注意$[L]$與$[U]$內之元素，都是由高斯消去法求得。

將$[A]$分解成$[L]$與$[U]$之乘積後，方程式之求解，可換另一種方式說明如下：由式(3.8)代入式(3.2)得

$$[L][U]\{X\} = \{B\} \tag{3.9}$$

令

$$[U]\{X\} = \{Y\} \tag{3.10}$$

則式(3.9)變成

$$[L]\{Y\} = \{B\} \tag{3.11}$$

式(3.10)即為前述高斯消去法所得之式(3.4)，其中$\{Y\}$值係由高斯消去法計算而得。但由式(3.11)之關係亦可直接計算$\{Y\}$值如下述：因$[L]$為下三角矩陣，故可由式(3.11)之第一方程式直接算得$y_1 = b_1/l_{11}$。將y_1代入第二方程式可得$y_2 = (b_2 - l_{21}y_1)/l_{22}$。將$y_1$及$y_2$代入第三方程式可得$y_3 = (b_3 - l_{31}y_1 - l_{32}y_2)/l_{33}$。餘類推，即可由第$i$方程式計算$y_i$如下：

$$y_i = (b_i - \sum_{k=1}^{i-1} l_{ik}y_k)/l_{ii} \tag{3.12}$$

上述$\{Y\}$值之計算係按$y_1,\ y_2, \cdots$至y_n之次序進行，因此稱為前進代入運算。若詳細比較前進代入運算計算y_i之方式與高斯消去法計算y_i之方式，可知前進代入運算實即高斯消去法之濃縮計算。例如$y_3 = (b_3 - l_{31}y_1 - l_{32}y_2)/l_{33}$之運算，在高斯消去法中係先由第一回合算出$\overline{b}_3 = b_3 - l_{31}y_1$，次由第二回合算出$\overline{\overline{b}}_3 = \overline{b}_3 - l_{32}y_2$，再由第三回合算得$y_3 = \overline{\overline{b}}_3/l_{33}$。

上面已說明過高斯消去法可得到$[L]$與$[U]$矩陣。現在亦試著仿照計算$\{Y\}$值的前進代入運算，將$[U]$之計算濃縮寫成（以第j行為例）：

$$u_{ij} = (a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj})/l_{ii}\,, \quad i = 1,\ 2, \cdots, j-1 \tag{3.13}$$
$$u_{jj} = 1$$

上式計算相當於高斯消去法先由第一回合計算$\overline{a}_{ij} = a_{ij} - l_{i1}u_{1j}$，次由第二回合計算$\overline{\overline{a}}_{ij} = \overline{a}_{ij} - l_{i2}u_{2j}$，依此類推至第$i$回合時除以$l_{ii}$而得$u_{ij}$。

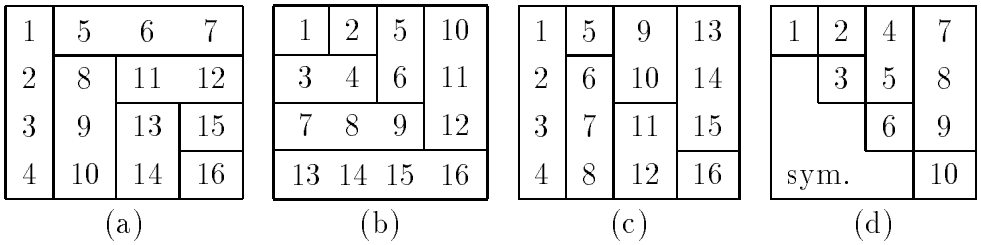
同時$[L]$之計算亦可濃縮成（以第i列為例）：

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj})\,, \qquad j = 1,\ 2, \cdots, i \tag{3.14}$$

以上各元素計算後，除$u_{jj} = 1$為固定數值可不必予以儲存外，$u_{ij}(i < j)$可放置於a_{ij}之儲存位置，l_{ij}亦可放置於$a_{ij}(i \geq j)$之儲存位置而成如下之排列

$$\begin{bmatrix} l_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & l_{22} & u_{23} & u_{24} \\ l_{31} & l_{32} & l_{33} & u_{34} \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} = \begin{bmatrix} 1 & 4 & 2 & 0 \\ 4 & 9 & 2 & 1 \\ 2 & 18 & 4 & 4 \\ 0 & 9 & 16 & 16 \end{bmatrix} \tag{3.15}$$

注意由式(3.13)與式(3.14)分別計算u_{ij}及l_{ij}時，各元素之計算次序可以有許多種安排方式，較常被採用者為如圖一中之(a)，(b)，(c)所示者，其中之數字為該元素之運算順序。計算次序的安排必須符合一個原則：計算某元素u_{ij}或l_{ij}時，必須已經算出該元素同列左邊所有之l_{ik}及該元素同行上方所有之u_{kj}。另請注意式(3.13)與式(3.14)中之$\sum$實即該元素同列左邊（除對角元素外）所有之l_{ik}所組成之向量$\{l_{i1}, l_{i2}, l_{i3}, \cdots\}$，與該元素同行上方（除對角元素外）所有之$u_{kj}$所組成之向量$\{u_{1j}, u_{2j}, u_{3j}, \cdots\}$之向量內積。以上所述之濃縮計算$[L]$與$[U]$之方法亦稱為克勞特 (Crout) 法。



(a) (b) (c) (d)

圖一 克勞特法及克雷斯基分解法

3.4 對稱矩陣之特殊處理與克雷斯基分解法

因為任一矩陣$[C]$與它的轉置矩陣$[C]^T$之乘積必為一對稱矩陣。即$[C]^T[C]=[A]$為對稱矩陣。因此若$[A]$矩陣為對稱，可考慮將式(3.8)中之$[L]$與$[U]$做適當之修改使修改後之$[\overline{L}]$與$[\overline{U}]$互為轉置矩陣。修改之法可由下述之簡單事實得知：若將$[L]$矩陣之第j行除以某數p，同時將$[U]$矩陣之第j列乘以該數p，則修改後二矩陣之乘積仍為$[A]$。因此可對$[L]$與$[U]$分別做如下之修改：以$[L]$矩陣之對角線元素l_{jj}之平方根$\sqrt{l_{jj}}$除$[L]$矩陣之第j行，同時以$\sqrt{l_{jj}}$乘$[U]$矩陣之第j列，則其乘積不變，並可發現修改後之$[\overline{L}]$與$[\overline{U}]$二矩陣互為轉置，即$[\overline{L}]=[\overline{U}]^T$或$\overline{l}_{ji}=\overline{u}_{ij}$。以式(3.7)為例即修改為

$$\begin{bmatrix}1&0&0&0\\4&3&0&0\\2&6&2&0\\0&3&8&4\end{bmatrix}\begin{bmatrix}1&4&2&0\\0&3&6&3\\0&0&2&8\\0&0&0&4\end{bmatrix}=\begin{bmatrix}1&4&2&0\\4&25&26&9\\2&26&44&34\\0&9&34&89\end{bmatrix}$$

對應之結果，亦可於消去過程中將式(3.13)與式(3.14)修改為

$$\overline{u}_{ij}=(a_{ij}-\sum_{k=1}^{i-1}\overline{u}_{ki}\overline{u}_{kj})/\overline{u}_{ii}\,,\qquad i=1,2,\cdots,j-1 \tag{3.16}$$

$$\overline{u}_{jj}=\sqrt{l_{jj}}=\sqrt{a_{jj}-\sum_{k=1}^{j-1}\overline{u}_{kj}\overline{u}_{kj}} \tag{3.17}$$

注意在式(3.16)中，$\overline{u}_{ki}=\overline{l}_{ik}=l_{ik}/\sqrt{l_{kk}}$，$\overline{u}_{kj}=u_{kj}\sqrt{l_{kk}}$，因此得$\overline{u}_{ki}\overline{u}_{kj}=l_{ik}u_{kj}$，故式(3.16)與式(3.13)等號右邊括號內之值相同。而$\overline{u}_{ii}=\overline{l}_{ii}=\sqrt{l_{ii}}$，故得$\overline{u}_{ij}=u_{ij}\sqrt{l_{ii}}$。又式(3.17)中，僅須計算在式(3.14)中$i=j$之情形。

由式(3.16)與式(3.17)計算$\overline{u}_{ij}$時，僅用到$[A]$之上三角部分元素，故可僅儲存上三角部分之元素。算得之$\overline{u}_{ij}$亦可直接置於a_{ij}之儲存位置。其計算次序可按[圖一(d)]所示者。按式(3.16)等將對稱矩陣$[A]$分解為$[\overline{L}]$與$[\overline{U}]$，使$[\overline{L}]=[\overline{U}]^T$之方法，稱為克雷斯基分解法。

克雷斯基分解法須對新的對角線元素開平方根，只有矩陣為恒正矩陣 (Positive definite matrix) 時所有新對角線元素才會都大於零。不過如果

將對稱矩陣$[A]$分解成$[A]=[U]^T[D][U]$時，即可避免負值開平方根之問題，此稱為修正克雷斯基分解法。其中$[D]$為對角線矩陣；$[U]$為上三角矩陣，其對角線元素皆為1。仍以式(3.7)為例，可修改為

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 2 & 2 & 1 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 16 \end{bmatrix} \begin{bmatrix} 1 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 2 & 0 \\ 4 & 25 & 26 & 9 \\ 2 & 26 & 44 & 34 \\ 0 & 9 & 34 & 89 \end{bmatrix}$$

事實上$[U]$矩陣即為式(3.8)中之$[U]$矩陣。$[D]$矩陣之對角線元素即為式(3.8)中$[L]$矩陣之對角線元素，即$d_{jj}=l_{jj}$，對應之運算可將式(3.13)及式(3.14)修改為

$$u_{ij}=(a_{ij}-\sum_{k=1}^{i-1}u_{ki}d_{kk}u_{kj})/d_{ii}\,,\quad i=1,\ 2,\cdots,j-1 \tag{3.18}$$

$$d_{jj}=a_{jj}-\sum_{k=1}^{j-1}u_{kj}d_{kk}u_{kj} \tag{3.19}$$

式(3.18)與式(3.19)中須計算三個數值的乘積之和，乘的運算為式(3.13)與式(3.14)之二倍。但可按下述做法避免二倍的乘算以免降低運算效率：運算次序仍照[圖一(d)]所示，但在利用式(3.18)計算u_{ij}時，暫時不除以d_{ii}，令其為$\tilde{u}_{ij}$，因此$\sum$號內之$(d_{kk}u_{kj})$即等於$\tilde{u}_{kj}$而可不必運算。然後在計算d_{jj}時，利用$\sum$計算之同時，將該行各元素$\tilde{u}_{kj}$除以其所在列之對角線元素d_{kk}，以補做式(3.18)中保留未做之除算而得u_{kj}。

3.5 帶狀矩陣之考慮

若$[A]$矩陣之非零元素僅集中在對角線及上下兩近側之平行斜線上而成如下之排列時，$[A]$矩陣稱為帶狀矩陣。

$$[A]=\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & a_{46} \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix} \tag{3.20}$$

對角線下側斜對角線數目稱為下半帶寬(ma)，對角線上側斜對角線數目稱為上半帶寬(mb)。上下半帶寬之和加1稱為全帶寬或簡稱帶寬(m)。若將$[A]$矩陣按式(3.13)及(3.14)計算而分解成$[L]$與$[U]$時，不難發現$[L]$矩陣之下半帶寬等於$[A]$矩陣之下半帶寬，$[U]$矩陣與$[A]$矩陣之上半帶寬亦相等。因此，在利用式(3.13)及式(3.14)計算時，有幾點值得注意：

1. 式(3.13)中i不必從1算起，而改為從$max(j-mb,1)$算起；計算和時，k亦不必從1算起，而改為從$max(i-ma,j-mb,1)$算起。

2. 式(3.14)中j亦改為從$max(i-ma,1)$算起；計算和時，k亦改為從$max(i-ma,j-mb,1)$算起。

至於對稱帶狀矩陣$(ma=mb=m)$，亦可做類似處理。即式(3.16)中i改為從$max(j-m,1)$算起；計算和時（含式(3.17)），k亦改為從$max(j-m,1)$算起。

3.6 變寬帶矩陣之考慮

若$[A]$矩陣之非零元素雖集中於對角線附近，但不成整齊之帶狀時，雖可劃出一條範圍視為帶狀矩陣，但所得帶狀矩陣之帶寬常常顯得過大。如果仔細觀察式(3.13)之計算，仍不難發現如式(3.21)之矩陣，經分解成$[L]$與$[U]$矩陣時，具有下列特點：$[U]$(或$[L]$)矩陣每一行(列)中由上(左)而下(右)第一個非零元素的位置與$[A]$矩陣每一行(列)中由上(左)而下(右)第一個非零元素的位置相同；而$[A]$矩陣每一行(列)中由上(左)而下(右)第一個非零元素至對角線元素間即使有零元素存在，在$[U]$(或$[L]$)矩陣之對應位置仍可能為非零元素。因此，在$[A]$矩陣中每一行(列)第一個非零元素至對角線間之元素將視為非零元素處理。每一行(列)之上(左)半帶寬則為第一個非零元素至對角線間之元素數目(包括其間零元素，但不含對角線)。因每一行(列)帶寬不等故稱其為變寬帶矩陣。

$$\begin{bmatrix} a_{11} & a_{12} & 0 & a_{14} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 \\ 0 & a_{42} & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix} =$$

$$\begin{bmatrix} l_{11} & & & & & \\ 0 & l_{22} & & & 0 & \\ 0 & 0 & l_{33} & & & \\ 0 & l_{42} & l_{43} & l_{44} & & \\ 0 & 0 & l_{53} & l_{54} & l_{55} & \\ 0 & 0 & 0 & 0 & l_{65} & l_{66} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & 0 & u_{14} & 0 & 0 \\ & 1 & u_{23} & u_{24} & 0 & 0 \\ & & 1 & u_{34} & u_{35} & 0 \\ & & & 1 & u_{45} & 0 \\ & 0 & & & 1 & u_{56} \\ & & & & & 1 \end{bmatrix} \tag{3.21}$$

$[L]$矩陣每一列中由左而右第一個非零元素的位置與$[A]$矩陣同一列中由左而右第一個非零元素的位置相同；其對應特點與$[U]$矩陣者相當。每一列之下半帶寬則為第一個非零元素至對角線間之元素數目(包括其間零元素，但不含對角線)。此時，下半帶寬亦可稱為左半帶寬。

在利用式(3.13)計算$[U]$時，i值改從$max(j-mb_j,1)$算起，k值改從$max(i-ma_i,j-mb_j,1)$算起，其中ma_i為第i列之左半帶寬，mb_j為第j行之上半帶寬。

在利用式(3.14)計算$[L]$時，j值改從$max(i-ma_i,1)$算起，k值改從$max(i-ma_i,j-mb_j,1)$算起，其中ma_i為第i列之左半帶寬，mb_j為第j行之上半帶寬。

3.7 對角線為零之處理

在前面所介紹的矩陣分解過程中，如果遇到對角線元素等於零時，則無法繼續運算下去，但矩陣本身並不一定為奇異方陣，必須在其下方同行之所有元素皆為零時，此矩陣才是奇異方陣。因此，為克服對角線上元素為零或數值太小而影響精度之問題，常於同行對角線及以下元素中找一絕對值最大者，稱部分尋樞紐 (partial pivoting)，並利用列變換（二列對調）將其調換至該行對角線位置，然後以該列為樞紐列繼續做消去或分解運算，元素運算順序則按[圖一(c)]所示者進行。

在消去或分解過程中，所有列變換必須加以記錄，以便對常數向量$\{B\}$從事相同之變換。在副程式$LUPPDC$中之$LL(N)$即用以記錄此項變換資料。如果$LL(J)=M$，即表示在第J回合消去運算之前先將第J列與第M列對調。

為進一步得到更精確之結果，於第K回合運算時可以由未曾與樞紐元素同行或同列之元素中（共有$(N+1-K)\times(N+1-K)$個元素）找一

絕對值最大者做為第K回合之樞紐元素，稱為徹底尋樞紐 (full pivoting)。如此做法須同時做列調換與行調換，將該樞紐元素調至第K列第K行位置。如第K回合消去前尋得$A(I,J)$之絕對值最大，則將第J行與第K行對調，並將第I列與第K列對調，使$A(I,J)$調至第K列第K行位置做為樞紐元素，再繼續做消去運算，並令$LL(1,K)=J$及$LL(2,K)=I$以記錄此項變換運算。

3.8 矩陣分解對列或行變換之處理

前節須在矩陣分解過程中做列調換或行調換，而使問題變得較複雜。為明瞭列或行調整對分解之影響，先回來討論簡單的列或行變換問題。

1. 所有列(或行)變換可視為由連續使用下列三種基本列(或行)變換而得

   (a) 某列(或行)乘一常數c。

   (b) 某二列(或行)對調。

   (c) 某列(或行)加入另一列(或行)乘一常數c。

2. 對一$n\times m$之矩陣$[A]$做列變換等於在矩陣$[A]$之前乘一$n\times n$之變換矩陣[R]，對應三個基本列變換之變換矩陣為：

   (a) 第i列乘以c：$[R]=[S_i(c)]$。式中$[S_i(c)]$為由一$n\times n$之單位矩陣之第i列乘以c而得。

   (b) 第i列與第j列對調：$[R]=[E_{ij}]$。式中$[E_{ij}]$為由一$n\times n$之單位矩陣之第i列與第j列對調而得。

   (c) 第i列加入第j列乘以c：$[R]=[P_{ij}(c)]$。式中$[P_{ij}(c)]$為由一$n\times n$之單位矩陣之第i列加入第j列乘以c而得。

   例如以$n=4$為例

$$[S_2(3)]=\begin{bmatrix}1&0&0&0\\0&3&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix};\ [E_{24}]=\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix};\ [P_{24}(5)]=\begin{bmatrix}1&0&0&0\\0&1&0&5\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

3. 對一$n \times m$之矩陣$[A]$做行變換等於在矩陣$[A]$之後乘一$m \times m$之變換矩陣$[C]$，對應三個基本行變換之變換矩陣為：

   (a) 第i行乘以c：$[C] = [S_i'(c)]$。式中$[S_i'(c)]$為由一$m \times m$之單位矩陣之第i行乘以c而得。

   (b) 第i行與第j行對調：$[C] = [E_{ij}']$。式中$[E_{ij}']$為由一$m \times m$之單位矩陣之第i行與第j行對調而得。

   (c) 第i行加入第j行乘以c：$[C] = [P_{ij}'(c)]$。式中$[P_{ij}'(c)]$為由一$m \times m$之單位矩陣之第i行加入第j行乘以c而得。

   例如以$m = 4$為例

$$[S_2'(3)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \ [E_{24}'] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} ; \ [P_{24}'(5)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 5 & 0 & 1 \end{bmatrix}$$

   注意$[S_i'(c)] = S_i(c)]$與$[E_{ij}'] = [E_{ij}]$均為對稱矩陣；而$[P_{ij}'(c)] = [P_{ij}(c)]^T = [P_{ji}(c)]$則不對稱。

   列變換矩陣或行變換矩陣本身並無差別，其所以為列變換或行變換端視其係前乘於$[A]$或後乘於$[A]$而定。

4. 若矩陣$[A]$與$[B]$之乘積等於$[C]$，即

$$[A][B] = [C] \tag{3.22}$$

   上式兩邊各前乘若干基本列變換$[R_1]$, $[R_2]$, $\cdots$, $[R_n]$，得

$$[R_n] \cdots [R_2][R_1][A][B] = [R_n] \cdots [R_2][R_1][C] \tag{3.23}$$

   或

$$[\overline{A}][B] = [\overline{C}] \tag{3.24}$$

   其中

$$[\overline{A}] = [R_n] \cdots [R_2][R_1][A] \tag{3.25}$$

$$[\overline{C}] = [R_n] \cdots [R_2][R_1][C] \tag{3.26}$$

注意$[\overline{A}]$與$[\overline{C}]$係分別由$[A]$與$[C]$做了相同的列變換而得。因此，可知對矩陣$[A]$作若干“列”變換後與原矩陣$[B]$之乘積等於矩陣$[C]$作若干相同之“列”變換。

5. 同理可證知：對矩陣$[B]$作若干“行”變換後與原矩陣$[A]$之乘積等於矩陣$[C]$作若干相同之“行”變換。

6. 現在考慮下式解聯立方程式與矩陣分解的問題。

$$[A]\{X\} = \{B\} \tag{3.27}$$

上式兩邊依序前乘若干列變換矩陣$[R_1],\ [R_2], \cdots,\ [R_m]$，並令

$$\{X\} = [C_1][C_2]\cdots[C_l]\{\overline{X}\} \tag{3.28}$$

$$\{\overline{B}\} = [R_m]\cdots[R_2][R_1]\{B\} \tag{3.29}$$

$$[\overline{A}] = [R_m]\cdots[R_2][R_1][A][C_1][C_2]\cdots[C_l] \tag{3.30}$$

得

$$[\overline{A}]\{\overline{X}\} = \{\overline{B}\} \tag{3.31}$$

若對$[A]$分解，可得

$$[\overline{A}] = [\overline{L}][\overline{U}] \tag{3.32}$$

然後可由下式得$\{\overline{Y}\}$

$$[\overline{L}]\{\overline{Y}\} = \{\overline{B}\} \tag{3.33}$$

再由下式得$\{\overline{X}\}$

$$[\overline{U}]\{\overline{X}\} = \{\overline{Y}\} \tag{3.34}$$

因此，對式(3.27)之解$\{X\}$可按下式步驟：

(a) 由式(3.29)求$\{\overline{B}\}$

(b) 由式(3.33)求$\{\overline{Y}\}$

(c) 由式(3.34)求$\{\overline{X}\}$

(d) 由式(3.28)求$\{X\}$

其中諸$[R]$矩陣為分解過程中對$[A]$所做之列對調，諸$[C]$矩陣為分解過程中對$[A]$所做之行對調，$[\overline{L}]$與$[\overline{U}]$為矩陣$[A]$做過列對調及行對調後之分解矩陣。分別為下三角矩陣與上三角矩陣。

注意，對$[A]$所做之行變換係按$[C_1],\ [C_2],\cdots$至$[C_l]$之順序，但於計算$\{X\}$時，對$\{\overline{X}\}$所做之變換為列變換，且須按$[C_l],\cdots,\ [C_2]$至$[C_1]$之順序（注意與原來順序相反）。

以上所述為徹底尋樞紐時，高斯消去或分解法對列及行對調之處理方式，於部分尋樞紐時，僅須對$[A]$做列變換，為上述之特例，不另贅述。

7. 前述列對調及行對調之運算，對於帶狀矩陣或變寬帶矩陣 及對稱矩陣會造成下述困難：

(a) 兼做列調換及行調換時，$[\overline{L}]$與$[\overline{U}]$之帶寬即可能會不斷地增加。

(b) 僅做列調換時，$[\overline{U}]$之帶寬最多增至左半帶寬與上半帶寬之和，但$[\overline{L}]$之帶寬仍可能會不斷地增加。

所幸$[\overline{L}]$下三角矩陣中每行之非零元素均不超過$(ma+1)$個，仍可回存於矩陣$[A]$之下半帶寬內，但是在分解過程中須做列調換時，不要對已算得之$[\overline{L}]$矩陣之元素做列調換，且在以$[\overline{L}]$做前進代入時，須於求得y_i後一次代入有關之所有式中，即當第$i+1$列須與其後之列調換時，須俟y_i代入有關之所有式中以後才能做此項調換。其詳細過程請見副程式$CVPPDC$與$CVPPSB$。

(c) 如欲保住矩陣之對稱性，則經列調換之後必須再做相對應的行調換，例如做第2列與第4列之對調後須再做第2行與第4行之對調。因此如只做列調換，則對稱特性即不復存在，故一般若須做列調換時，常不考慮對稱之特性。

(d) 下式等號左邊之對稱矩陣，可以分解成如式所示之$[U]^T[T][U]$之型式。其中$[U]$為上三角矩陣，$[T]$為對稱之三對角矩陣但不是對角矩陣。這種分解法是為了保住對稱特性的一種變通辦法，但

有時也難免需要做二列對調及二行對調的變換，因此對於帶狀矩陣或變寬帶矩陣仍然會造成(a)項所述之困難。對此問題將於第四章提供一種有效而容易的處理方法，經由增加未知數數目的方式，可以不做列或行的對調，而分解變寬帶矩陣為$[L][D][U]$，或對稱矩陣時之$[U]^T[D][U]$。

$$\begin{bmatrix} 0 & 2 & 2 & 4 \\ 2 & 0 & 6 & 2 \\ 2 & 6 & 12 & 15 \\ 4 & 2 & 15 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.9 副程式$CUDECP$-不對稱帶狀矩陣

[表一]中副程式$CUDECP(A,N,MA,MB,MM)$為利用高斯消去法按[圖一(b)]之次序計算$[L]$及$[U]$矩陣，該二矩陣仍合併放在儲存$[A]$矩陣之位置中。其中N為$[A]$矩陣之階數，MA與MB分別為矩陣$[A]$之左半帶寬與上半帶寬，MM為$[A]$矩陣之宣告列數，暫時可視為等於N，該程式不但可做帶狀矩陣之分解，亦可做滿矩陣（即所有元素皆可不為零之矩陣）之分解，此時只要在呼叫該程式時，令$MA = MB \geq (N-1)$即可。

副程式$CUSOLX(A,B,N,MA,MB,MM)$則利用由副程式$CUDECP$算得而存於$[A]$矩陣中之$[L]$與$[U]$矩陣，對常數向量$\{B\}$先做前進代入再做反向代入而得$\{X\}$，$\{X\}$仍放在儲存$\{B\}$之位置中。

若要計算$[A]$矩陣之逆矩陣$[A]^{-1} = [A_{INV}]$，可利用下列數行指令：

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(10,10),AINV(10,10)
      ...
      CALL CUDECP(A,N,MA,MB,10)
      DO 20 J=1,N
      DO 10 I=1,N
   10 AINV(I,J)=0.0
      AINV(J,J)=1.0
   20 CALL CUSOLX(A,AINV(1,J),N,MA,MB,10)
      ...
      END
```

[表一]　不對稱帶狀矩陣或滿矩陣之分解與求解

```
************************************************************************
      SUBROUTINE CUDECP(A,N,MA,MB,MM)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      DIMENSION A(MM,1)
C**   ============================================================    **
C**   DIMENSION A(MM,N+(N-1)/MM) or A((N-1)*MM+N)                     **
C**   ------------------------------------------------------------    **
C**   Find [L] & [U] for given [A],         | a11 a12 a13  0   0  |    **
C**   Such that  [A] = [L] [U],             | a21 a22 a23 a24  0  |    **
C**                                  [A] = |  0  a32 a33 a34 a35 |    **
C**   For example:                          |  0   0  a43 a44 a45 |    **
C**   N=5, MA=1, MB=2, MM=3 (>=MA+MB):      |  0   0   0  a54 a55 |    **
C**                                                                   **
C**         | l11  0   0   0   0  |         |  1  u12 u13  0   0  |    **
C**         | l21 l22  0   0   0  |         |  0   1  u23 u24  0  |    **
C**   [L] = |  0  l32 l33  0   0  |, [U] = |  0   0   1  u34 u35 |    **
C**         |  0   0  l43 l44  0  |         |  0   0   0   1  u45 |    **
C**         |  0   0   0  l54 l55 |         |  0   0   0   0   1  |    **
C**                                                                   **
C**   INPUT  data in A(3,6) =  | a11 a12 a13\a43 a44 a45 |             **
C**                            |\a21 a22 a23 a24\a54 a55 |             **
C**                            |  ? \a32 a33 a34 a35\ ?  |             **
C**   or in 1-D array:                                                **
C**   A(17) = (a11 a21 ? a12 a22 a32 a13 a23 a33 a43 a24
C**                                      a34 a44 a54 a35 a45 a55);    **
C**                                                                   **
C**   OUTPUT data in A(3,6) =  | l11 u12 u13\l43 l44 u45 |             **
C**                            |\l21 l22 u23 u24\l54 l55 |             **
C**                            |  ? \l32 l33 u34 u35\ ?  |             **
C**   or in 1-D array:                                                **
C**   A(17) = (l11 l21 ? u12 l22 l32 u13 u23 l33 l43 u24
C**                                      u34 l44 l54 u35 u45 l55).    **
C**   ============================================================    **
      DO 50 J=1,N
      JMB=MAX0(J-MB,1)
      JMA=MAX0(J-MA,1)
C**     +-----------------------------------------------------+       **
C**     | AIJ = A(I,J) - \sum_{K=IMA}^{I-1} L(I,K)*U(K,J)     |       **
C**     | A(I,K)=L(I,K),    A(K,J)=U(K,J)                     |       **
C**     | A(I,J)=A(I,J) --> U(I,J)=AIJ/L(I,I)                 |       **
C**     +-----------------------------------------------------+       **
        DO 20 I=JMB,J-1
        IMA=MAX0(I-MA,JMB)
          AIJ=A(I,J)
          DO 10 K=IMA,I-1
          AIJ=AIJ-A(I,K)*A(K,J)
   10     CONTINUE
          A(I,J)=AIJ/A(I,I)
   20   CONTINUE
C**     +-----------------------------------------------------+       **
C**     | AJI = A(J,I) - \sum_{K=IMB}^{I-1} L(J,K)*U(K,I)     |       **
C**     | A(J,K)=L(J,K),    A(K,I)=U(K,I)                     |       **
C**     | A(J,I)=A(J,I) --> L(J,I)=AJI                        |       **
C**     +-----------------------------------------------------+       **
        DO 40 I=JMA,J
        IMB=MAX0(I-MB,JMA)
          AJI=A(J,I)
          DO 30 K=IMB,I-1
          AJI=AJI-A(J,K)*A(K,I)
   30     CONTINUE
          A(J,I)=AJI
   40   CONTINUE
C**     -----------------------------------------------------       **
```

```
        IF(A(J,J).EQ.0.0) GO TO 55
   50 CONTINUE
      RETURN
C**   +-------------------------------------------------------+    **
C**   | Return with error : L(J,J)=0                          |    **
C**   +-------------------------------------------------------+    **
   55 WRITE(*,'('' A('',I3,'','',I3,'')=0''/)') J,J
      RETURN
      END
*************************************************************************
      SUBROUTINE CUSOLX(A,B,N,MA,MB,MM)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(MM,1),B(N)
C**   ===========================================================    **
C**   DIMENSION A(MM,N+(N-1)/MM),B(N) or A((N-1)*MM+N),B(N)          **
C**   -----------------------------------------------------------    **
C**   Find {X} from [A]{X}={B} for given {B} & [A]=[L][U].           **
C**   [L] & [U] are given & put in A(*) (Get by subroutine CUDECP)   **
C**   [L][U]{X}={B} ===> [L]{Y}={B},  [U]{X}={Y}                     **
C**   ===========================================================    **
C**   +-----------------------------------------------------------+    **
C**   | Find {Y} from {B} by forward substitution of [L]{Y}={B}  |    **
C**   | BI = B(I) - \sum_{K=IMA}^{I-1} L(I,K)*Y(K)              |    **
C**   | A(I,K)=L(I,K), B(K)=Y(K), B(I)=B(I) --> Y(I)=BI/L(I,I)  |    **
C**   +-----------------------------------------------------------+    **
      B(1)=B(1)/A(1,1)
      DO 70 I=2,N
      IMA=MAX0(I-MA,1)
        BI=B(I)
        DO 60 K=IMA,I-1
        BI=BI-A(I,K)*B(K)
   60   CONTINUE
        B(I)=BI/A(I,I)
   70 CONTINUE
C**   +-----------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | X(I) = BI = Y(I) - \sum_{K=I+1}^{IMB} U(I,K)*X(K)       |    **
C**   | A(I,K)=U(I,K), B(K)=X(K), B(I)=Y(I) --> X(I)=BI         |    **
C**   +-----------------------------------------------------------+    **
      DO 90 I=N-1,1,-1
      IMB=MIN0(I+MB,N)
        BI=B(I)
        DO 80 K=I+1,IMB
        BI=BI-A(I,K)*B(K)
   80   CONTINUE
        B(I)=BI
   90 CONTINUE
      RETURN
      END
*************************************************************************
```

[表二]示一主程式用以讀入矩陣階數N，左半帶寬MA及上半帶寬MB。然後讀入係數矩陣$AGEN(N,N)$及常數向量$BG(N)$。其後為輸入資料及輸出結果，其中副程式$PRINTA$可用以印出各種大小之矩陣。

[表二(a)] 不對稱帶狀矩陣之分解與求解主程式

```
************************************************************************
C**   PROGRAM FOR DECOMPOSITION OF AN UNSYMMETRIC BANDED MATRIX       **
C**   AND SOLVE THE EQUATION BY SUBSTITUTION                          **
C**   ================================================================**
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AGEN(8,8),BG(8)
      MAG=8
C**   ================================================================**
      READ(*,'(3I4)') N,MA,MB
      DO 20 I=1,N
   20 READ(*,'(10F8.0)') (AGEN(I,J),J=1,N)
      READ(*,'(10F8.0)') (BG(J),J=1,N)
C**   ----------------------------------------------------------------**
      WRITE(*,'(/'' GIVEN MATRIX IN GENERAL FORM'')')
      CALL PRINTA(AGEN,N,N,MAG)
      WRITE(*,'(/'' GIVEN CONSTANT VECTOR''//(5X,10F6.2))')(BG(J),J=1,N)
C**   ----------------------------------------------------------------**
      CALL CUDECP(AGEN,N,MA,MB,MAG)
      CALL CUSOLX(AGEN,BG,N,MA,MB,MAG)
C**   ----------------------------------------------------------------**
      WRITE(*,'(/'' DECOMPOSED MATRIX IN GENERAL FORM'')')
      CALL PRINTA(AGEN,N,N,MAG)
      WRITE(*,'(/'' RESULT VECTOR''//(5X,10F6.2))') (BG(J),J=1,N)
      STOP
      END
************************************************************************
      SUBROUTINE PRINTA(A,N,M,NN)
C**   ================================================================**
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,M)
      DATA MM/10/
C**   ================================================================**
      DO 20 JL=1,M,MM
      JH=MIN0(JL+MM-1,M)
      WRITE(*,'(/ 5X,10I6)') (J,J=JL,JH)
      DO 20 I=1,N
   20 WRITE(*,'(I5,10F6.2)') I,(A(I,J),J=JL,JH)
      RETURN
      END
************************************************************************
```

[表二(b)] 輸入資料

```
   8   1   2
2.0     1.0     2.0     0       0       0       0       0
1.0     2.0     1.3     0.6     0       0       0       0
0       2.0     3.4     2.6     0.9     0       0       0
0       0       1.5     3.3     0.93    1.2     0       0
0       0       0       0.8     1.36    0.88    0.36    0
0       0       0       0       0.3     0.62    0.34    0.1
0       0       0       0       0       0.4     2.5     1.46
0       0       0       0       0       0       1.1     3.86
1.1     0.94    4.66    6.51    0.94    0.84    -1.29   2.21
```

[表二(c)] 輸出結果

```
GIVEN MATRIX IN GENERAL FORM

        1     2     3     4     5     6     7     8
   1 2.00  1.00  2.00   .00   .00   .00   .00   .00
   2 1.00  2.00  1.30   .60   .00   .00   .00   .00
   3  .00  2.00  3.40  2.60   .90   .00   .00   .00
   4  .00   .00  1.50  3.30   .93  1.20   .00   .00
   5  .00   .00   .00   .80  1.36   .88   .36   .00
   6  .00   .00   .00   .00   .30   .62   .34   .10
   7  .00   .00   .00   .00   .00   .40  2.50  1.46
   8  .00   .00   .00   .00   .00   .00  1.10  3.86

GIVEN CONSTANT VECTOR

      1.10   .94  4.66  6.51   .94   .84 -1.29  2.21

DECOMPOSED MATRIX IN GENERAL FORM

        1     2     3     4     5     6     7     8
   1 2.00   .50  1.00   .00   .00   .00   .00   .00
   2 1.00  1.50   .20   .40   .00   .00   .00   .00
   3  .00  2.00  3.00   .60   .30   .00   .00   .00
   4  .00   .00  1.50  2.40   .20   .50   .00   .00
   5  .00   .00   .00   .80  1.20   .40   .30   .00
   6  .00   .00   .00   .00   .30   .50   .50   .20
   7  .00   .00   .00   .00   .00   .40  2.30   .60
   8  .00   .00   .00   .00   .00   .00  1.10  3.20

RESULT VECTOR

      -.50  -.30  1.20   .80 -1.00  2.50 -1.50  1.00
```

3.10 副程式$CBDECP$-對稱帶狀矩陣

副程式$CBDECP(A,N,M,MM)$相當於$CUDECP$，但$[A]$矩陣須為對稱矩陣。一為利用克雷斯基法另一為利用修正克雷斯基法，均按[圖一(d)]之次序計算$[U]$或$[D]$及$[U]$矩陣，該等矩陣仍合併放在儲存$[A]$矩陣之位置。$[A]$矩陣只需有對角線及以上之元素值即可。N為$[A]$之階數，M為半帶寬，MM為[A]之宣告列數，暫時可視為等於N。

副程式$CBSOLX(A,B,N,M,MM)$則利用由副程式$CBDECP$算得而存於$[A]$矩陣中之$[U]$或$[D]$及$[U]$，對常數向量$\{B\}$做前進及反向代入而得$\{X\}$，仍放於儲存$\{B\}$之位置中。

副程式亦同樣可做對稱滿矩陣之分解，令$M \geq (N-1)$即可。逆矩陣$[A]^{-1}$之計算指令亦與非對稱矩陣者相似，於此從略。程式之呼用法將於第五章介紹帶狀矩陣及變寬帶矩陣儲存法之後舉例說明之。

[表三(a)] 對稱帶狀矩陣之分解與求解-克雷斯基法

```
************************************************************************
      SUBROUTINE CBDECP(A,M,N,MM)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1)
C**   ============================================================    **
C**   DIMENSION A(MM,N+(N-1)/MM) or A((N-1)*MM+N)                     **
C**   ------------------------------------------------------------    **
C**   Find [U] for given [A],              | a11 a12 a13  0   0  |     **
C**   Such that  [A] = [U]'[U],            | a12 a22 a23 a24  0  |     **
C**   where [U]'= transpose of [U].  [A] = | a13 a23 a33 a34 a35 |     **
C**                                        |  0  a24 a34 a44 a45 |     **
C**   For example: N=5, M=2, MM=2:         |  0   0  a35 a45 a55 |     **
C**                                                                    **
C**         | u11  0   0   0   0  |        | u11 u12 u13  0   0  |     **
C**         | u12 u22  0   0   0  |        |  0  u22 u23 u24  0  |     **
C**   [U]'= | u13 u23 u33  0   0  |, [U] = |  0   0  u33 u34 u35 |     **
C**         |  0  u24 u34 u44  0  |        |  0   0   0  u44 u45 |     **
C**         |  0   0  u35 u45 u55 |        |  0   0   0   0  u55 |     **
C**                                                                    **
C**   INPUT  data in A(2,7) =  |\a11 a12 a13\a33 a34 a35\a55 |         **
C**                            |  ? \a22 a23 a24\a44 a45  ? \|         **
C**   or in 1-D array:                                                 **
C**   A(13) = (a11 ? a12 a22 a13 a23 a33 a24 a34 a44 a35 a45 a55);     **
C**                                                                    **
C**   OUTPUT data in A(2,7) =  |\u11 u12 u13\u33 u34 u35\u55 |         **
C**                            |  ? \u22 u23 u24\u44 u45  ? \|         **
C**   or in 1-D array:                                                 **
C**   A(13) = (u11 ? u12 u22 u13 u23 u33 u24 u34 u44 u35 u45 u55).     **
C**   ============================================================    **
C**   +----------------------------------------------------------+     **
C**   | AIJ = A(I,J) - \sum_{K=JM}^{I-1} U(K,I)*U(K,J)           |     **
C**   | I<J : U(I,J)=AIJ/U(I,I);        I=J : U(I,J)=sqrt(AIJ)   |     **
C**   | A(K+IA)=U(K,I), A(K+JA)=U(K,J), A(I+JA)=A(I,J) --> U(I,J)|     **
C**   +----------------------------------------------------------+     **
      JA=0
      DO 40 J=1,N
      JM=MAX0(J-M,1)
        IA=(JM-1)*MM
        DO 30 I=JM,J
          AIJ=A(I+JA)
          DO 10 K=JM,I-1
          AIJ=AIJ-A(K+IA)*A(K+JA)
   10     CONTINUE
          IF(I.NE.J) THEN
            A(I+JA)=AIJ/A(I+IA)
          ELSE IF(AIJ.GT.0.0) THEN
            A(I+JA)=SQRT(AIJ)
          ELSE
            WRITE(*,'('' A('',I3,'','',I3,'') <= 0''/)') J,J
            RETURN
          ENDIF
   30   IA=IA+MM
   40 JA=JA+MM
      RETURN
      END
************************************************************************
      SUBROUTINE CBSOLX(A,B,M,N,MM)
```

```
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(N)
C**   ================================================================    **
C**   DIMENSION A(MM,N+(N-1)/MM),B(N) or A((N-1)*MM+N),B(N)               **
C**   ----------------------------------------------------------------    **
C**   Find {X} from [A]{X}={B} for given {B} & [A]=[U]'[U].               **
C**   [U] is given & put in A(*) (Get by subroutine CBDECP)               **
C**   [U]'[U]{X}={B} ===> [U]'{Y}={B},  [U]{X}={Y}                        **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Find {Y} from {B} by forward substitution of [U]'{Y}={B}     |    **
C**   | U(I,I)*Y(I) = BI = B(I) - \sum_{K=IM}^{I-1} U'(I,K)*Y(K)     |    **
C**   | A(K+IA)=U(K,I)=U'(I,K), B(K)=Y(K), B(I)=B(I) --> Y(I)        |    **
C**   +--------------------------------------------------------------+    **
      IA=0
      DO 70 I=1,N
      IM=MAX0(I-M,1)
        BI=B(I)
        DO 60 K=IM,I-1
        BI=BI-A(K+IA)*B(K)
   60   CONTINUE
        B(I)=BI/A(I+IA)
   70 IA=IA+MM
C**   +--------------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y}    |    **
C**   | For I=N:1  Compute  B(K)=B(K)-U(K,I)*X(I) for K=IM:(I-1)     |    **
C**   | A(K+IA)=U(K,I), B(K)=Y(K) --> U(K,K)*X(K) ==> BI=X(I)        |    **
C**   +--------------------------------------------------------------+    **
      DO 90 I=N,1,-1
      IA=IA-MM
      IM=MAX0(I-M,1)
        BI=B(I)/A(I+IA)
        DO 80 K=IM,I-1
        B(K)=B(K)-A(K+IA)*BI
   80   CONTINUE
        B(I)=BI
   90 CONTINUE
      RETURN
      END
**********************************************************************
```

[表三(b)]　對稱帶狀矩陣之分解與求解-修正克雷斯基法

```
**********************************************************************
      SUBROUTINE CBDECP(A,M,N,MM)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1)
C**   ================================================================    **
C**   DIMENSION A(MM,N+(N-1)/MM) or A((N-1)*MM+N)                         **
C**   ----------------------------------------------------------------    **
C**   Find [D] & [U] for given [A],        | a11 a12 a13  0   0  |        **
C**   Such that  [A] = [U]'[D] [U],        | a12 a22 a23 a24  0  |        **
C**   where [U]'= transpose of [U].  [A] = | a13 a23 a33 a34 a35 |        **
C**                                        |  0  a24 a34 a44 a45 |        **
C**   For example: N=5, M=2, MM=2:         |  0   0  a35 a45 a55 |        **
C**                                                                       **
C**         | d11  0   0   0   0  |        |  1  u12 u13  0   0  |        **
```

```
C**          |  0  d22  0   0   0  |         |  0   1  u23 u24  0  |     **
C**    [D] = |  0   0  d33  0   0  |, [U] = |  0   0   1  u34 u35 |     **
C**          |  0   0   0  d44  0  |         |  0   0   0   1  u45 |     **
C**          |  0   0   0   0  d55 |         |  0   0   0   0   1  |     **
C**                                                                     **
C**    INPUT  data in A(2,7) =  |\a11 a12 a13\a33 a34 a35\a55 |         **
C**                            |  ? \a22 a23 a24\a44 a45  ? \|         **
C**    or in 1-D array:                                                 **
C**    A(13) = (a11 ? a12 a22 a13 a23 a33 a24 a34 a44 a35 a45 a55);     **
C**                                                                     **
C**    OUTPUT data in A(2,7) =  |\d11 u12 u13\d33 u34 u35\d55 |         **
C**                            |  ? \d22 u23 u24\d44 u45  ? \|         **
C**    or in 1-D array:                                                 **
C**    A(13) = (d11 ? u12 d22 u13 u23 d33 u24 u34 d44 u35 u45 d55).     **
C**    ================================================================ **
C**    +--------------------------------------------------------------+ **
C**    | D(I,I)U(I,J)=A(I,J)-\sum_{K=JM}^{I-1} D(K,K)U(K,I)U(K,J)     | **
C**    | A(K+IA)=U(K,I),     A(K+JA)=D(K,K)*U(K,J)                    | **
C**    | A(I+JA)=AIJ=A(I,J) --> D(I,I)*U(I,J)                         | **
C**    +--------------------------------------------------------------+ **
       JA=0
       DO 40 J=1,N
       JM=MAX0(J-M,1)
         IA=JM*MM
         DO 20 I=JM+1,J-1
           AIJ=A(I+JA)
           DO 10 K=JM,I-1
           AIJ=AIJ-A(K+IA)*A(K+JA)
    10     CONTINUE
           A(I+JA)=AIJ
    20   IA=IA+MM
C**    +--------------------------------------------------------------+ **
C**    | D(J,J) =A(J,J) - \sum_{K=JM}^{J-1} D(K,K)*U(K,J)*U(K,J)      | **
C**    | A(K+JA)=D(K,K)*U(K,J) --> U(K,J),  AKJ=U(K,J)                | **
C**    | A(J+JA)=AJJ=A(J,J) --> D(J,J),     A(K+KA)=D(K,K)            | **
C**    +--------------------------------------------------------------+ **
         AJJ=A(J+JA)
         KA=(JM-1)*MM
         DO 30 K=JM,J-1
           AKJ=A(K+JA)/A(K+KA)
           AJJ=AJJ-AKJ*A(K+JA)
           A(K+JA)=AKJ
    30   KA=KA+MM
         A(J+JA)=AJJ
         IF(A(J+JA).EQ.0.0) GO TO 50
    40 JA=JA+MM
       RETURN
C**    +--------------------------------------------------------------+ **
C**    | Return with error : D(J,J)=0                                 | **
C**    +--------------------------------------------------------------+ **
    50 WRITE(*,'('' A('',I3,'','',I3,'')=0''/)') J,J
       RETURN
       END
*************************************************************************
       SUBROUTINE CBSOLX(A,B,M,N,MM)
C**    ================================================================ **
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION A(1),B(N)
C**    ================================================================ **
C**    DIMENSION A(MM,N+(N-1)/MM),B(N) or A((N-1)*MM+N),B(N)            **
```

```
C**   ------------------------------------------------------------    **
C**   Find {X} from [A]{X}={B} for given {B} & [A]=[U]'[D][U].        **
C**   [D] & [U] are given & put in A(*) (Get by subroutine CBDECP)    **
C**   [U]'[D][U]{X}={B} ===> [U]'{Z}={B},  [D]{Y}={Z},  [U]{X}={Y}    **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Find {Z} from {B} by forward substitution of [U]'{Z}={B} |    **
C**   | Z(I) = BI = B(I) - \sum_{K=IM}^{I-1} U'(I,K)*Z(K)        |    **
C**   | A(K+IA)=U(K,I)=U'(I,K), B(K)=Z(K), B(I)=B(I) --> Z(I)    |    **
C**   +----------------------------------------------------------+    **
      IA=0
      DO 70 I=1,N
      IM=MAX0(I-M,1)
        BI=B(I)
        DO 60 K=IM,I-1
        BI=BI-A(K+IA)*B(K)
   60   CONTINUE
        B(I)=BI
   70 IA=IA+MM
C**   +----------------------------------------------------------+    **
C**   | Find {Y} from {Z} from the equation [D]{Y}={Z}           |    **
C**   | Y(I) = Z(I) / D(I,I)                                    |    **
C**   | B(I) = Z(I) --> Y(I),     A(I+IA)=D(I,I)                |    **
C**   +----------------------------------------------------------+    **
      IA=0
      DO 75 I=1,N
        B(I)=B(I)/A(I+IA)
   75 IA=IA+MM
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | For I=N:2  Compute  B(K)=B(K)-U(K,I)*X(I) for K=IM:(I-1) |    **
C**   | A(K+IA)=U(K,I),     B(I)=BI=X(I),     B(K)=Y(K) --> X(K) |    **
C**   +----------------------------------------------------------+    **
      DO 90 I=N,2,-1
      IA=IA-MM
      IM=MAX0(I-M,1)
        BI=B(I)
        DO 80 K=IM,I-1
        B(K)=B(K)-A(K+IA)*BI
   80   CONTINUE
   90 CONTINUE
      RETURN
      END
*************************************************************************
```

3.11 副程式$VBDECP$-對稱變寬帶矩陣

副程式$VBDECP$及$VBSOLX$與$CBDECP$及$CBSOLX$相當，同樣須$[A]$矩陣為對稱矩陣，但此時矩陣$[A]$係為變寬帶矩陣。關於變寬帶矩陣之儲存方法及帶狀矩陣之濃縮存法將於第五章介紹，此時暫不討論其詳細用法。

[表四(a)] 對稱變寬帶矩陣之分解與求解-克雷斯基法

```
************************************************************************
      SUBROUTINE VBDECP(A,LL,N)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),LL(N)
C**   ============================================================    **
C**   DIMENSION A(N+LL(N)),LL(N)                                      **
C**   ------------------------------------------------------------    **
C**   Find [U] for given [A],              | a11  0  a13  0   0  |    **
C**   Such that  [A] = [U]'[U],            |  0  a22 a23  0  a25 |    **
C**   where [U]'= transpose of [U].  [A] = | a13 a23 a33 a34 a35 |    **
C**   For example:                         |  0   0  a34 a44 a45 |    **
C**   N=5  and  LL(5)=(0 0 2 3 6):         |  0  a25 a35 a45 a55 |    **
C**                                                                   **
C**         | u11  0   0   0   0  |        | u11  0  u13  0   0  |    **
C**         |  0  u22  0   0   0  |        |  0  u22 u23  0  u25 |    **
C**   [U]'= | u13 u23 u33  0   0  |, [U] = |  0   0  u33 u34 u35 |    **
C**         |  0   0  u34 u44  0  |        |  0   0   0  u44 u45 |    **
C**         |  0  u25 u35 u45 u55 |        |  0   0   0   0  u55 |    **
C**                                                                   **
C**   Given LL(5) = (  0   0           2       3              6),     **
C**     I + LL(J) :  1+0 2+0 1+2 2+2 3+2 3+3 4+3 2+6 3+6 4+6 5+6      **
C**   Location    :    1   2   3   4   5   6   7   8   9  10  11      **
C**   INPUT A(11) = (a11 a22 a13 a23 a33 a34 a44 a25 a35 a45 a55);    **
C**   OUTPU A(11) = (u11 u22 u13 u23 u33 u34 u44 u25 u35 u45 d55).    **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | AIJ = A(I,J) - \sum_{K=IM}^{I-1} U(K,I)*U(K,J)           |    **
C**   | I<J : U(I,J)=AIJ/U(I,I);        I=J : U(I,J)=sqrt(AIJ)   |    **
C**   | A(K+IA)=U(K,I), A(K+JA)=U(K,J), A(I+JA)=A(I,J) --> U(I,J)|    **
C**   +----------------------------------------------------------+    **
      JB=0
      DO 40 J=1,N
      JA=LL(J)-LL(1)
      JM=J-JA+JB
        IB=0
        IF(JM.GT.1) IB=LL(JM-1)-LL(1)
        DO 30 I=JM,J
        IA=LL(I)-LL(1)
        IM=MAX0(I-IA+IB,JM)
          AIJ=A(I+JA)
          DO 10 K=IM,I-1
          AIJ=AIJ-A(K+IA)*A(K+JA)
   10     CONTINUE
          IF(I.NE.J) THEN
            A(I+JA)=AIJ/A(I+IA)
          ELSE IF(AIJ.GT.0.0) THEN
            A(I+JA)=SQRT(AIJ)
          ELSE
            WRITE(*,'('' A('',I3,'','',I3,'') <= 0''/)') J,J
            RETURN
          ENDIF
   30   IB=IA
   40 JB=JA
      RETURN
      END
************************************************************************
      SUBROUTINE VBSOLX(A,B,LL,N)
```

```
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(N),LL(N)
C**   ============================================================    **
C**   DIMENSION A(N+LL(N)),B(N),LL(N)                                 **
C**   ------------------------------------------------------------    **
C**   Find {X} from [A]{X}={B} for given {B} & [A]=[U]'[U].           **
C**   [U] is given & put in A(*) (Get by subroutine VBDECP)           **
C**   [U]'[U]{X}={B} ===> [U]'{Y}={B},  [U]{X}={Y}                    **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Find {Y} from {B} by forward substitution of [U]'{Y}={B} |    **
C**   | U(I,I)*Y(I) = BI = B(I) - \sum_{K=IM}^{I-1} U'(I,K)*Y(K) |    **
C**   | A(K+IA)=U(K,I)=U'(I,K), B(K)=Y(K), B(I)=B(I) --> Y(I)    |    **
C**   +----------------------------------------------------------+    **
      IB=0
      DO 70 I=1,N
      IA=LL(I)-LL(1)
      IM=I-IA+IB
        BI=B(I)
        DO 60 K=IM,I-1
        BI=BI-A(K+IA)*B(K)
   60   CONTINUE
        B(I)=BI/A(I+IA)
   70 IB=IA
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | For I=N:2  Compute  B(K)=B(K)-U(K,I)*X(I) for K=IM:(I-1) |    **
C**   | A(K+IA)=U(K,I), B(K)=Y(K) --> U(K,K)*X(K) ==> BI=X(I)    |    **
C**   +----------------------------------------------------------+    **
      DO 90 I=N,2,-1
      IA=LL(I)-LL(1)
      IM=I-LL(I)+LL(I-1)
        BI=B(I)/A(I+IA)
        DO 80 K=IM,I-1
        B(K)=B(K)-A(K+IA)*BI
   80   CONTINUE
        B(I)=BI
   90 CONTINUE
      B(1)=B(1)/A(1)
      RETURN
      END
*************************************************************************
```

[表四(b)]　對稱變寬帶矩陣之分解與求解-修正克雷斯基法

```
*************************************************************************
      SUBROUTINE VBDECP(A,LL,N)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),LL(N)
C**   ============================================================    **
C**   DIMENSION A(N+LL(N)),LL(N)                                      **
C**   ------------------------------------------------------------    **
C**   Find [D] & [U] for given [A],         | a11  0  a13  0   0  |    **
C**   Such that  [A] = [U]'[D] [U],         |  0  a22 a23  0  a25 |    **
C**   where [U]'= transpose of [U].  [A] =  | a13 a23 a33 a34 a35 |    **
C**   For example:                          |  0   0  a34 a44 a45 |    **
C**   N=5  and  LL(5)=(0 0 2 3 6):          |  0  a25 a35 a45 a55 |    **
C**                                                                   **
```

```
C**          | d11  0   0   0   0  |         |  1   0  u13  0   0  |     **
C**          |  0  d22  0   0   0  |         |  0   1  u23  0  u25 |     **
C**    [D] = |  0   0  d33  0   0  |, [U] =  |  0   0   1  u34 u35 |     **
C**          |  0   0   0  d44  0  |         |  0   0   0   1  u45 |     **
C**          |  0   0   0   0  d55 |         |  0   0   0   0   1  |     **
C**                                                                     **
C**    Given LL(5) = (  0   0           2       3               6),     **
C**      I + LL(J) :  1+0 2+0 1+2 2+2 3+2 3+3 4+3 2+6 3+6 4+6 5+6       **
C**    Location    :    1   2   3   4   5   6   7   8   9  10  11       **
C**    INPUT A(11) = (a11 a22 a13 a23 a33 a34 a44 a25 a35 a45 a55);     **
C**    OUTPU A(11) = (d11 d22 u13 u23 d33 u34 d44 u25 u35 u45 d55).     **
C**    ================================================================ **
C**    +--------------------------------------------------------------+ **
C**    | D(I,I)U(I,J)=A(I,J)-\sum_{K=IM}^{I-1} D(K,K)U(K,I)U(K,J)     | **
C**    | A(K+IA)=U(K,I),     A(K+JA)=D(K,K)*U(K,J)                    | **
C**    | A(I+JA)=AIJ=A(I,J) --> D(I,I)*U(I,J)                         | **
C**    +--------------------------------------------------------------+ **
       JB=0
       DO 40 J=1,N
       JA=LL(J)-LL(1)
       JM=J-JA+JB
         DO 20 I=JM+1,J-1
         IA=LL(I)-LL(1)
         IM=MAX0(I-LL(I)+LL(I-1),JM)
           AIJ=A(I+JA)
           DO 10 K=IM,I-1
           AIJ=AIJ-A(K+IA)*A(K+JA)
    10     CONTINUE
           A(I+JA)=AIJ
    20   CONTINUE
C**    +--------------------------------------------------------------+ **
C**    | D(J,J) =A(J,J) - \sum_{K=JM}^{J-1} D(K,K)*U(K,J)*U(K,J)      | **
C**    | A(K+JA)=D(K,K)*U(K,J) --> U(K,J),  AKJ=U(K,J)                | **
C**    | A(J+JA)=AJJ=A(J,J) --> D(J,J),     A(K+KA)=D(K,K)            | **
C**    +--------------------------------------------------------------+ **
         AJJ=A(J+JA)
         DO 30 K=JM,J-1
         KA=LL(K)-LL(1)
           AKJ=A(K+JA)/A(K+KA)
           AJJ=AJJ-AKJ*A(K+JA)
           A(K+JA)=AKJ
    30   CONTINUE
         A(J+JA)=AJJ
         IF(A(J+JA).EQ.0.0) GO TO 50
    40 JB=JA
       RETURN
C**    +--------------------------------------------------------------+ **
C**    | Return with error : D(J,J)=0                                 | **
C**    +--------------------------------------------------------------+ **
    50 WRITE(*,'('' A('',I3,'','',I3,'')=0''/)') J,J
       RETURN
       END
*************************************************************************
       SUBROUTINE VBSOLX(A,B,LL,N)
C**    ================================================================ **
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION A(1),B(N),LL(N)
C**    ================================================================ **
C**    DIMENSION A(N+LL(N)),B(N),LL(N)                                  **
C**    ---------------------------------------------------------------- **
```

```
C**   Find {X} from [A]{X}={B} for given {B} & [A]=[U]'[D][U].        **
C**   [D] & [U] are given & put in A(*) (Get by subroutine VBDECP)    **
C**   [U]'[D][U]{X}={B} ===> [U]'{Z}={B},  [D]{Y}={Z},  [U]{X}={Y}    **
C**   ================================================================    **
C**   +----------------------------------------------------------+    **
C**   | Find {Z} from {B} by forward substitution of [U]'{Z}={B} |    **
C**   | Z(I) = BI = B(I) - \sum_{K=IM}^{I-1} U'(I,K)*Z(K)        |    **
C**   | A(K+IA)=U(K,I)=U'(I,K), B(K)=Z(K), B(I)=B(I) --> Z(I)    |    **
C**   +----------------------------------------------------------+    **
      IB=0
      DO 70 I=1,N
      IA=LL(I)-LL(1)
      IM=I-IA+IB
        BI=B(I)
        DO 60 K=IM,I-1
        BI=BI-A(K+IA)*B(K)
   60   CONTINUE
        B(I)=BI
   70 IB=IA
C**   +----------------------------------------------------------+    **
C**   | Find {Y} from {Z} from the equation [D]{Y}={Z}           |    **
C**   | Y(I) = Z(I) / D(I,I)                                     |    **
C**   | B(I) = Z(I) --> Y(I),     A(I+IA)=D(I,I)                 |    **
C**   +----------------------------------------------------------+    **
      DO 75 I=1,N
      IA=LL(I)-LL(1)
        B(I)=B(I)/A(I+IA)
   75 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | For I=N:2  Compute  B(K)=B(K)-U(K,I)*X(I) for K=IM:(I-1) |    **
C**   | A(K+IA)=U(K,I),     B(I)=BI=X(I),     B(K)=Y(K) --> X(K) |    **
C**   +----------------------------------------------------------+    **
      DO 90 I=N,2,-1
      IA=LL(I)-LL(1)
      IM=I-LL(I)+LL(I-1)
        BI=B(I)
        DO 80 K=IM,I-1
        B(K)=B(K)-A(K+IA)*BI
   80   CONTINUE
   90 CONTINUE
      RETURN
      END
*************************************************************************
```

3.12 副程式$LUPPDC$-不對稱滿矩陣部分樞紐

副程式$LUPPDC$用以對一$N \times N$之矩陣$[A]$做列調換，並分解成$[\overline{L}][\overline{U}]$，且將$[\overline{L}]$與$[\overline{U}]$合併回存於$[A]$。副程式$LUPPSB$則利用由副程式$LUPPDC$算得之$[\overline{L}][\overline{U}]$矩陣對向量$\{B\}$根據列調換指標$LL(N)$做列調換及前進代入再做反向代入，得向量$\{X\}$仍回存於$\{B\}$。$LL(J)=M$表示第$J$列與第$M$列二列對調。$NN$為矩陣$[A]$之宣告列數。

［表五］ 不對稱滿矩陣之分解與求解-部分尋樞紐

```
************************************************************************
      SUBROUTINE LUPPDC(A,LL,N,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),LL(N)
C**   ============================================================    **
C**   Find [L] & [U] for given [A], Such that  [R][A] = [L][U].       **
C**   where [R] is ROW_exchanges Matrix indicated by LL(J)=M          **
C**   LL(J)=M : ROW_exchange between J_th_row & M_th_row              **
C**   ============================================================    **
      DO 50 J=1,N
C**     +----------------------------------------------------------+    **
C**     | AIJ = A(I,J) - \sum_{K=1}^{min(I,J)-1} L(I,K)*U(K,J)   |    **
C**     | A(I,K)=L(I,K),    A(K,J)=U(K,J)                        |    **
C**     +----------------------------------------------------------+    **
        AJJ=0.0
        DO 30 I=1,N
          AIJ=A(I,J)
          DO 10 K=1,MIN0(I-1,J-1)
          AIJ=AIJ-A(I,K)*A(K,J)
   10     CONTINUE
C**       +----------------------------------------------------------+    **
C**       | I <  J : A(I,J)=A(I,J) --> U(I,J)=AIJ/L(I,I)           |    **
C**       | I >= J : A(I,J)=A(I,J) --> L(I,J)=AIJ                  |    **
C**       +----------------------------------------------------------+    **
          IF(I.LT.J) THEN
            A(I,J)=AIJ/A(I,I)
          ELSE
            A(I,J)=AIJ
C**         +----------------------------------------------------+    **
C**         | Find M & AJJ=abs(L(M,J))=max(abs(L(I,J)), I=J,N)   |    **
C**         +----------------------------------------------------+    **
            IF(AJJ.LT.ABS(AIJ)) THEN
              AJJ=ABS(AIJ)
              M=I
            ENDIF
          ENDIF
   30   CONTINUE
C**     ------------------------------------------------------------    **
        IF(AJJ.EQ.0.0) GO TO 55
C**     +----------------------------------------------------------+    **
C**     | Set LL(J)=M   &   Exchange J_th_row & M_th_row           |    **
C**     +----------------------------------------------------------+    **
        LL(J)=M
        IF(M.NE.J) THEN
          DO 40 I=1,N
          AIJ=A(M,I)
          A(M,I)=A(J,I)
          A(J,I)=AIJ
   40     CONTINUE
        ENDIF
   50 CONTINUE
      RETURN
C**   +------------------------------------------------------------+    **
C**   | Return on singular matrix : L(J,J)=0                       |    **
C**   +------------------------------------------------------------+    **
   55 WRITE(*,'(/'' Matrix is singular: A('',I3,'','',I3,'')=0''/)') J,J
```

```
      RETURN
      END
*************************************************************************
      SUBROUTINE LUPPSB(A,B,LL,N,NN)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),B(N),LL(N)
C**   ================================================================    **
C**   Find {X} from [A]{X}={B} for given {B} & [R][A]=[L][U].             **
C**   [L] & [U] are given & put in A(NN,N),  [R] is given by LL(N)        **
C**   [L][U]{X}=[R]{B}  ==>  [L]{Y}=[R]{B},  [U]{X}={Y}                   **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Find {Y} from [R]{B} by forward subst. of [L]{Y}=[R]{B}  |    **
C**   | BI = B(I) - \sum_{K=1}^{I-1} L(I,K)*Y(K)                 |    **
C**   | A(I,K)=L(I,K), B(K)=Y(K), B(I)=B(I) --> Y(I)=BI/L(I,I)   |    **
C**   +--------------------------------------------------------------+    **
      DO 70 I=1,N
        M=LL(I)
        BI=B(M)
        B(M)=B(I)
        DO 60 K=1,I-1
        BI=BI-A(I,K)*B(K)
   60   CONTINUE
        B(I)=BI/A(I,I)
   70 CONTINUE
C**   +--------------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | X(I) = BI = Y(I) - \sum_{K=I+1}^{N} U(I,K)*X(K)          |    **
C**   | A(I,K)=U(I,K), B(K)=X(K), B(I)=Y(I) --> X(I)=BI          |    **
C**   +--------------------------------------------------------------+    **
      DO 90 I=N-1,1,-1
        BI=B(I)
        DO 80 K=I+1,N
        BI=BI-A(I,K)*B(K)
   80   CONTINUE
        B(I)=BI
   90 CONTINUE
      RETURN
      END
*************************************************************************
```

3.13 副程式$LUFPDC$-不對稱滿矩陣徹底樞紐

副程式$LUFPDC$用以對一$N \times N$之矩陣$[A]$做列調換與行調換（徹底尋樞紐）後，並分解成$[\overline{L}][\overline{U}]$，且將$[\overline{L}]$與$[\overline{U}]$合併回存於$[A]$。副程式$LUFPSB$則利用由副程式$LUFPDC$算得之$[\overline{L}][\overline{U}]$矩陣對向量$\{B\}$根據列調換指標$LL(2,*)$做列調換及前進代入再做反向代入，即得$\{\overline{X}\}$，最後並根據行調換指標$LL(1,*)$對$\{\overline{X}\}$做列調換而得$\{X\}$仍回存於$\{B\}$。$LL(1,K) = JK$表示第$K$行與第$JK$行二行對調。$LL(2,K) = IK$表示第$K$列與第$IK$列二列對調。因徹底尋樞紐可以獲得矩陣之奇異度$NSD$故副

程式$LUFPDC$多傳回此值。且該法可求得奇異解故提供副程式$LUFPSG$與$LUFPBS$備用。各奇異解之線性組合亦為聯立式之解。

該程式有二個版本分別列於[表六(a)]與[表六(b)]。[表六(a)]完全照式(3.27)至式(3.34)所述之方式運算。[表六(b)]則為了減少一些列對調及行對調之時間，而做了一些改變：$LUFPDC$僅將$DO\ 30\ J=1,N$與$DO\ 40\ I=1,N$改為$DO\ 30\ J=K,N$與$DO\ 40\ I=K,N$。此舉可減少對$L(*,1:(K-1))$各行之列對調及對$U(1:(K-1),*)$各列之行對調。$LUFPSB$之前進代入部分則須將已求得之$Y(K)$同時代入其後之式中，即提前在其後之$B(I)$尚未被後繼之列對調改變位置之時完成，以配合對$L(*,K)$少做之列對調。反向代入部分則須在求得$X(K)$後馬上做該值與其後之$X(JK)$之對調，以配合對$U(K,J)$少做之行對調。由於在做反向代入時已附帶對X做了列變換，因此也可省去[表六(a)]中之$DO\ 95$迴路。

上述減少部分列對調對於帶狀矩陣之分解還有一個重要的好處為儲存下三角矩陣$[L]$之元素之“儲存帶寬”與矩陣$[A]$之左半帶寬相同。注意$[L]$之實際帶寬係隨列對換而可能不斷增加。[表八]之副程式$CVPPDC$與$CVPPSB$即採用此方式而寫成。

[表六(a)]　不對稱滿矩陣之分解與求解－徹底尋樞紐

```
************************************************************************
      SUBROUTINE LUFPDC(A,LL,N,NSD,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),LL(2,N)
      DATA EPS/1.0D-7/
C**   ============================================================    **
C**   Find [L] & [U] for given [A], Such that  [R][A][C] = [L][U].    **
C**   where [C] is COL_exchanges Matrix indicated by LL(1,K)=JK       **
C**   where [R] is ROW_exchanges Matrix indicated by LL(2,K)=IK       **
C**   LL(1,K)=JK : COL_exchange between K_th_col & JK_th_col          **
C**   LL(2,K)=IK : ROW_exchange between K_th_row & IK_th_row          **
C*O   NSD = Number of singular degrees of the given matrix            **
C**   ============================================================    **
      NSD=0
      DO 80 K=1,N
C**   +-------------------------------------------------------------+    **
C**   | Find IK, JK, AKK=|A(IK,JK)|=max(|A(I,J)|, I=K,N & J=K,N) |    **
C**   | A(I,J)=A(I,J)-\sum_{k=1)^{K-1} L(I,k)*U(k,J)              |    **
C**   +-------------------------------------------------------------+    **
        AKK=0.0
        DO 20 J=K,N
        DO 10 I=K,N
          IF(AKK.GE.DABS(A(I,J))) GO TO 10
          AKK=DABS(A(I,J))
          IK=I
          JK=J
   10   CONTINUE
```

```
   20   CONTINUE
        IF(K.EQ.1) ADD=AKK
        IF(AKK.LE.ADD*EPS) GO TO 90
C**     +-------------------------------------------------------+    **
C**     | Set LL(1,K)=JK  & Exchange K_th_col & JK_th_col        |    **
C**     | Set LL(2,K)=IK  & Exchange K_th_row & IK_th_row        |    **
C**     +-------------------------------------------------------+    **
        LL(1,K)=JK
        IF(K.NE.JK) THEN
          J=JK
          DO 30 I=1,N
          AKJ=A(I,J)
          A(I,J)=A(I,K)
          A(I,K)=AKJ
   30     CONTINUE
        ENDIF
        LL(2,K)=IK
        IF(K.NE.IK) THEN
          I=IK
          DO 40 J=1,N
          AKJ=A(I,J)
          A(I,J)=A(K,J)
          A(K,J)=AKJ
   40     CONTINUE
        ENDIF
C**     +-------------------------------------------------------+    **
C**     | A(I,J)=A(I,J) - L(I,K)*U(K,J)  for I=K+1,N & J=K+1,N  |    **
C**     | A(I,K)=L(I,K),  A(K,J)=L(K,K)*U(K,J) --> U(K,J)=AKJ   |    **
C**     +-------------------------------------------------------+    **
        IF(K.EQ.N) RETURN
        DO 60 J=K+1,N
          AKJ=A(K,J)/A(K,K)
          DO 50 I=K+1,N
          A(I,J)=A(I,J)-A(I,K)*AKJ
   50     CONTINUE
          A(K,J)=AKJ
   60   CONTINUE
   80 CONTINUE
C**   +-----------------------------------------------------+    **
C**   | Return on singular matrix : L(K,K)=0                |    **
C**   +-----------------------------------------------------+    **
   90 NSD=N-K+1
      IF(ADD.EQ.0.0) ADD=1.0
      DO 95 K=N-NSD+1,N
      LL(1,K)=K
      LL(2,K)=K
      A(K,K)=DMAX1(DABS(A(K,K)),ADD*EPS*EPS)
   95 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE LUFPSB(A,B,LL,N,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),B(N),LL(2,N)
C**   ============================================================    **
C**   Find {X} from [A]{X}={B} for given {B} & [R][A][C]=[L][U].      **
C**   [C] is given by LL(1,N),   [R] is given by LL(2,N),             **
C**   [L] & [U] are given & put in A(NN,N),  [C]'=inverse of [C]      **
C**   [L][U][C]'{X}=[R]{B} => [L]{Z}=[R]{B}, [U]{Y}={Z}, {X}=[C]{Y}   **
```

```
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Find {Z} from [R]{B} by forward subst. of [L]{Z}=[R]{B}  |    **
C**   | B(I) <==> B(LL(2,I))                                     |    **
C**   | BI = B(I) - \sum_{K=1}^{I-1} L(I,K)*Z(K)                 |    **
C**   | A(I,K)=L(I,K), B(K)=Z(K), B(I)=B(I) --> Z(I)=BI/L(I,I)   |    **
C**   +----------------------------------------------------------+    **
      DO 70 I=1,N
        M=LL(2,I)
        BI=B(M)
        B(M)=B(I)
        DO 60 K=1,I-1
        BI=BI-A(I,K)*B(K)
   60   CONTINUE
        B(I)=BI/A(I,I)
   70 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find {Y} from {Z} by backward substitution of [U]{Y}={Z} |    **
C**   | Y(I) = BI = Z(I) - \sum_{K=I+1}^{N} U(I,K)*Y(K)          |    **
C**   | A(I,K)=U(I,K), B(K)=Y(K), B(I)=Z(I) --> Y(I)=BI          |    **
C**   +----------------------------------------------------------+    **
      DO 90 I=N-1,1,-1
        BI=B(I)
        DO 80 K=I+1,N
        BI=BI-A(I,K)*B(K)
   80   CONTINUE
        B(I)=BI
   90 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by ROW_exchanges : {X}=[C]{Y}          |    **
C**   +----------------------------------------------------------+    **
      DO 95 I=N-1,1,-1
        M=LL(1,I)
        BI=B(I)
        B(I)=B(M)
        B(M)=BI
   95 CONTINUE
      RETURN
      END
**********************************************************************
      SUBROUTINE LUFPBS(A,B,LL,N,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),B(N),LL(2,N)
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Find {Y} from {Z} by backward substitution of [U]{Y}={Z} |    **
C**   | Y(I) = BI = Z(I) - \sum_{K=I+1}^{N} U(I,K)*Y(K)          |    **
C**   | A(I,K)=U(I,K), B(K)=Y(K), B(I)=Z(I) --> Y(I)=BI          |    **
C**   +----------------------------------------------------------+    **
      DO 90 I=N-1,1,-1
        BI=B(I)
        DO 80 K=I+1,N
        BI=BI-A(I,K)*B(K)
   80   CONTINUE
        B(I)=BI
   90 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by ROW_exchanges : {X}=[C]{Y}          |    **
C**   +----------------------------------------------------------+    **
```

```
      DO 95 I=N-1,1,-1
        M=LL(1,I)
        BI=B(I)
        B(I)=B(M)
        B(M)=BI
   95 CONTINUE
      RETURN
      END
*****************************************************************************
      SUBROUTINE LUFPSG(A,B,LL,N,NSD,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),B(NN,NSD),LL(2,N)
C**   ============================================================    **
C**   Find the solution vectors [B] of the singular matrix [A]        **
C**   ------------------------------------------------------------    **
C*O   B(N,NSD) = Solution vectors of singular matrix A(N,N)           **
C**   ============================================================    **
      DO 50 J=1,NSD
        DO 30 I=1,N
        B(I,J)=0.0
   30   CONTINUE
        B(N-NSD+J,J)=1.0
        CALL LUFPBS(A,B(1,J),LL,N,NN)
   50 CONTINUE
      RETURN
      END
*****************************************************************************
```

[表六(b)]　不對稱滿矩陣之分解與求解-徹底尋樞紐(略省時)

```
*****************************************************************************
      SUBROUTINE LUFPDC(A,LL,N,NSD,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),LL(2,N)
      DATA EPS/1.0D-7/
C**   ============================================================    **
C**   Find [L] & [U] for given [A], Such that  [R][A][C] = [L][U].    **
C**   where [C] is COL_exchanges Matrix indicated by LL(1,K)=JK       **
C**   where [R] is ROW_exchanges Matrix indicated by LL(2,K)=IK       **
C**   LL(1,K)=JK : COL_exchange between K_th_col & JK_th_col          **
C**   LL(2,K)=IK : ROW_exchange between K_th_row & IK_th_row          **
C*O   NSD = Number of singular degrees of the given matrix            **
C**   ============================================================    **
      NSD=0
      DO 80 K=1,N
C**   +-----------------------------------------------------------+    **
C**   | Find IK, JK, AKK=|A(IK,JK)|=max(|A(I,J)|, I=K,N & J=K,N) |    **
C**   | A(I,J)=A(I,J)-\sum_{k=1)^{K-1} L(I,k)*U(k,J)              |    **
C**   +-----------------------------------------------------------+    **
        AKK=0.0
        DO 20 J=K,N
        DO 10 I=K,N
          IF(AKK.GE.DABS(A(I,J))) GO TO 10
          AKK=DABS(A(I,J))
          IK=I
          JK=J
   10   CONTINUE
   20   CONTINUE
```

```
        IF(K.EQ.1) ADD=AKK
        IF(AKK.LE.ADD*EPS) GO TO 90
C**     +-------------------------------------------------------+    **
C**     | Set LL(1,K)=JK  &  Exchange K_th_col & JK_th_col       |    **
C**     | Set LL(2,K)=IK  &  Exchange K_th_row & IK_th_row       |    **
C**     | To save time No exchange to L(I,1:(K-1)), U(1:(K-1),J) |    **
C**     +-------------------------------------------------------+    **
        LL(1,K)=JK
        IF(K.NE.JK) THEN
          J=JK
          DO 30 I=K,N
          AKJ=A(I,J)
          A(I,J)=A(I,K)
          A(I,K)=AKJ
   30     CONTINUE
        ENDIF
        LL(2,K)=IK
        IF(K.NE.IK) THEN
          I=IK
          DO 40 J=K,N
          AKJ=A(I,J)
          A(I,J)=A(K,J)
          A(K,J)=AKJ
   40     CONTINUE
        ENDIF
C**     +-------------------------------------------------------+    **
C**     | A(I,J)=A(I,J) - L(I,K)*U(K,J)  for I=K+1,N & J=K+1,N   |    **
C**     | A(I,K)=L(I,K),  A(K,J)=L(K,K)*U(K,J) --> U(K,J)=AKJ    |    **
C**     +-------------------------------------------------------+    **
        IF(K.EQ.N) RETURN
        DO 60 J=K+1,N
          AKJ=A(K,J)/A(K,K)
          DO 50 I=K+1,N
          A(I,J)=A(I,J)-A(I,K)*AKJ
   50     CONTINUE
          A(K,J)=AKJ
   60   CONTINUE
   80 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Return on singular matrix : L(K,K)=0                   |    **
C**   +-------------------------------------------------------+    **
   90 NSD=N-K+1
      IF(ADD.EQ.0.0) ADD=1.0
      DO 95 K=N-NSD+1,N
      LL(1,K)=K
      LL(2,K)=K
      A(K,K)=DMAX1(DABS(A(K,K)),ADD*EPS*EPS)
   95 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE LUFPSB(A,B,LL,N,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),B(N),LL(2,N)
C**   ============================================================    **
C**   Find {X} from [A]{X}={B} for given {B} & [R][A][C]=[L][U].      **
C**   [C] is given by LL(1,N),   [R] is given by LL(2,N),             **
C**   [L] & [U] are given & put in A(NN,N),    [C]'=inverse of [C]    **
C**   [L][U][C]'{X}=[R]{B} ===> [L]{Y}=[R]{B}, [U][C]'{X}={Y}         **
```

```
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Find {Y} from [R]{B} by forward subst. of [L]{Y}=[R]{B}      |    **
C**   | B(K) <==> B(LL(2,K));          Y(K)=BK=B(LL(2,K))/L(K,K)    |    **
C**   | Subst. Y(K) to Eq.I=(K+1):N : B(I)=B(I)-L(I,K)*Y(K)         |    **
C**   | A(I,K)=L(I,K), A(K,K)=L(K,K), B(I)=B(I) --> L(I,I)*Y(I)     |    **
C**   +--------------------------------------------------------------+    **
      DO 70 K=1,N
        IK=LL(2,K)
        BK=B(IK)/A(K,K)
        B(IK)=B(K)
        DO 60 I=K+1,N
        B(I)=B(I)-A(I,K)*BK
   60   CONTINUE
        B(K)=BK
   70 CONTINUE
C**   +--------------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward subst. of [U][C]'{X}={Y}      |    **
C**   | X(K) = BK = Y(K) - \sum_{J=K+1}^{N} U(K,J)*X(J)             |    **
C**   | A(K,J)=U(K,J), B(J)=X(J);         X(K) <==> X(LL(1,K))     |    **
C**   +--------------------------------------------------------------+    **
      DO 90 K=N-1,1,-1
        JK=LL(1,K)
        BK=B(K)
        B(K)=B(JK)
        DO 80 J=K+1,N
        BK=BK-A(K,J)*B(J)
   80   CONTINUE
        B(JK)=BK
   90 CONTINUE
      RETURN
      END
**********************************************************************
      SUBROUTINE LUFPBS(A,B,LL,N,NN)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),B(N),LL(2,N)
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward subst. of [U][C]'{X}={Y}      |    **
C**   | X(K) = BK = Y(K) - \sum_{J=K+1}^{N} U(K,J)*X(J)             |    **
C**   | A(K,J)=U(K,J), B(J)=X(J);         X(K) <==> X(LL(1,K))     |    **
C**   +--------------------------------------------------------------+    **
      DO 90 K=N-1,1,-1
        JK=LL(1,K)
        BK=B(K)
        B(K)=B(JK)
        DO 80 J=K+1,N
        BK=BK-A(K,J)*B(J)
   80   CONTINUE
        B(JK)=BK
   90 CONTINUE
      RETURN
      END
**********************************************************************
      SUBROUTINE LUFPSG(A,B,LL,N,NSD,NN)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,N),B(NN,NSD),LL(2,N)
C**   ================================================================    **
```

```
C**   Find the solution vectors [B] of the singular matrix [A]        **
C**   ------------------------------------------------------------    **
C*O   B(N,NSD) = Solution vectors of singular matrix A(N,N)           **
C**   ============================================================    **
      DO 50 J=1,NSD
        DO 30 I=1,N
        B(I,J)=0.0
   30   CONTINUE
        B(N-NSD+J,J)=1.0
        CALL LUFPBS(A,B(1,J),LL,N,NN)
   50 CONTINUE
      RETURN
      END
*************************************************************************
```

3.14 一般矩陣之分解與求解用例

[表七(a)]為一主程式，用以讀入下列聯立式：

$$[A]\{X\} = \{B\}$$

$$\begin{bmatrix} 0.24759 & 0.16235 & 0.46231 \\ 0.14725 & 0.09589 & -.13253 \\ 0.26951 & 0.28965 & -.14794 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0.00647 \\ 0.10475 \\ -.06789 \end{Bmatrix}$$

該程式首先呼叫$LUPPDC$，將矩陣$[A]$做列調換並分解成$[\overline{L}][\overline{U}]$，然後呼叫$LUPPSB$，對向量$\{B\}$做相同之列調換及前進代入並做反向代入而得解$\{X\}$。

通常，如果矩陣$[A]$不是良性矩陣時，所得之解$\{X\}$常有相當誤差而使$[A]\{X\} \neq \{B\}$。如欲求得更精確之$\{X\}$，可將其差值$\{R\} = \{B\} - [A]\{X\}$算出，再由下式計算$\{X\}$之改正值$\{C\}$。

$$[A]\{C\} = \{R\}$$

將$\{X\} + \{C\}$做為新的解，如果誤差仍大，可重覆以上過程數次。[表七(a)]之主程式乃為上述重覆改正法之一範例。

[表七(a)] 一般矩陣之分解與求解用例

```
*************************************************************************
C**   PROGRAM FOR DECOMPOSITION WITH PARTIAL PIVOTING                  **
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 X,AX
```

```
      DIMENSION A(10,10),B(10),LL(10)
      DIMENSION AA(10,10),BB(10),X(10)
      NMAX=10
C**   ============================================================    **
   10 READ(*,'(2I4,F8.6)') N,ITR,EPS
      IF(N.EQ.0.OR.N.GT.NMAX) STOP
      IF(ITR.EQ.0) ITR=1
      IF(EPS.EQ.0.0) EPS=0.000001
      DO 20 I=1,N
   20 READ(*,'(10F8.0)') (A(I,J),J=1,N)
      READ(*,'(10F8.0)') (B(J),J=1,N)
C**   ------------------------------------------------------------    **
      DO 30 J=1,N
      BB(J)=B(J)
      X(J)=0.0
      DO 30 I=1,N
   30 AA(I,J)=A(I,J)
C**   ------------------------------------------------------------    **
      CALL LUPPDC(A,LL,N,NMAX)
      WRITE(*,'(/'' LL(I)'',20I5)') (LL(I),I=1,N)
      WRITE(*,'(/'' DECOMPOSED MATRIX''/)')
      DO 40 I=1,N
   40 WRITE(*,'(1X,10F10.6)') (A(I,J),J=1,N)
C**   ------------------------------------------------------------    **
      DO 80 K=1,ITR
        CALL LUPPSB(A,B,LL,N,NMAX)
        DO 50 J=1,N
   50   X(J)=X(J)+B(J)
        DO 70 I=1,N
        AX=BB(I)
        DO 60 J=1,N
   60   AX=AX-AA(I,J)*X(J)
   70   B(I)=AX
        WRITE(*,'(/'' SOLUTION X''//(1X,10F10.6))') (X(J),J=1,N)
        WRITE(*,'(/'' RESIDUAL R''//(1X,10F10.6))') (B(J),J=1,N)
        DO 75 I=1,N
        IF(ABS(B(I)).GT.EPS) GO TO 80
   75   CONTINUE
        GO TO 10
   80 CONTINUE
      GO TO 10
      END
*************************************************************************
```

[表七(b)] 一般矩陣之分解與求解輸入輸出資料

```
   3   2 0.00001
 0.24759 0.16235 0.46231
 0.14725 0.09589 -.13253
 0.26951 0.28965 -.14794
 0.00647 0.10475 -.06789

LL(I)    3    3    3

DECOMPOSED MATRIX   .269510  1.074728  -.548922
                    .247590  -.103742 -5.766400
                    .147250  -.062364  -.411315

SOLUTION X         1.840817 -2.071955  -.244243

RESIDUAL R          .000000   .000000   .000000
```

3.15 副程式$CVPPDC$-不對稱帶狀矩陣部分樞紐

副程式$CVPPDC$用以對一$N\times N$之矩陣$[A]$做列調換，並分解成$[\overline{L}][\overline{U}]$，且將$[\overline{L}]$與$[\overline{U}]$合併回存於$[A]$。其中$MA$為左半帶寬，$MB$為上半帶寬，$MM$為宣告列數。由於做列調換可能使上半帶寬增為$MA+MB$，故須$MM\geq(MA+MB+MA)$（但可不必大於$N$），且在所有上半帶寬內之零元素須於呼叫本副程式前定義之，向量$LL(2,N)$為暫時運算位置。$LL(2,*)$用以記錄列調換情形：$LL(2,K)=I$時表示第K回合運算前曾將I, K兩列先對調。$LL(1,*)$用以記錄列調換及分解後，$[\overline{U}]$矩陣中每行之非零頂列。$LL(1,K)=I$時表示第K行之非零頂列為I。

副程式$CVPPSB$則利用由$CVPPDC$算得之$[\overline{L}][\overline{U}]$矩陣對向量$\{B\}$做前進代入及根據$LL(2,*)$做必要之列調換，最後根據$LL(1,*)$做反向代入求得$\{X\}$而回存於$\{B\}$。注意該程式也適用於不對稱滿矩陣。

［表八］ 不對稱帶狀矩陣或滿矩陣之分解與求解-部分尋樞紐

```
************************************************************************
      SUBROUTINE CVPPDC(A,LL,N,MA,MB,MM)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(MM,1),LL(2,N)
C**   ================================================================    **
C**   DIMENSION A(MM,N+(N-1)/MM) or A((N-1)*MM+N) : MM >= MA+MB+MA    **
C**   ----------------------------------------------------------------    **
C**   Find [L] & [U] for given [A],         | a11 a12  0   0   0  |    **
C**   Such that  [R][A] = [L][U],           | a21 a22 a23  0   0  |    **
C**   [R] is ROW-exchanges Matrix.   [A] =  |  0  a32 a33 a34  0  |    **
C**   For example:                          |  0   0  a43 a44 a45 |    **
C**   N=5, MA=1, MB=1, MM=3:                |  0   0   0  a54 a55 |    **
C**                                                                    **
C**         | l11  0   0   0   0  |         |  1  u12 u13  0   0  |    **
C**         | l21 l22  0   0   0  |         |  0   1  u23 u24  0  |    **
C**   [L] = |  0  l32 l33  0   0  |, [U] =  |  0   0   1  u34 u35 |    **
C**         |  0   0  l43 l44  0  |         |  0   0   0   1  u45 |    **
C**         |  0   0   0  l54 l55 |         |  0   0   0   0   1  |    **
C**                                                                    **
C**   INPUT  data in A(3,6) =   | a11 a12  0 \a43 a44 a45 |            **
C**                             |\a21 a22 a23  0 \a54 a55 |            **
C**                             |  ? \a32 a33 a34  0 \ ?  |            **
C**   or in 1-D array:                                                 **
C**   A(17) = (a11 a21 ? a12 a22 a32  0  a23 a33 a43  0                **
C**                                      a34 a44 a54  0  a45 a55);     **
C**                                                                    **
C**   OUTPUT data in A(3,6) =   | l11 u12 u13\l43 l44 u45 |            **
C**                             |\l21 l22 u23 u24\l54 l55 |            **
C**                             |  ? \l32 l33 u34 u35\ ?  |            **
C**   or in 1-D array:                                                 **
C**   A(17) = (l11 l21 ? u12 l22 l32 u13 u23 l33 l43 u24               **
C**                                      u34 l44 l54 u35 u45 l55).     **
C**   ================================================================    **
```

```
C**   +--------------------------------------------------------+    **
C**   | Initialize LL(1,J) : Top_Non_Zero_Row_No. of J_th col.  |    **
C**   +--------------------------------------------------------+    **
      DO 10 J=1,N
   10 LL(1,J)=MAX0(J-MB,1)
C**   +--------------------------------------------------------+    **
C**   | MC  : New Upper_Band_Width = max of (KMB-K)            |    **
C**   | KMB : Right_Non_Zero_Col_No. of K_th row (Pivoting_row) |    **
C**   +--------------------------------------------------------+    **
      MC=MB
      KMB=1
      DO 80 K=1,N
      KMA=MIN0(K+MA,N)
C**     +------------------------------------------------------+    **
C**     | Find Pivoting element A(I,K) for I=K,KMA             |    **
C**     +------------------------------------------------------+    **
        AKK=0.0
        DO 20 I=K,KMA
        IF(AKK.LT.ABS(A(I,K))) AKK=ABS(A(I,K))
   20   CONTINUE
        IF(AKK.EQ.0.0) GO TO 90
        AKK=AKK*0.25
        DO 30 I=K,KMA
        IF(AKK.LE.ABS(A(I,K))) GO TO 35
   30   CONTINUE
C**     +------------------------------------------------------+    **
C**     | KMB : Right_Non_Zero_Col_No of K_th row (Pivoting_row)|    **
C**     | Update MC=max(MC,KMB-K)                              |    **
C**     +------------------------------------------------------+    **
   35   KMB=MIN0(MAX0(MB+I,KMB),N)
        MC=MAX0(KMB-K,MC)
        IF(MA+MC.GT.MM) STOP
C**     +------------------------------------------------------+    **
C**     | Set LL(2,K)=I  &  Exchange K_th_row & I_th_row       |    **
C**     | Update LL(1,J)=min(LL(1,J),K)                        |    **
C**     +------------------------------------------------------+    **
        LL(2,K)=I
        IF(K.NE.I) THEN
          DO 40 J=K,KMB
          IF(LL(1,J).GT.K) LL(1,J)=K
          AKK=A(I,J)
          A(I,J)=A(K,J)
          A(K,J)=AKK
   40     CONTINUE
        ENDIF
C**     +------------------------------------------------------+    **
C**     | A(I,J)=A(I,J) - L(I,K)*U(K,J) for I=K+1,KMA & J=K+1,KMB|    **
C**     | A(I,K)=L(I,K),  A(K,J)=L(K,K)*U(K,J) --> U(K,J)=AKJ  |    **
C**     +------------------------------------------------------+    **
        IF(K.EQ.N) RETURN
        DO 60 J=K+1,KMB
          AKJ=A(K,J)/A(K,K)
          DO 50 I=K+1,KMA
          A(I,J)=A(I,J)-A(I,K)*AKJ
   50     CONTINUE
          A(K,J)=AKJ
   60   CONTINUE
   80 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Return on singular matrix : L(K,K)=0                   |    **
```

```
C**   +--------------------------------------------------------+    **
   90 WRITE(*,'(/'' Matrix is singular: A('',I3,'','',I3,'')=0''/)') K,K
      RETURN
      END
*************************************************************************
      SUBROUTINE CVPPSB(A,B,LL,N,MA,MB,MM)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(MM,1),B(N),LL(2,N)
C**   ============================================================    **
C**   Find {X} from [A]{X}={B} for given {B} & [R][A]=[L][U].         **
C**   [L] & [U] are given & put in A(NN,N), [R] is given by LL(2,N)   **
C**   [L][U]{X}=[R]{B}  ==>  [L]{Y}=[R]{B}, [U]{X}={Y}               **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Find {Y} from [R]{B} by forward subst. of [L]{Y}=[R]{B}  |    **
C**   | For K=1:N  Compute B(I)=B(I)-L(I,K)*Y(K) for I=(K+1):KMA |    **
C**   | A(I,K)=L(I,K), B(K)=Y(K), B(I)=B(I) --> Y(I)=BK/L(K,K)   |    **
C**   +----------------------------------------------------------+    **
      DO 70 K=1,N
      KMA=MIN0(K+MA,N)
        I=LL(2,K)
        BK=B(I)/A(K,K)
        B(I)=B(K)
        DO 60 I=K+1,KMA
        B(I)=B(I)-A(I,K)*BK
   60   CONTINUE
        B(K)=BK
   70 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | For K=N:2  Compute B(I)=B(I)-U(I,K)*X(K) for I=KMB:(K-1) |    **
C**   | A(I,K)=U(I,K), B(K)=X(K), B(I)=Y(I) ==> X(K)=BK          |    **
C**   +----------------------------------------------------------+    **
      DO 90 K=N,2,-1
      KMB=LL(1,K)
        BK=B(K)
        DO 80 I=KMB,K-1
        B(I)=B(I)-A(I,K)*BK
   80   CONTINUE
   90 CONTINUE
      RETURN
      END
*************************************************************************
```

習題

1. 試寫一主程式安排下列方程式後呼叫本章之適當副程式將方程式解

出並印出。其正確結果為$x_1 = x_2 = x_3 = x_4 = 1$。

$$\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 25/12 \\ 77/60 \\ 19/20 \\ 319/420 \end{Bmatrix}$$

2. 計算下列對稱帶狀矩陣之聯立方程式之解。

$$\begin{bmatrix} 4 & 2 & 1 & & & & \\ 2 & 4 & 2 & 1 & & & \\ 1 & 2 & 4 & 2 & 1 & & \\ & 1 & 2 & 4 & 2 & 1 & \\ & & 1 & 2 & 4 & 2 & 1 \\ & & & 1 & 2 & 4 & 2 \\ & & & & 1 & 2 & 4 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix}$$

3. 試寫一主程式安排上題等式中之方陣並呼叫本章適當副程式求出方陣之逆矩陣。

4. 參考[表六(b)]副程式之做法，將[表五]副程式$LUPPDC$及$LUPPSB$修改為不做已算出之$[L]$之列對調，注意原有副程式之分解運算方式不適宜達成此目地，因此須採用[表六(b)]程式所用之方式。

第四章

大型稀疏矩陣之分解

4.1 前言

對於n元線性聯立式$[A]\{X\}=\{B\}$，若其係數矩陣$[A]$為奇異，或於矩陣分解時對角元素（設為$\bar{a}_{ii}$）為零，一般須做列或行調換，使做為樞紐元素之該對角元素不為零。但此動作會影響稀疏矩陣（帶狀矩陣或變寬帶矩陣）之元素儲存順序，而造成分解運算時之困難。本章將介紹一種簡單方法可不必做列或行調換，而是加一正值p於該對角元素$\bar{a}_{ii}$，即於聯立式之第i式加入px_i使該樞紐元素不為零。若新增一變數x_k，令其等於x_i，則於第i式再加入$-px_k$可抵消前述加入px_i之影響。$x_k=x_i$之條件可於聯立式之後增列$-px_i+px_k=0$為第k式而達成。原聯立式之解可由擴增聯立式之解直接取得。根據此方法本章分別針對不對稱變寬帶矩陣與對稱變寬帶矩陣 提供二套矩陣分解程式與對應之聯立方程式之求解程式。這些程式極適合於解下列諸問題：(1)接近極限狀態之非線性分析，(2)挫曲後之分析，(3)含束制條件之問題，(4)由特徵值求特徵向量。

4.2 簡介

考慮下列線性聯立方程式：

$$[A]\{X\}=\{B\}$$

或

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{Bmatrix} \tag{4.1}$$

若式中之$[A]$為稀疏矩陣，如帶狀矩陣或變寬帶矩陣，則為保持稀疏矩陣之特點，一般均將矩陣$[A]$分解為下三角矩陣$[L]$，對角矩陣$[D]$與上三角矩陣$[U]$之乘積，亦即$[A]=[L][D][U]$。若分解時遇到對角元素為零，則須做列或行對調以使做為樞紐元素之對角元素不為零。但此運算會影響稀疏矩陣之元素儲存順序，而造成分解運算時之困難。

$Sharifi$與$Popov$[1]建議加一矩陣$c\{B\}\{B\}^T$至矩陣$[A]$使其變為正定矩陣，其中c為正值常數。因此可由下式求得$\{\bar{X}\}$：

$$([A]+c\{B\}\{B\}^T)\{\bar{X}\}=\{B\} \tag{4.2}$$

再由下式可求得原方程式之解$\{X\}$：

$$\{X\}=\{\bar{X}\}(1-c\{B\}^T\{\bar{X}\})^{-1} \tag{4.3}$$

該法雖簡單但有下述二大缺點：(1)對於任意之一般性常數向量$\{B\}$會將稀疏矩陣變為滿矩陣。(2)式(4.3)的簡單關係不能同時適用於二個或以上之右邊常數向量。

$Stewart$[2]也提供一種可避免做列對調之方法。該法係加一正值c於對角元素，設為a_{ii}，而由下式先求得$\{\bar{X}\}$：

$$([A]+c\{e_i\}\{e_i\}^T)\{\bar{X}\}=\{B\} \tag{4.4}$$

式中$\{e_i\}$為單位矩陣之第i行。再利用Sherman–Morrison–Woodbury 公式[3]求原方程式之解$\{X\}$如下：

$$\{X\} = ([\bar{A}]^{-1}-\frac{[\bar{A}]^{-1}\{e_i\}c\{e_i\}^T[\bar{A}]^{-1}}{c\{e_i\}^T[\bar{A}]^{-1}\{e_i\}-1})\{B\} \tag{4.5}$$

$$= \{\bar{X}\}-(\frac{c\bar{x}_i}{c\{e_i\}^T[\bar{A}]^{-1}\{e_i\}-1})[\bar{A}]^{-1}\{e_i\} \tag{4.6}$$

式中$\bar{x}_i$為向量$\{\bar{X}\}$之第i個元素，$[\bar{A}]=[A]+c\{e_i\}\{e_i\}^T$，$[\bar{A}]^{-1}\{e_i\}$為逆矩陣$[\bar{A}]^{-1}$之第$i$行，可由$[\bar{A}]$之分解矩陣$[L][D][U]$做前進及反向代入求得。

該法可能會因矩陣$[A]$為奇異致使式(4.5)與式(4.6)之分母為零而造成問題。且若需處理一個以上之對角元素則顯得相當複雜。

本章所提供方法不僅修改對角元素，同時還增加新變數及新方程式，而使原方程式之變數之解等於修改後之方程式之對應變數之解。因此使得矩陣之分解及代入均十分簡單而直接。

4.3 原理及方法

仔細比較下列擴大後之方程式與式(1)之原方程式，不難看出擴大方程式之前n個變數之解與原方程式之解完全相同。

$$\begin{bmatrix} a_{11} & \cdots & a_{1i} & \cdots & a_{1j} & \cdots & a_{1n} & & \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & & \\ a_{i1} & \cdots & a_{ii}+p_1 & \cdots & a_{ij} & \cdots & a_{in} & -p_1 & \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & & \\ a_{j1} & \cdots & a_{ji} & \cdots & a_{jj}+p_2 & \cdots & a_{jn} & & -p_2 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & & \\ a_{n1} & \cdots & a_{ni} & \cdots & a_{nj} & \cdots & a_{nn} & & \\ & & -p_1 & & & & & p_1 & \\ & & & & -p_2 & & & & p_2 \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_j \\ \vdots \\ x_n \\ x_{n+1} \\ x_{n+2} \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_j \\ \vdots \\ b_n \\ 0 \\ 0 \end{Bmatrix} \tag{4.7}$$

式中之p_k僅當分解過程中之對角元素為零（或很小）且將做為樞紐元素時才加入對角元素中。

為了解此種修改運算之物理意義，茲以桁架結構為例說明之。假設一勁度為$\begin{bmatrix} p & -p \\ -p & p \end{bmatrix}$之虛擬桁架元素被加到原來之結構系統中，而該虛擬元素係連接原系統之一自由度至一不屬於原系統之新自由度，稱虛擬自由度。若無外力作用於該虛擬自由度，則此虛擬自由度之位移必等於其所連接之系統自由度之位移，且系統自由度之位移不會因加入虛擬元素而改變。

4.4 p 值之決定

本節將討論如何決定p值。在以下式計算對角元素時：

$$\bar{a}_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk} d_{kk} u_{kj} \tag{4.8}$$

可以由下式順便求得中間過程之最大絕對值$\tilde{a}_{jj}$：

$$\tilde{a}_{jj} = \max(|a_{jj} - \sum_{k=1}^{i-1} l_{jk} d_{kk} u_{kj}|, \quad i = 1, 2, ..., j) \tag{4.9}$$

若令

$$|\bar{a}_{jj}| = \alpha \cdot \tilde{a}_{jj} \cdot 2^{-s}, \quad 2^{-1} \leq \alpha < 1 \tag{4.10}$$

則可知新的對角元素$\bar{a}_{jj}$已至少失去s個位元之精度（二進位數值）。因此，如果選用$p = \tilde{a}_{jj}$加到對角元素$p = \bar{a}_{jj}$，將可使該做為樞紐元素之對角元素之精度最大，同時又可使捨去之誤差最小。因為，較大的p值雖可以增加樞紐元素的有效位元，但亦會捨去$\bar{a}_{jj}$中原本可能準確的位元。而加入該p值將僅捨去s個本來就不準確的最後位元，因此該值為：在不進一步捨去原有精度下，又能儘量增加樞紐元素之有效位元之考量下之理想選擇。

注意，若式(4.8)中最大絕對值之項，設為$l_{jk} d_{kk} u_{kj}$，早已失去一些位元之精度，則實際的不準位元數s可能較式(4.9)算得者為大。但若想獲知實際的s值，則須計算每個元素之誤差，而須大量的記憶空間 及計算時間。因此，一般實際計算並不考慮去求得每個元素的誤差；而較實際且簡單的做法為儘可能使樞紐元素之有效位元增加。

其次應注意，若原矩陣為奇異，則最後對角元素將為零值，此時若試著加一正值使其不為零，則新增之對角元素經分解運算後亦將仍然為零。因此對於最後的零對角元素不能亦不必繼續做修改。詳細做法將於下節介紹。

4.5 程式之處理

做分解計算時有下列二個重要事實應予考慮：

一、對角元素為負值，設為$-c$，而須做為樞紐元素時，雖然可以加一較大正值p使該對角元素變為正值$p-c>0$，但新加入之對角元素經分解運算後仍然會再度變為負值，即$p-\frac{p^2}{p-c}=\frac{-pc}{p-c}<0$。因此表面上看來雖永遠都可將負對角元素變為正值，但事實上如欲使所有負對角元素均變為正值，將永遠做不停。故本文程式不考慮對負對角元素做修改，而將矩陣分解為$[L][D][U]$，其中對角矩陣$[D]$中之元素可為負值。但若矩陣$[A]$為對稱，則$[L]=[U]^T$。

二、對角元素為零值，而須做為樞紐元素時，雖然可以加一正值p使該對角元素變為正值，但如該對角元素為最後一個對角元素，則新加入之對角元素經分解運算後又會再度變為零。理論上，如果分解後的$[U]$矩陣之最後$m\times m$元子矩陣均為零，則表示矩陣之奇異度為m，修改對角元素之運算即應停止並可結束分解運算。而奇異矩陣之聯立式之解可直接由$[U]$矩陣之最後m行用$[U]$矩陣之前面$n-m$行做反向代入運算求得。以下為本文程式之處理方式：

1. 在未分解前：先設定NSD=0, NAT=0, NBT=NOT, NNT=NOT。其中NOT=n。NNT為矩陣擴增後之元數，將隨對角元素擴增而遞增。

2. 開始依序分解NAT+1至NBT行間之矩陣，若遇對角元素為零，則：

2a. 若零元素為第NNT個對角元素，且NSD=0，則可確定矩陣為一度奇異之特別情形。因NSD=0表示允許零對角元素繼續被修改並擴增對角元素（見步驟2b1.），但至最後一行（即第NBT行）為止並沒有其他需經修改之零對角元素（即$NNT-NBT$=0），故該對角元素為唯一需經修改之對角元素，因此為一度奇異矩陣。此時並設定奇異度為NSD=1，且阻止繼續對零對角元素做修改（見步驟2b2.）。

2b. 依NSD值決定是否對零對角元素做修改：

2b1. 若NSD=0，則修改對角元素，並擴增一新對角元素，且NNT加1。

2b2. 若NSD>0，則不再修改對角元素，亦不擴增新對角元素。

3. 分解完NBT行後：$NNT-NBT$為新擴增之對角元素之數目，若大於零則須回步驟2.繼續分解新增之矩陣。但對下列情況須先修改NSD

值(原為0)，然後令NAT=NBT, NBT=NNT ，回步驟2.。

(1)對於對稱矩陣：若$NNT-NBT$等於$NBT-NAT$且NAT=0：即表示新擴增的對角元素與前一次擴增的對角元素數目相同，而可確定最後一次擴增之子矩陣經分解運算後全為零，故設定NSD=$NNT-NBT$，如此可阻止回步驟2.後繼續對零對角元素做反覆不停之修改（見步驟2b2.），因此在回步驟2.完成後即會結束分解之運算。

(2)對於不對稱矩陣：直接設定NSD=$NNT-NBT$，如此可阻止回步驟2.後繼續對零對角元素做修改（見步驟2b2.）。回步驟2.僅用以補做新增矩陣元素於高斯消去之前NOT個回合應做之運算，此步驟完成後即由步驟4.改用徹底尋樞紐法對新增之$NSD \times NSD$子矩陣做矩陣分解。

4. 此時為$NNT-NBT$=0即無新增元素待做分解：

(1)對於對稱矩陣：則矩陣分解完成，回呼叫程式。

(2)對於不對稱矩陣：則將新增之$NSD \times NSD$子矩陣取出，直接呼叫第三章之[表六(b)]之副程式$LUFPDC$用徹底尋樞紐法做矩陣分解。用徹底尋樞紐法做矩陣分解之目的是可以獲得矩陣之奇異度數。（用部分尋樞紐法一但確定矩陣為奇異後即無法繼續做分解運算，除非分解運算已至最後一列可確定為一度奇異外，否則即無法知道還有多少奇異度。）另一個用徹底尋樞紐法的原因是如下列矩陣之零對角元素無法以擴增新對角元素之方式消除。由於擴增之子矩陣元數$NSD \times NSD$一般不會很大，因此以滿矩陣用徹底尋樞紐法處理應為理想之做法。

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

另請注意：對稱矩陣只有在矩陣元素全為零時，其零對角元素才無法以擴增新對角元素之方式消除。但此時奇異度數已獲知，且分解運算已算完成，因此可結束運算並不造成問題。

另外本章程式在$NSD > 0$時雖不再修改零對角元素為正值，但為了以後求解之方便，仍將其改為不為零之非常小之正值，使求解時不會發生除以零之錯誤，同時又可能將奇異解以相當大的值顯現出來。但須注意

，即使有此項功能，等式右邊常數向量對應於對角元素為零之元素經前進代入後之值仍須不為零，此點一般可利用所謂的不完美載重達成。不過要求得奇異解，最簡單也是最直接的方法，為以分解後之上三角矩陣之最後NSD行利用其前面$NNT-NSD$行做反向代入運算求得，此項運算可以直接呼叫副程式$VSSNGX$或$USSNGX$完成之。

4.6 對稱變寬帶矩陣之分解與求解程式

本章程式所處理之矩陣須採用變寬帶儲存方式（帶狀矩陣會因未知數的擴增而增加帶寬，且帶寬可能增加很多。不過帶狀矩陣實為變寬帶矩陣之特例，以變寬帶矩陣處理並不造成問題）。本章共提供二套程式以分別處理對稱矩陣及不對稱矩陣之分解與求解。資料之儲存方式將分別以二個實例說明之。

例一：對稱矩陣

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4.11}$$

上列聯立方程式之係數為3階對稱矩陣，經擴增為下列之5階對稱矩陣

$$\begin{bmatrix} 0+1 & 1 & 1 & -1 & \\ 1 & 1+1 & 0 & 0 & -1 \\ 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ & -1 & 0 & 0 & 1 \end{bmatrix}$$

並分解為

$$\begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & -1 & 1 & & \\ -1 & 1 & -\frac{2}{3} & 1 & \\ & -1 & \frac{1}{3} & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & -3 & & \\ & & & \frac{1}{3} & \\ & & & & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 & \\ & 1 & -1 & 1 & -1 \\ & & 1 & -\frac{2}{3} & \frac{1}{3} \\ & & & 1 & 1 \\ & & & & 0 \end{bmatrix} \tag{4.12}$$

因此擴增方程式之解可由下式求得

$$\begin{bmatrix} 1 & 1 & 1 & -1 & \\ & 1 & -1 & 1 & -1 \\ & & 1 & -\frac{2}{3} & \frac{1}{3} \\ & & & 1 & 1 \\ & & & & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4.13}$$

或

$$\begin{bmatrix} 1 & 1 & 1 & -1 \\ & 1 & -1 & 1 \\ & & 1 & -\frac{2}{3} \\ & & & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = -x_5 \begin{Bmatrix} 0 \\ -1 \\ \frac{1}{3} \\ 1 \end{Bmatrix} \tag{4.14}$$

由反向代入可得擴增方程式之解為

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = -x_5 \begin{Bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{Bmatrix} \quad \text{or} \quad \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = -x_5 \begin{Bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{Bmatrix} \tag{4.15}$$

因此原方程式之解為$\{X\} = -x_2\{1, -1, 1\}$。

擴增之5階對稱矩陣之下三角或上三角元素之儲存順序如下：

$$\begin{bmatrix} 1 & & & & \\ 2 & 3 & & & \\ 4 & 5 & 6 & & \\ 7 & 8 & 9 & 10 & \\ & 11 & 12 & 13 & 14 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 4 & 7 & \\ & 3 & 5 & 8 & 11 \\ & & 6 & 9 & 12 \\ & & & 10 & 13 \\ & & & & 14 \end{bmatrix}$$

儲存指標$LL(*)$之內容為使i+$LL(j)$值對應到元素a_{ij}之儲存順序。例如：a_{34}之i=3, j=4，而$LL(4)$=6，3+$LL(4)$=9，即a_{34}存於A之第9個位置。依此規則可推知：$LL(1)$=0，其餘之$LL(J)$等於上三角部分之第J行除對角元素外應予儲存之元素之數目加$LL(J-1)$。

因此呼叫副程式$VSDECP$時部分參數之內容為：

$A(1:6)=\{0,1,1,1,0,-1\}$, $LL(1:3)=\{0,1,3\}$, $NOT=3$ 。

經副程式$VSDECP$分解後部分參數之內容為：

$A\,(1:14)=\{1,1,1,1,-1,1,-1,1,-\frac{2}{3},1,-1,\frac{1}{3},1,0\}$

$LL(1:5)=\{0,1,3,6,9\}$, $NOT=3$, $NAT=4$, $NBT=5$ 。

［表一］ 對稱矩陣之分解與求解副程式

```
************************************************************************
      SUBROUTINE VSDECP(A,LL,NOT,NNT,NSD,MAXA,MAXN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(MAXA),LL(MAXN)
      REAL*8 EPS,ADD
      DATA EPS/1.0D-7/
C**   ============================================================    **
C**   Modified Cholesky decomposition [A]=[U'][D][U]                  **
C**   ------------------------------------------------------------    **
C*I   A(NOT+LL(NOT)) = The given original matrix A(NOT,NOT)           **
C*O   A(NNT+LL(NNT)) = Decomposed matrix D+U-I : U(NNT,NNT),D(NNT)    **
C*I   LL(1:NOT)      = Original storage index of A(NOT,NOT)           **
C*O   LL(NOT+1:NNT)  = Expanded storage index OF U(NNT,NNT)           **
C*I   NOT     = The order of the original matrix A(NOT,NOT)          **
C*O   NNT     = The order of the expanded matrix A(NNT,NNT)          **
C*O   NSD     = Number of singular degrees of the given matrix       **
C*I   MAXA    = The maximum allowed dimension of A (MAXA)            **
C*I   MAXN    = The maximum allowed dimension of LL(MAXN)            **
C**   EPS     = Epsilon for accuracy control                         **
C**   ------------------------------------------------------------    **
C**   Find [D] & [U] for given [A], Such that  [A] = [U]'[D] [U]      **
C**   where [U]'= transpose of [U], [D] is a diagonal matrix.        **
C**   For example:                                                   **
C**   Input : NOT=3, LL(3)=(0 1 3), [A] is a NOT*NOT matrix          **
C**   Output: NNT=5, LL(5)=(0 1 3 6 9), [D][U] are NNT*NNT matrices  **
C**                                                                  **
C**         | a11 a12 a13 |           | a11+p1 a12   a13 -p1  0  |    **
C**   [A] = | a12 a22 a23 |           |  a12  a22+p2 a23  0  -p2 |    **
C**         | a13 a23 a33 |           |  a13   a23   a33  0   0  |    **
C**   If it is expanded to ---> [A] = |  -p1    0     0  +p1  0  |    **
C**   Then decomposed matrices are:   |   0    -p2    0   0  +p2 |    **
C**                                                                  **
C**         | d11  0   0   0   0  |         |  1  u12 u13 u14  0  |    **
C**         |  0  d22  0   0   0  |         |  0   1  u23 u24 u25 |    **
C**   [D] = |  0   0  d33  0   0  |, [U] =  |  0   0   1  u34 u35 |    **
C**         |  0   0   0  d44  0  |         |  0   0   0   1  u45 |    **
C**         |  0   0   0   0  d55 |         |  0   0   0   0   1  |    **
C**                                                                  **
C*O  LL(5):( 0       1           3 |             6               9)   **
C**I+LL(J):1+0 1+1 2+1 1+3 2+3 3+3 1+6 2+6 3+6 4+6 2+9 3+9 4+9 5+9   **
C**   Loc : 1   2   3   4   5   6   7   8   9   10  11  12  13  14   **
C*I  A(6):(a11 a12 a22 a13 a23 a33)                                  **
C*O A(14):(d11 u12 d22 u13 u23 d33 u14 u24 u34 d44 u25 u35 u45 d55)  **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Initial set NSD=0 : this also allows matrix to expend    |    **
C**   | Set to decompose the original given matrix A(NOT,NOT)    |    **
C**   +----------------------------------------------------------+    **
```

```
      NSD=0
      NAT=0
      NBT=NOT
      NNT=NOT
C**   +--------------------------------------------------------+    **
C**   | Decompose the sub-matrix A((NAT+1):NBT,(NAT+1):NBT)     |    **
C**   +--------------------------------------------------------+    **
C**   | D(I,I)U(I,J)=A(I,J)-\sum_{K=IM}^{I-1} U(K,I)D(K,K)U(K,J) |    **
C**   | A(K+IA)=U(K,I),     A(K+JA)=D(K,K)*U(K,J)               |    **
C**   | A(I+JA)=AIJ=A(I,J) --> D(I,I)*U(I,J)                    |    **
C**   +--------------------------------------------------------+    **
      JB=0
   10 DO 80 J=NAT+1,NBT
      JA=LL(J)-LL(1)
      JM=J-JA+JB
      JB=JA
        DO 30 I=JM+1,J-1
        IA=LL(I)-LL(1)
        IM=MAX0(I-LL(I)+LL(I-1),JM)
          AIJ=A(I+JA)
          DO 20 K=IM,I-1
          AIJ=AIJ-A(K+IA)*A(K+JA)
   20     CONTINUE
          A(I+JA)=AIJ
   30   CONTINUE
C**   +--------------------------------------------------------+    **
C**   | D(J,J) =A(J,J) - \sum_{K=JM}^{J-1} U(K,J)*D(K,K)*U(K,J)  |    **
C**   | A(K+JA)=D(K,K)*U(K,J) --> U(K,J),  AKJ=U(K,J)           |    **
C**   | A(J+JA)=AJJ=A(J,J) --> D(J,J),     A(K+KA)=D(K,K)       |    **
C**   +--------------------------------------------------------+    **
        AJJ=A(J+JA)
        ADD=DABS(AJJ)
        DO 40 K=JM,J-1
        KA=LL(K)-LL(1)
          AKJ=A(K+JA)/A(K+KA)
          AJJ=AJJ-AKJ*A(K+JA)
          A(K+JA)=AKJ
          ADD=DMAX1(ADD,DABS(AJJ))
   40   CONTINUE
        A(J+JA)=AJJ
C**     +--------------------------------------------------------+    **
C**     | IF the pivoting element A(J,J) > ADD*EPS ?              |    **
C**     | Yes:80 : OK and Continue                                |    **
C**     | No     : Add pivoting by ADD and Expand one column      |    **
C**     +--------------------------------------------------------+    **
        IF(DABS(A(J+JA)).GT.ADD*EPS) GO TO 80
C**     +--------------------------------------------------------+    **
C**     | IF ADD.EQ.0 ?                                           |    **
C**     | Yes: No reference value available, get from diagonal    |    **
C**     +--------------------------------------------------------+    **
        IF(ADD.EQ.0.0) THEN
          DO 45 K=NAT+1,NBT
          KA=LL(K)-LL(1)
          ADD=DMAX1(ADD,DABS(A(K+KA)))
   45     CONTINUE
        ENDIF
        IF(ADD.EQ.0.0) ADD=1.0
C**     +--------------------------------------------------------+    **
C**     | If J.EQ.NNT.AND.NSD.EQ.0                                |    **
C**     | Then it is the Only & the Last ZERO_diagonal,           |    **
```

```
C**     | So set NSD=1 to prevent from any further expanding     |    **
C**     +--------------------------------------------------------+    **
        IF(J.EQ.NNT.AND.NSD.EQ.0) NSD=1
C**     +--------------------------------------------------------+    **
C**     | If NSD.GT.0 : NO MORE expanding is allowed             |    **
C**     +--------------------------------------------------------+    **
        IF(NSD.GT.0) THEN
          A(J+JA)=DMAX1(DABS(A(J+JA)),ADD*EPS*EPS)
C**     +--------------------------------------------------------+    **
C**     | If NSD.EQ.0 : Expand the matrix by                     |    **
C**     | Modify the diagonal : A(J,J)=A(J,J)+ADD                |    **
C**     | Add column : A(J,NNT)=-ADD, A(NNT,NNT)=ADD, A(*,NNT)=0 |    **
C**     +--------------------------------------------------------+    **
        ELSE
          A(J+JA)=A(J+JA)+ADD
          NNT=NNT+1
          IF(NNT.GT.MAXN) STOP 'VSDECP : NNT.GT.MAXN'
          LL(NNT)=LL(NNT-1)+NNT-J
          JA=LL(NNT)-LL(1)
          IF(NNT+JA.GT.MAXA) STOP 'VSDECP : NNT+JA.GT.MAXA'
          A(J+JA)=-ADD
          DO 60 I=J+1,NNT-1
          A(I+JA)=0.0
   60     CONTINUE
          A(NNT+JA)=ADD
        ENDIF
   80 CONTINUE
C**   +--------------------------------------------------------+    **
c**   | If NNT.GT.NBT Then decompose the newly expand matrix    |    **
C**   +--------------------------------------------------------+    **
      IF(NNT.GT.NBT) THEN
C**     +--------------------------------------------------------+    **
C**     | If newly expand numbers .EQ. previously expand numbers |    **
C**     | then it indicates that :                               |    **
C**     | A((NBT+1):NNT,(NBT+1):NNT)=A((NAT+1):NBT,(NAT+1):NBT)  |    **
C**     | So set NSD=NNT-NBT to prevent from ANY further expand  |    **
C**     +--------------------------------------------------------+    **
        IF(NNT-NBT.EQ.NBT-NAT.AND.NAT.NE.0) NSD=NNT-NBT
C**     +--------------------------------------------------------+    **
C**     | Decompose the newly expand matrix (NBT+1):NNT          |    **
C**     +--------------------------------------------------------+    **
        NAT=NBT
        NBT=NNT
        GO TO 10
      ENDIF
      RETURN
      END
*************************************************************************
      SUBROUTINE VSSOLX(A,P,LL,NOT,NNT,B)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),P(NOT),LL(NNT),B(NNT)
C**   ============================================================    **
C**   DIMENSION A(NNT+LL(NNT)),P(NOT),LL(NNT),B(NNT)                  **
C**   ------------------------------------------------------------    **
C**   Find {X} from [A]{X}={P} for given {P} & [A]=>[U]'[D][U].       **
C**   [D] & [U] are given & put in A(*)  (Expanded by sub. VSDECP)    **
C**   [U]'[D][U]{X}={B} ===> [U]'{Z}={B},  [D]{Y}={Z},  [U]{X}={Y}    **
C**   ------------------------------------------------------------    **
C*I   P(NOT) = The given constant vector                              **
```

```
C*O   P(NOT) = The solution vector                                    **
C*W   B(NNT) = The expanded vector (form by VSSOLX)                   **
C**   ============================================================    **
C**   +------------------------------------------------------------+  **
C**   | Expand P(NOT) to B(NNT)                                    |  **
C**   +------------------------------------------------------------+  **
      DO 40 I=1,NOT
      B(I)=P(I)
   40 CONTINUE
      DO 45 I=NOT+1,NNT
      B(I)=0.0
   45 CONTINUE
C**   +------------------------------------------------------------+  **
C**   | Find {Z} from {B} by forward substitution of [U]'{Z}={B}   |  **
C**   | Z(I) = BI = B(I) - \sum_{K=IM}^{I-1} U'(I,K)*Z(K)          |  **
C**   | A(K+IA)=U(K,I)=U'(I,K), B(K)=Z(K), B(I)=B(I) --> Z(I)      |  **
C**   +------------------------------------------------------------+  **
      DO 70 I=2,NNT
      IA=LL(I)-LL(1)
      IM=I-LL(I)+LL(I-1)
        BI=B(I)
        DO 60 K=IM,I-1
        BI=BI-A(K+IA)*B(K)
   60   CONTINUE
        B(I)=BI
   70 CONTINUE
C**   +------------------------------------------------------------+  **
C**   | Find {Y} from {Z} from the equation [D]{Y}={Z}             |  **
C**   | Y(I) = Z(I) / D(I,I)                                       |  **
C**   | B(I) = Z(I) --> Y(I),     A(I+IA)=D(I,I)                   |  **
C**   +------------------------------------------------------------+  **
      DO 75 I=1,NNT
      IA=LL(I)-LL(1)
        B(I)=B(I)/A(I+IA)
   75 CONTINUE
C**   +------------------------------------------------------------+  **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y}  |  **
C**   | For I=N:2  Compute  B(K)=B(K)-U(K,I)*X(I) for K=IM:(I-1)  |  **
C**   | A(K+IA)=U(K,I),     B(I)=BI=X(I),     B(K)=Y(K) --> X(K)  |  **
C**   +------------------------------------------------------------+  **
      DO 90 I=NNT,2,-1
      IA=LL(I)-LL(1)
      IM=I-LL(I)+LL(I-1)
        BI=B(I)
        DO 80 K=IM,I-1
        B(K)=B(K)-A(K+IA)*BI
   80   CONTINUE
   90 CONTINUE
C**   +------------------------------------------------------------+  **
C**   | Move solution vector B(NOT) to P(NOT)                      |  **
C**   +------------------------------------------------------------+  **
      DO 95 I=1,NOT
      P(I)=B(I)
   95 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE VSBKSB(A,B,LL,NOT,NNT)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      DIMENSION A(1),B(NNT),LL(NNT)
C**   ============================================================    **
C**   DIMENSION A(NNT+LL(NNT)),B(NNT),LL(NNT)                         **
C**   ------------------------------------------------------------    **
C**   Find {X} from {Y} by backward substitution of [U]{X}={Y}        **
C**   ------------------------------------------------------------    **
C*I   B(NNT) = The given vector {Y}                                   **
C*O   B(NNT) = The solution vector {X}                                **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | For I=NNT:2 Compute B(K)=B(K)-U(K,I)*X(I) for K=IM:(I-1) |    **
C**   | A(K+IA)=U(K,I),     B(I)=BI=X(I),     B(K)=Y(K) --> X(K) |    **
C**   +----------------------------------------------------------+    **
      DO 90 I=NNT,2,-1
      IA=LL(I)-LL(1)
      IM=I-LL(I)+LL(I-1)
        BI=B(I)
        DO 80 K=IM,I-1
        B(K)=B(K)-A(K+IA)*BI
   80   CONTINUE
   90 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE VSSNGX(A,S,LL,NOT,NNT,NSD,B)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),S(1),LL(NNT),B(NNT)
C**   ============================================================    **
C**   DIMENSION A(NNT+LL(NNT)),S(NOT,NSD),LL(NNT),B(NNT)              **
C**   ------------------------------------------------------------    **
C**   Find the solution vectors [S] of the singular matrix [A]        **
C**   ------------------------------------------------------------    **
C*I   A(NNT+LL(NNT)) = Decomposed matrix D+U-I : U(NNT,NNT),D(NNT)     **
C*I   LL(NNT)        = Expanded storage index OF U(NNT,NNT)           **
C*O   S(NOT,NSD) = Solution vectors of singular matrix A(NOT,NOT)     **
C*I   NOT     = The order of the original matrix A(NOT,NOT)           **
C*I   NNT     = The order of the expanded matrix A(NNT,NNT)           **
C*I   NSD     = Number of singular degrees of the given matrix        **
C*W   B(NNT)  = The working array                                     **
C**   ============================================================    **
      DO 50 J=NNT-NSD+1,NNT
      JS=(J-NNT+NSD-1)*NOT
        DO 30 I=1,NNT
        B(I)=0.0
   30   CONTINUE
        B(J)=1.0
        CALL VSBKSB(A,B,LL,NOT,NNT)
        DO 45 I=1,NOT
        S(I+JS)=B(I)
   45   CONTINUE
   50 CONTINUE
      RETURN
      END
*************************************************************************
```

［表二］ 對稱矩陣分解副程式之試用程式

```
*************************************************************************
```

```
      PROGRAM VSTEST
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(2000),B(2000),LL(100),P(100)
      DATA MAXA,MAXN/2000,100/
C**   ============================================================    **
   10 READ (*,'(20I4)') NOT,NLT
      IF(NOT.LE.0) STOP
      READ (*,'(20I4)') (LL(I),I=1,NOT)
      READ (*,'(10F8.0)') A(1)
      DO 20 J=2,NOT
   20 READ (*,'(10F8.0)') (A(I),I=LL(J-1)+J,LL(J)+J)
      READ (*,'(10F8.0)') (B(I),I=1,NOT*NLT)
C**   ------------------------------------------------------------    **
      WRITE(*,'(/'' NOT,NLT:    '',9I6)') NOT,NLT
      WRITE(*,'('' LL(NOT):    '',10I6)') (LL(I),I=1,NOT)
      WRITE(*,'('' A(*,*):'',1P6E12.5)') A(1)
      DO 30 J=2,NOT
   30 WRITE(*,'('' A(*,*):'',1P6E12.5)') (A(I),I=LL(J-1)+J,LL(J)+J)
      WRITE(*,'('' B(*,*):'',1P6E12.5)') (B(I),I=1,NOT*NLT)
C**   ------------------------------------------------------------    **
      CALL VSDECP(A,LL,NOT,NNT,NSD,MAXA,MAXN)
C**   ------------------------------------------------------------    **
      WRITE(*,'('' NOT,NNT,NSD:'',10I6)') NOT,NNT,NSD
      WRITE(*,'('' LL(NNT):    '',10I6)') (LL(I),I=1,NNT)
      WRITE(*,'('' U(*,*):'',1P6E12.5)') A(1)
      DO 60 J=2,NNT
   60 WRITE(*,'('' U(*,*):'',1P6E12.5)') (A(I),I=LL(J-1)+J,LL(J)+J)
C**   ------------------------------------------------------------    **
      DO 70 J=1,NLT
      JB=(J-1)*NOT
      CALL VSSOLX(A,B(1+JB),LL,NOT,NNT,P)
   70 WRITE(*,'('' X(*,*):'',1P6E12.5)') (B(I+JB),I=1,NOT)
C**   ------------------------------------------------------------    **
      IF(NSD.GT.0) THEN
        CALL VSSNGX(A,B,LL,NOT,NNT,NSD,P)
        DO 95 J=1,NSD
        JB=(J-1)*NOT
   95   WRITE(*,'('' S(*,*):'',1P6E12.5)') (B(I+JB),I=1,NOT)
      ENDIF
      GO TO 10
      END
*************************************************************************
```

［表三］ 例一之輸出 輸入

```
NOT,NLT:         3     1                                        3,1
LL(NOT):         0     1     3                                  0,1,3
A(*,*): 0.00000E+00                                           0
A(*,*): 1.00000E+00 1.00000E+00                               1,1
A(*,*): 1.00000E+00 0.00000E+00-1.00000E+00                   1,0,-1
B(*,*): 0.00000E+00 1.00000E+00 0.00000E+00                   0,1,0
NOT,NNT,NSD:     3     5     1
LL(NNT):         0     1     3     6     9
U(*,*): 1.00000E+00
U(*,*): 1.00000E+00 1.00000E+00
U(*,*): 1.00000E+00-1.00000E+00-3.00000E+00
U(*,*):-1.00000E+00 1.00000E+00-6.66667E-01 3.33333E-01
U(*,*):-1.00000E+00 3.33333E-01 1.00000E+00 1.00000E-14
X(*,*):-1.00000E+14 1.00000E+14-1.00000E+14
S(*,*):-1.00000E+00 1.00000E+00-1.00000E+00
```

4.7 不對稱變寬帶矩陣之分解與求解程式

不對稱變寬帶矩陣之儲存分二部分考慮：對角線元素存於下三角矩陣部分，上三角部分則不含對角線元素。分解後上三角矩陣與下三角矩陣之對角線元素均為1，可不用儲存，而留給對角矩陣儲存其對角線元素。儲存時上三角元素以行序排列，下三角元素以列序排列：因第一行無上三角矩陣之元素，故先存第一列之a_{11}；次存第二行之a_{21}及第二列之$a_{21},\ a_{22}$，依序至第n行及第n列為止。以第k行及第k列為例而言：先由第k行最上第一個非零元素向下至對角元素之前一元素即$a_{k-1,k}$為止依序儲存，其後接第k列由最左第一個非零元素向右至對角元素即a_{kk}為止依序儲存。由於各行及各列應儲存元素之數目沒一定規則，因此分別用二組位置指標：$LI(N)$為列指標使a_{ij}元素可由$A(LI(i)+j)$取得；$LJ(N)$為行指標使a_{ij}元素可由$A(i+LJ(j))$取得。

例二：不對稱矩陣

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ & & -1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ -1 \end{Bmatrix} \tag{4.16}$$

上列聯立方程式之係數為3階不對稱矩陣，經擴增為下列之5階矩陣

$$\begin{bmatrix} 0+1 & 1 & 1 & -1 & \\ 1 & 1+1 & 0 & 0 & -1 \\ & & -1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ & -1 & 0 & 0 & 1 \end{bmatrix}$$

並分解為

$$\begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ & & -1 & & \\ -1 & 1 & 2 & -1 & \\ & -1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & -3 & & \\ & & & \frac{1}{3} & \\ & & & & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 & \\ & 1 & -1 & 1 & -1 \\ & & 1 & 0 & 0 \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix} \tag{4.17}$$

常數向量擴增為$\{1,1,-1,0,0\}$後，利用前進代入及反向代入，可得其解為$\{1,0,1,1,0\}$，因此原方程式之解為$\{1,0,1\}$。

擴增之5階不對稱矩陣元素之儲存順序如下：

$$\begin{bmatrix} 1 & 2 & 5 & 8 & \\ 3 & 4 & 6 & 9 & 15 \\ & & 7 & 10 & 16 \\ 11 & 12 & 13 & 14 & 17 \\ & 18 & 19 & 20 & 21 \end{bmatrix}$$

上三角元素第J行之儲存指標為$LR(J)$，下三角元素（含對角元素）第J列之儲存指標為$LL(J)$。$LR(J)$之內容為使$i+LR(j)$值對應到上三角元素a_{ij}之儲存順序。$LL(J)$之內容為使$i+LL(j)$值對應到下三角元素a_{ji}之儲存順序。例如：a_{34}之i=3, j=4，而$LR(4)$=7，3+$LR(4)$=10，即a_{34}存於A之第10個位置。a_{43}之i=3, j=4，而$LL(4)$=10，3+$LL(4)$=13，即a_{43}存於A之第13個位置。依此規則可推知：

$LR(1)$=0，其餘之$LR(J)$等於上三角部分之第J行除對角元素外應予儲存之元素之數目加$LL(J-1)$。本範例之$LR(1:5)=\{0,1,4,7,13\}$。

$LL(1)$=0，其餘之$LL(J)$等於下三角部分之第J列除對角元素外應予儲存之元素之數目加$LR(J)$。本範例之$LL(1:5)=\{0,2,4,10,16\}$。

因此呼叫副程式$USDECP$時部分參數之內容為：NOT=3,
$A(1:7)=\{0,1,1,1,1,0,-1\}$, $LR(1:3)=\{0,1,4\}$, $LL(1:3)=\{0,2,4\}$。

經副程式$VSDECP$分解後部分參數之內容為：NAT=5, NBT=5,
$A\,(1:21)=\{1,1,1,1,1,-1,-1,-1,1,0,-1,1,2,-1,-1,0,-1,-1,-1,1,1\}$,
$LR(1:5)=\{0,1,4,7,13\}$, $LL(1:5)=\{0,2,4,10,16\}$。

［表四］ 不對稱矩陣之分解與求解副程式

```
************************************************************************
      SUBROUTINE USDECP(A,LL,NOT,NNT,NSD,MAXA,MAXN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(MAXA),LL(2,MAXN)
      REAL*8 EPS,ADD
      DATA EPS/1.0D-7/
C**   ============================================================    **
C**   LU decomposition [A]=[L][U]                                     **
C**   ------------------------------------------------------------    **
C*I   A(NOT+L*(NOT))  = The given original matrix A(NOT,NOT)          **
C*O   A(NNT+L*(NNT))  = Decomposed matrix L+U-I : L\U(NNT,NNT)        **
C*I   LL(2,1:NOT)     = Original storage index of A(NOT,NOT)          **
C*O   LL(2,NOT+1:NNT) = Expanded storage index of L\U(NNT,NNT)        **
```

```
C*I   NOT     = The order of the original matrix A(NOT,NOT)          **
C*O   NNT     = The order of the expanded matrix A(NNT,NNT)          **
C*O   NSD     = Number of singular degrees of the given matrix       **
C*I   MAXA    = The maximum allowed dimension of A(MAXA)             **
C*I   MAXN    = The maximum allowed dimension of LL(2,MAXN)          **
C**   EPS     = Epsilon for accuracy control                         **
C**   ---------------------------------------------------------------**
C**   Find [L] & [U] for given [A], Such that  [A] => [L] [U]        **
C**   For example:                                                   **
C**   Input : NOT=3, LR(3)=(0 1 4),      LL(3)=(0 2 4)               **
C**   Output: NNT=5, LR(5)=(0 1 4 7 13), LL(5)=(0 2 4 10 16)         **
C**                                                                  **
C**         | a11 a12 a13 |           | a11+p1 a12   a13 -p1  0  |   **
C**   [A] = | a21 a22 a23 |           |  a21  a22+p2 a23  0  -p2 |   **
C**         |  0   0  a33 |           |   0     0    a33  0   0  |   **
C**   If it is expanded to ---> [A] = |  -p1    0     0  +p1  0  |   **
C**   Then decomposed matrices are:   |   0    -p2    0   0  +p2 |   **
C**                                                                  **
C**         | l11  0   0   0   0  |         |  1  u12 u13 u14  0  |   **
C**         | l21 l22  0   0   0  |         |  0   1  u23 u24 u25 |   **
C**   [L] = |  0   0  l33  0   0  |, [U] =  |  0   0   1  u34 u35 |   **
C**         | l41 l42 l43 l44  0  |         |  0   0   0   1  u45 |   **
C**         |  0  l52 l53 l54 l55 |         |  0   0   0   0   1  |   **
C**                                                                  **
C*I    A(7):(a11 a12 a21 a22 a13 a23 a33)                            **
C*O   A(21):(l11 u12 l21 l22 u13 u23 l33 u14 u24 u34                 **
C*O                 l41 l42 l43 l44 u25 u35 u45 l52 l53 l54 d55)     **
C**   ===============================================================**
C**   LL(1,J)=Storage index of upper triangular part W/O  diagonal   **
C**           A(I,J) is referred as A(I+LL(1,J))                     **
C**   LL(2,I)=Storage index of lower triangular part with diagonal   **
C**           A(I,J) is referred as A(J+LL(2,I))                     **
C**   ---------------------------------------------------------------**
C**   The storage Indexes/Locations of the matrix elements:          **
C**   LL(1,J) 0  1  4  7 13                                          **
C**   LL(2,I)               (I) For examples:                        **
C**      0    1\ 2  5  8      1  2+LL(1,5)=15 for A(2,5) in Loc.15   **
C**      2    3  4\ 6  9 15   2  3+LL(1,5)=16 for A(3,5) in Loc.16   **
C**      4         7\10 16    3  4+LL(1,5)=17 for A(4,5) in Loc.17   **
C**     10   11 12 13 14\17    4  2+LL(2,5)=18 for A(5,2) in Loc.18   **
C**     16      18 19 20 21\   5  3+LL(2,5)=19 for A(5,3) in Loc.19   **
C**                            4+LL(2,5)=20 for A(5,4) in Loc.20     **
C**     (J)   1  2  3  4  5   5+LL(2,5)=21 for A(5,5) in Loc.21     **
C**   ===============================================================**
C**   +-------------------------------------------------------------+**
C**   | Initial set NSD=0 : this also allows matrix to expend       |**
C**   | Set to decompose the original given matrix A(NOT,NOT)       |**
C**   +-------------------------------------------------------------+**
      NSD=0
      NAT=0
      NBT=NOT
      NNT=NOT
C**   +-------------------------------------------------------------+**
C**   | Decompose the sub-matrix A((NAT+1):NBT,(NAT+1):NBT)         |**
C**   +-------------------------------------------------------------+**
      JC=0
   10 DO 80 J=NAT+1,NBT
      JA=LL(2,J)-LL(1,1)
      JB=LL(1,J)-LL(1,1)
      JMA=J-JA+JB
```

```
      JMB=J-JB+JC
      JC=JA
C**     +-------------------------------------------------------+    **
C**     | AIJ = A(I,J) - \sum_{K=IMA}^{I-1} L(I,K)*U(K,J)       |    **
C**     | A(K+IA)=L(I,K),    A(K+JB)=U(K,J)                     |    **
C**     | A(I+JB)=A(I,J) --> U(I,J)=AIJ/L(I,I)                  |    **
C**     +-------------------------------------------------------+    **
        DO 20 I=JMB,J-1
        IA=LL(2,I)-LL(1,1)
        IMA=MAX0(I-LL(2,I)+LL(1,I),JMB)
          AIJ=A(I+JB)
          DO 15 K=IMA,MIN0(I-1,NOT)
          AIJ=AIJ-A(K+IA)*A(K+JB)
   15     CONTINUE
          A(I+JB)=AIJ
          IF(I.LE.NOT) A(I+JB)=AIJ/A(I+IA)
   20   CONTINUE
C**     +-------------------------------------------------------+    **
C**     | AJI = A(J,I) - \sum_{K=IMB}^{I-1} L(J,K)*U(K,I)       |    **
C**     | A(K+JA)=L(J,K),    A(K+IB)=U(K,I)                     |    **
C**     | A(I+JA)=A(J,I) --> L(J,I)=AJI                         |    **
C**     +-------------------------------------------------------+    **
        DO 30 I=JMA+1,J-1
        IB=LL(1,I)-LL(1,1)
        IMB=MAX0(I-LL(1,I)+LL(2,I-1),JMA)
          AJI=A(I+JA)
          DO 25 K=IMB,MIN0(I-1,NOT)
          AJI=AJI-A(K+JA)*A(K+IB)
   25     CONTINUE
          A(I+JA)=AJI
   30   CONTINUE
C**     +-------------------------------------------------------+    **
C**     | Compute L(J,J)                                        |    **
C**     | AJJ = A(J,J) - \sum_{K=JM*}^{J-1} L(J,K)*U(K,J)       |    **
C**     | A(K+JA)=L(J,K),    A(K+JB)=U(K,J)                     |    **
C**     | A(J+JA)=A(J,J) --> L(J,J)=AJJ                         |    **
C**     +-------------------------------------------------------+    **
        AJJ=A(J+JA)
        ADD=DABS(AJJ)
        DO 40 K=MAX0(JMA,JMB),MIN0(J-1,NOT)
        AJJ=AJJ-A(K+JA)*A(K+JB)
        ADD=DMAX1(ADD,DABS(AJJ))
   40   CONTINUE
        A(J+JA)=AJJ
        IF(J.GT.NOT) GO TO 80
C**     +-------------------------------------------------------+    **
C**     | IF the pivoting element A(J,J) > ADD*EPS ?            |    **
C**     | Yes:80 : OK and Continue                              |    **
C**     | No     : Add pivoting by ADD and Expand one column    |    **
C**     +-------------------------------------------------------+    **
        IF(DABS(A(J+JA)).GT.ADD*EPS) GO TO 80
C**     +-------------------------------------------------------+    **
C**     | IF ADD.EQ.0 ?                                         |    **
C**     | Yes: No reference value available, get from diagonal  |    **
C**     +-------------------------------------------------------+    **
        IF(ADD.EQ.0.0) THEN
          DO 45 K=NAT+1,NBT
          KA=LL(2,K)-LL(1,1)
          ADD=DMAX1(ADD,DABS(A(K+KA)))
   45     CONTINUE
```

```
        ENDIF
        IF(ADD.EQ.0.0) ADD=1.0
C**     +-------------------------------------------------------+    **
C**     | If J.EQ.NNT.AND.NSD.EQ.0                              |    **
C**     | Then it is the Only & the Last ZERO_diagonal,         |    **
C**     | So set NSD=1 to prevent from any further expanding    |    **
C**     +-------------------------------------------------------+    **
        IF(J.EQ.NNT.AND.NSD.EQ.0) NSD=1
C**     +-------------------------------------------------------+    **
C**     | If NSD.GT.0 : NO MORE expanding is allowed            |    **
C**     +-------------------------------------------------------+    **
        IF(NSD.GT.0) THEN
          A(J+JA)=DMAX1(DABS(A(J+JA)),ADD*EPS*EPS)
C**     +-------------------------------------------------------+    **
C**     | If NSD.EQ.0 : Expand the matrix by                    |    **
C**     | Modify the diagonal : A(J,J)=A(J,J)+ADD               |    **
C**     | Add column : A(J,NNT)=-ADD, A(NNT,NNT)=ADD, A(*,NNT)=0 |    **
C**     +-------------------------------------------------------+    **
        ELSE
          A(J+JA)=A(J+JA)+ADD
          NNT=NNT+1
          IF(NNT.GT.MAXN) STOP 'USDECP : NNT.GT.MAXN'
          LL(1,NNT)=LL(2,NNT-1)+NNT-J
          LL(2,NNT)=LL(1,NNT)  +NNT-J
          JB=LL(1,NNT)-LL(1,1)
          JA=LL(2,NNT)-LL(1,1)
          IF(NNT+JA.GT.MAXA) STOP 'USDECP : NNT+JA.GT.MAXA'
          A(J+JB)=-ADD
          A(J+JA)=-ADD
          DO 60 I=J+1,NNT-1
          A(I+JB)=0.0
          A(I+JA)=0.0
   60     CONTINUE
          A(NNT+JA)=ADD
        ENDIF
   80 CONTINUE
C**   +-------------------------------------------------------+    **
c**   | If NNT.GT.NBT Then decompose the newly expand matrix   |    **
C**   | Set NSD=NNT-NBT to prevent from further expanding      |    **
C**   +-------------------------------------------------------+    **
      IF(NNT.GT.NBT) THEN
        NSD=NNT-NBT
        NAT=NBT
        NBT=NNT
        GO TO 10
      ENDIF
C**   +-------------------------------------------------------+    **
C**   | If NNT.EQ.NOT Then return, Else:                      |    **
C**   | Form a sub-system [A]{X}={B} to solve by full pivoting |    **
C**   +-------------------------------------------------------+    **
      IF(NNT.EQ.NOT) RETURN
      IJ=NNT+LL(2,NNT)
      IF(NNT+NSD.GT.MAXN) STOP 'USDECP : MAXN TOO SMALL'
      IF(IJ+NSD*NSD.GT.MAXA) STOP 'USDECP : MAXA TOO SMALL'
      DO 95 J=NOT+1,NNT
      DO 90 I=NOT+1,NNT
      IJ=IJ+1
      IF(I.LT.J) A(IJ)=A(I+LL(1,J)-LL(1,1))
      IF(I.GE.J) A(IJ)=A(J+LL(2,I)-LL(1,1))
   90 CONTINUE
```

```
   95 CONTINUE
      NAD=NSD
      CALL LUFPDC(A(NNT+LL(2,NNT)+1),LL(1,NNT+1),NAD,NSD,NAD)
      RETURN
      END
*****************************************************************************
      SUBROUTINE USSOLX(A,P,LL,NOT,NNT,B)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),P(NOT),LL(2,NNT),B(NNT)
C**   ============================================================    **
C**   DIMENSION A(NNT+LL(2,NNT)),P(NOT),LL(2,NNT),B(NNT)              **
C**   ------------------------------------------------------------    **
C**   Find {X} from [A]{X}={P} for given {P} & [A]=>[L][U].           **
C**   [L] & [U] are given & put in A(*)  (Expanded by sub. USDECP)    **
C**   [L][U]{X}={B} ===> [L]{Y}={B},  [U]{X}={Y}                      **
C**   ------------------------------------------------------------    **
C*I   P(NOT) = The given constant vector                              **
C*O   P(NOT) = The solution vector                                    **
C*W   B(NNT) = The expanded vector (form by USSOLX)                   **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Expand P(NOT) to B(NNT)                                  |    **
C**   +----------------------------------------------------------+    **
      DO 40 I=1,NOT
      B(I)=P(I)
   40 CONTINUE
      DO 45 I=NOT+1,NNT
      B(I)=0.0
   45 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find {Y} from {B} by forward substitution of [L]{Y}={B}  |    **
C**   | BI = B(I) - \sum_{K=IMA}^{I-1} L(I,K)*Y(K)               |    **
C**   | A(K+IA)=L(I,K), B(K)=Y(K), B(I)=B(I) --> Y(I)=BI/L(I,I)  |    **
C**   +----------------------------------------------------------+    **
      DO 70 I=1,NNT
      IA=LL(2,I)-LL(1,1)
      IMA=I-LL(2,I)+LL(1,I)
        BI=B(I)
        DO 60 K=IMA,MIN0(I-1,NOT)
        BI=BI-A(K+IA)*B(K)
   60   CONTINUE
        B(I)=BI
        IF(I.LE.NOT) B(I)=BI/A(I+IA)
   70 CONTINUE
C**   ------------------------------------------------------------    **
      NAD=NNT-NOT
      IF(NAD.GT.0) CALL LUFPSB
     *(A(NNT+LL(2,NNT)+1),B(NOT+1),LL(1,NNT+1),NAD,NAD)
C**   +----------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | For I=NNT:2 Compute B(K)=B(K)-U(K,I)*X(I) for K=IMB:(I-1)|    **
C**   | A(K+IB)=U(K,I),     B(I)=BI=X(I),     B(K)=Y(K) --> X(K) |    **
C**   +----------------------------------------------------------+    **
      DO 90 I=NNT,2,-1
      IB=LL(1,I)-LL(1,1)
      IMB=I-LL(1,I)+LL(2,I-1)
        BI=B(I)
        DO 80 K=IMB,MIN0(I-1,NOT)
        B(K)=B(K)-A(K+IB)*BI
```

```
   80   CONTINUE
   90 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Move solution vector B(NOT) to P(NOT)                  |    **
C**   +--------------------------------------------------------+    **
      DO 95 I=1,NOT
      P(I)=B(I)
   95 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE USBKSB(A,B,LL,NOT,NNT)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(NNT),LL(2,NNT)
C**   ========================================================    **
C**   DIMENSION A(NNT+LL(2,NNT)),B(NNT),LL(2,NNT)                 **
C**   --------------------------------------------------------    **
C**   Find {X} from {Y} by backward substitution of [U]{X}={Y}    **
C**   --------------------------------------------------------    **
C*I   B(NNT) = The given vector {Y}                               **
C*O   B(NNT) = The solution vector {X}                            **
C**   ========================================================    **
      NAD=NNT-NOT
      IF(NAD.GT.0) CALL LUFPBS
     *(A(NNT+LL(2,NNT)+1),B(NOT+1),LL(1,NNT+1),NAD,NAD)
C**   +--------------------------------------------------------+    **
C**   | Find {X} from {Y} by backward substitution of [U]{X}={Y} |    **
C**   | For I=NNT:2 Compute B(K)=B(K)-U(K,I)*X(I) for K=IMB:(I-1)|    **
C**   | A(K+IB)=U(K,I),     B(I)=BI=X(I),     B(K)=Y(K) --> X(K) |    **
C**   +--------------------------------------------------------+    **
      DO 90 I=NNT,2,-1
      IB=LL(1,I)-LL(1,1)
      IMB=I-LL(1,I)+LL(2,I-1)
        BI=B(I)
        DO 80 K=IMB,MIN0(I-1,NOT)
        B(K)=B(K)-A(K+IB)*BI
   80   CONTINUE
   90 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE USSNGX(A,S,LL,NOT,NNT,NSD,B)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),S(1),LL(2,NNT),B(NNT)
C**   ========================================================    **
C**   DIMENSION A(NNT+LL(2,NNT)),S(NOT,NSD),LL(2,NNT),B(NNT)      **
C**   --------------------------------------------------------    **
C**   Find the solution vectors [S] of the singular matrix [A]    **
C**   --------------------------------------------------------    **
C*I   A(NNT+L*(NNT)) = Decomposed matrix L+U-I : L\U(NNT,NNT)      **
C*I   LL(2,NNT)      = Expanded storage index of L\U(NNT,NNT)      **
C*O   S(NOT,NSD) = Solution vectors of singular matrix A(NOT,NOT)  **
C*I   NOT     = The order of the original matrix A(NOT,NOT)       **
C*I   NNT     = The order of the expanded matrix A(NNT,NNT)       **
C*I   NSD     = Number of singular degrees of the given matrix    **
C*W   B(NNT)  = The working array                                 **
C**   ========================================================    **
      DO 50 J=NNT-NSD+1,NNT
```

```
      JS=(J-NNT+NSD-1)*NOT
        DO 30 I=1,NNT
        B(I)=0.0
   30   CONTINUE
        B(J)=1.0
        CALL USBKSB(A,B,LL,NOT,NNT)
        DO 45 I=1,NOT
        S(I+JS)=B(I)
   45   CONTINUE
   50 CONTINUE
      RETURN
      END
*************************************************************************
```

［表五］ 不對稱矩陣分解副程式之試用程式

```
*************************************************************************
      PROGRAM USTEST
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(2000),B(2000),LL(2,100),P(100)
      DATA MAXA,MAXN/2000,100/
C**   ============================================================    **
   10 READ (*,'(20I4)') NOT,NLT
      IF(NOT.LE.0) STOP
      READ (*,'(20I4)') (LL(1,I),LL(2,I),I=1,NOT)
      READ (*,'(10F8.0)') A(1)
      DO 20 J=2,NOT
   20 READ (*,'(10F8.0)') (A(I),I=LL(2,J-1)+J,LL(2,J)+J)
      READ (*,'(10F8.0)') (B(I),I=1,NOT*NLT)
C**   ------------------------------------------------------------    **
      WRITE(*,'(/'' NOT,NLT:   '',9I6)') NOT,NLT
      WRITE(*,'('' LL(2,NOT): '',10I6)') (LL(1,I),LL(2,I),I=1,NOT)
      WRITE(*,'('' A(*,*):'',1P6E12.5)') A(1)
      DO 30 J=2,NOT
   30 WRITE(*,'('' A(*,*):'',1P6E12.5)') (A(I),I=LL(2,J-1)+J,LL(2,J)+J)
      WRITE(*,'('' B(*,*):'',1P6E12.5)') (B(I),I=1,NOT*NLT)
C**   ------------------------------------------------------------    **
      CALL USDECP(A,LL,NOT,NNT,NSD,MAXA,MAXN)
C**   ------------------------------------------------------------    **
      WRITE(*,'('' NOT,NNT,NSD'',10I6)') NOT,NNT,NSD
      WRITE(*,'('' LL(2,NNT): '',10I6)') (LL(1,I),LL(2,I),I=1,NNT)
      WRITE(*,'('' U(*,*):'',1P6E12.5)') A(1)
      DO 60 J=2,NNT
   60 WRITE(*,'('' U(*,*):'',1P6E12.5)') (A(I),I=LL(2,J-1)+J,LL(2,J)+J)
C**   ------------------------------------------------------------    **
      DO 70 J=1,NLT
      JB=(J-1)*NOT
      CALL USSOLX(A,B(1+JB),LL,NOT,NNT,P)
   70 WRITE(*,'('' X(*,*):'',1P6E12.5)') (B(I+JB),I=1,NOT)
C**   ------------------------------------------------------------    **
      IF(NSD.GT.0) THEN
        CALL USSNGX(A,B,LL,NOT,NNT,NSD,P)
        DO 95 J=1,NSD
        JB=(J-1)*NOT
   95   WRITE(*,'('' S(*,*):'',1P6E12.5)') (B(I+JB),I=1,NOT)
      ENDIF
      GO TO 10
      END
*************************************************************************
```

[表六] 例二之輸出 輸入

```
NOT,NLT:        3     1                                          3,1
LL(2,NOT):      0     0     1     2     4     4                  0,0,1,2,4,4
A(*,*): 0.00000E+00                                            0
A(*,*): 1.00000E+00 1.00000E+00 1.00000E+00                    1,1,1
A(*,*): 1.00000E+00 0.00000E+00-1.00000E+00                    1,0,-1
B(*,*): 1.00000E+00 1.00000E+00-1.00000E+00                    1,1,-1
NOT,NNT,NSD     3     5     0
LL(2,NNT):      0     0     1     2     4     4     7    10    13    16
U(*,*): 1.00000E+00
U(*,*): 1.00000E+00 1.00000E+00 1.00000E+00
U(*,*): 1.00000E+00-1.00000E+00-1.00000E+00
U(*,*):-1.00000E+00 1.00000E+00 0.00000E+00-1.00000E+00 1.00000E+00 2.00000
U(*,*):-1.00000E+00
U(*,*):-1.00000E+00 0.00000E+00 1.00000E+00-1.00000E+00-1.00000E+00 1.00000
U(*,*): 0.00000E+00
X(*,*): 1.00000E+00 0.00000E+00 1.00000E+00
```

參考文獻

1. Sharifi, P. and Popov, E.P., "Nonlinear Buckling Analysis of Sandwich Arches," Journal of the Engineering Mechanics Division, ASCE, Vol.97, No. EM5, pp.1397–1412, 1971.

2. Stewart, G.W., "Modifying Pivot Elements in Gaussian Elimination," Mathematics of Computation, Vol.28, No.126, pp.537–542, 1974.

3. Householder, A.S., *The Theory of Matrices in Numerical Analysis,* Blaisdell, New York, MR 30 #5475, p.123, 1964.

4. 林聰悟,蕭興臺,黃金田, "Decomposition of Singular Large Sparse Matrix by Adding Dummy Links and Dummy Degrees," Journal of the Chinese Institute of Engineers, Vol.15, No.6, pp.723-727, 1992.

第五章

矩陣之儲存方法

5.1 前言

本章先介紹幾點有關矩陣程式設計常見之問題，如副程式對於矩陣之處理，呼叫矩陣運算副程式之方法，副程式中矩陣宣告階數 (Declared DIMENSION of matrix) 可隨呼叫程式改變之處理方法及動態矩陣分配之簡易方法等。然後，介紹帶狀矩陣之儲存方法及運算。最後，再介紹變寬帶矩陣之儲存方法及運算。

5.2 矩陣程式設計的幾個問題

(1) 使副程式在不影響速率的情況下儘可能增加其廣泛性。

以矩陣相乘副程式為例，一般常寫成如 [表一] 者，其中 [A] 為 N 列 M 行，則其矩陣宣告階數亦須為 A(N,M) 而限制了使用該程式的機會。若改如 [表二] 者，雖其中 [A] 宣告階數為 A(NA,M)，但其實際矩陣為 N 列 M 行，只要 N≤NA 即可。另二矩陣亦類似，雖增加三個參數，但也增加了使用性，且不減少其速率。

[表一] 副程式CMPY之1

```
      SUBROUTINE CMPY(A,B,C,N,M,L)
      DIMENSION A(N,M),B(M,L),C(N,L)
      DO 10 J=1,L
      DO 10 I=1,N
      C(I,J)=0.0
      DO 10 K=1,M
   10 C(I,J)=C(I,J)+A(I,K)*B(K,J)
      RETURN
      END
```

[表二] 副程式CMPY之2

```
      SUBROUTINE CMPY(A,B,C,N,M,L,NA,NB,NC)
      DIMENSION A(NA,M),B(NB,L),C(NC,L)
      DO 10 J=1,L
      DO 10 I=1,N
      C(I,J)=0.0
      DO 10 K=1,M
   10 C(I,J)=C(I,J)+A(I,K)*B(K,J)
      RETURN
      END
```

(2)在一大矩陣中的子矩陣，可以直接由其他副程式操作，而不須將該子矩陣移至另一矩陣中再操作。

將下列之矩陣[A]分成四個子矩陣[A11], [A21], [A12]及[A22]。

$$[A]=\left[\begin{array}{cc|ccccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \end{array}\right]=\left[\begin{array}{cc} A11 & A12 \\ A21 & A22 \end{array}\right]$$

如欲計算[A21]乘[A12]，並把結果存為[A22]，即可直接呼叫[表二]程式如下：

```
DIMENSION A(6,7)
...
CALL CMPY(A(3,1),A(1,3),A(3,3),4,2,5,6,6,6)
```

注意最後三個參數(即6,6,6)皆為[A]矩陣之宣告列數。

如用[表一]之副程式，勢必先將[A21]矩陣移至宣告階數為4 x 2之矩陣中，[A12]移至2 x 5之另一矩陣中，再呼叫[表一]副程式，其乘積矩陣並須再移回[A22]中，如此既增加搬移時間又增加許多處理程式及儲存位置。

至於矩陣與向量之相乘，亦可以直接呼叫[表二]程式，如8x15之矩陣[A]乘以15x1行矩陣{B}得8x1行矩陣{X}可按下列方式呼叫之。

```
DIMENSION A(10,20),B(20),X(10)
...
CALL CMPY(A,B,X,8,15,1,10,15,8) ! CALL CMPY(A,B,X,8,15,1,10,1,1)
```

又如1x8列矩陣⟨A⟩乘以8x15之矩陣[B]得1x15之列矩陣⟨Y⟩可按下列方式呼叫[表二]副程式。

```
DIMENSION A(10),B(10,20),Y(20)
....
CALL CMPY(A,B,Y,1,8,15,1,10,1)
```

其中最後三個參數(即1,10,1)，除10為[B]矩陣之宣告列數外，餘皆須為1。

同理向量與向量之乘積亦可呼叫[表二]副程式如下：

```
DIMENSION A(10),B(15),C(10,15)
...
CALL CMPY(A,B,AB,1,N,1,1,N,1)  ! CALL CMPY(A,B,AB,1,N,1,1,1,N)
...
CALL CMPY(A,B,C,N,1,M,10,1,10) ! CALL CMPY(A,B,C,N,1,M,1,1,10)
```

在以上各例中，亦可用!後所列之指令，其中不同者為有些矩陣之宣告列數可改用1(或其他值)，因為這些矩陣只有一行，而不必用到其宣告列數。

(3)一般常用副程式，如矩陣相乘，求逆矩陣，印出矩陣，解聯立方程式等副程式，最好不要固定其矩陣宣告階數。

如[表二]副程式，其矩陣宣告列數NA,NB,NC，皆由呼叫程式決定，而不須修改程式。其中關鍵指令DIMENSION A(NA,M),B(NB,L), C(NC,L)之宣告列數為呼叫本副程式之虛擬參數 (Dummy argument)，可隨呼叫程式給定之值而改變，一般稱為可變宣告階數 (Variable DIMENSION)。但有少數FORTRAN不允許宣告階數為變數，此時可利用以下介紹之技巧，將矩陣(二維陣列)轉換成以向量(一維陣列)儲存，因此在宣告陣列之階數時，即可不必用到矩陣之列數(為可變之變數)，但程式須用指令將矩陣行列指標換算為向量位置指標，亦即其儲存位置對應之向量指標。類似之技巧亦可用在只須儲存上(或下)三角部分元素之對稱矩陣 或三角矩陣之情形。

例如[表二]副程式中之二維矩陣可改為[表三(a)(b)]副程式中之一維向量。[表三(a)]中之IJ,IK,KJ即為矩陣轉換成以向量儲存時之位置指標，或稱向量指標，其與矩陣之行列指標I,J,K之關係即如表中指令所示，此計算方式係因FORTRAN陣列以行序 (column wise)排列，亦即第一行在前第二行在後之順序，故矩陣元素之排列順序為 A(1,1),A(2,1),A(3,1),...,A(NA,1), A(1,2),A(2,2),A(3,2),...,A(NA,2), A(1,3),A(2,3),A(3,3),...,A(NA,3), ...。

[表三(a)]中計算向量指標時，須經二次加算及一次乘算，效率較差。實用程式常改以加算方式計算如[表三(b)]副程式所示。此方式雖增加了計算效率，但卻使程式設計及覆閱較為困難。以下則介紹本書所用之處理方法。

[表三(a)] 副程式CMPY之3

```
      SUBROUTINE CMPY(A,B,C,N,M,L,NA,NB,NC)
      DIMENSION A(1),B(1),C(1)
      DO 10 J=1,L
      DO 10 I=1,N
      IJ=I+(J-1)*NC
      C(IJ)=0.0
      DO 10 K=1,M
      IK=I+(K-1)*NA
      KJ=K+(J-1)*NB
   10 C(IJ)=C(IJ)+A(IK)*B(KJ)
      RETURN
      END
```

[表三(b)] 副程式CMPY之4

```
      SUBROUTINE CMPY(A,B,C,N,M,L,NA,NB,NC)
      DIMENSION A(1),B(1),C(1)
      JB=0
      JC=0
      DO 30 J=1,L
      IJ=JC
      DO 10 I=1,N
      IJ=IJ+1
      IK=I
      KJ=JB
      C(IJ)=0.0
      DO 10 K=1,M
      KJ=KJ+1
      C(IJ)=C(IJ)+A(IK)*B(KJ)
   10 IK=IK+NA
      JB=JB+NB
   30 JC=JC+NC
      RETURN
      END
```

[表四(a)]副程式與[表二]或[表三(a)(b)]副程式作相同之運算，注意其中之宣告為DIMENSION A(1,1),B(1,1),C(1,1)，乍看似乎不合理，事實上其道理與[表三]中之宣告為DIMENSION A(1),B(1),C(1)相似，但減少了程式設計及覆閱之困難。注意程式中各矩陣之列指標與[表二]同，未予改變，僅將行指標加以修改，使其能正確算得儲存位置。如[表二]中之C(I,J)改為C(I,JC)，其中JC=(J–1)*NC+1。因[表二]中矩陣[C]每行有NC列，按行序排列，故元素C(I,J)排列於第(J–1)*NC+I位置。但[表四(a)]程式中[C]之宣告階數為1列1行，元素C(I,JC)按程式推算應排於第(JC–1)*1+I=(J–1)*NC+I位置，即與[表二]中元素之向量位置指標相符。同理，[表二]之A(I,K)與B(K,J)則分別改為A(I,KA)與B(K,JB)，其中KA=(K–1)*NA+1，JB=(J–1)*NB+1，其向量位置分別為(KA–1)*1+I=(K–1)*NA+I，(JB–1)*1+K=(J–1)*NB+K。

與[表四(a)]類似之方式為[表四(b)]之做法。這二種方式(即下表之方式(3)(4))之特點為列與行之指標分開計算，以使程式較清楚且較容易設計或維護。JA之計算可緊接在J值改變之處，I值改變時不影響JA之值。[表三(a)]副程式以方式(2)計算位置指標，則只要I或J之一改變即須重算，效率可能較差且程式較不清楚。

宣告指令	引用元素	向量位置算法	各指標之值	方式
$A(NA,M)$	$A(I,J)$	$(J-1)*NA+I$		(1)
$A(1)$	$A(IJ)$	IJ	$IJ=(J-1)*NA+I$	(2)
$A(1,1)$	$A(I,JA)$	$(JA-1)*1+I$	$JA=(J-1)*NA+1$	(3)
$A(1)$	$A(I+JA)$	$I+JA$	$JA=(J-1)*NA$	(4)

[表四(a)] 副程式CMPY之5

```
      SUBROUTINE CMPY(A,B,C,N,M,L,NA,NB,NC)
      DIMENSION A(1,1),B(1,1),C(1,1)
      DO 10 J=1,L
      JB=(J-1)*NB+1
      JC=(J-1)*NC+1
      DO 10 I=1,N
      C(I,JC)=0.0          ! add line  KA=1-NA
      DO 10 K=1,M
      KA=(K-1)*NA+1        ! change to KA=KA+NA
   10 C(I,JC)=C(I,JC)+A(I,KA)*B(K,JB)
      RETURN
      END
```

[表四(b)] 副程式CMPY之6

```
      SUBROUTINE CMPY(A,B,C,N,M,L,NA,NB,NC)
      DIMENSION A(1),B(1),C(1)
      DO 10 J=1,L
      JB=(J-1)*NB
      JC=(J-1)*NC
      DO 10 I=1,N
      C(I+JC)=0.0          ! add line  KA=-NA
      DO 10 K=1,M
      KA=(K-1)*NA          ! change to KA=KA+NA
   10 C(I+JC)=C(I+JC)+A(I+KA)*B(K+JB)
      RETURN
      END
```

為了效率之改進，亦可將[表四(a)(b)]之部分指令改如!後所示者。JB與JC因不在最內層迴圈計算，對效率影響不大，故可不必更改。至於I+JC之計算，因I與JC在K迴圈內之值不變，如利用編譯程式(Compiler)之最佳化功能(Optimize)，即不會於每次K改變時重覆計算。雖然I+KA亦可進一步改為IK如[表三(b)]者，但此舉又使向量位置與行列指標之關係不明顯，故不建議採用。

(4)動態矩陣分配法

一般FORTRAN皆規定程式中非虛擬參數之矩陣宣告階數須為固定數(Constant)。前項所述之宣告階數為變數之矩陣必須為虛擬參數，因虛擬矩陣之宣告並不預留矩陣所需之儲存位置，其所需位置必須由呼叫程式提供。而非虛擬矩陣之宣告會照其階數保留所需之儲存位置。但事實上大部分矩陣階數須隨分析問題之大小而改變，故須於使用程式前先檢查各矩陣宣告階數是否夠大。此項工作十分費事，偶一不慎即會錯誤，且有時不容易發現有錯。為克服此困難與避免錯誤，最好利用動態矩陣分配法分配矩陣所需之儲存位置。

茲舉一例以示較常用亦簡單可行的一種動態矩陣分配法，假若副程式SUB1需要二矩陣S(NS,NS),L(NS,NL)及一向量P(NS)，而副程式SUB2需要矩陣S(NS,NS)及Q(NS,2)，呼叫程式可按下列方式由陣列A中分配適當之位置給S,L,P,Q。若L及P僅在SUB1中使用，以後程式不再使用，則Q可以佔用L之位置，此時僅須增加'C'卡之指令，將矩陣Q(NS,2)之起始位置N4改至矩陣L(NS,NL)之起始位置N2即可。

[表五(a)] 動態矩陣分配示例

```
*************************************************************************
      DIMENSION A(2000),M(2000)
      EQUIVALENCE (A,M)
      DATA NDIM/2000/,N1/1/
      ...
      READ(*,'(10I5)') NS,NL,...     ! Read (or calculate) : NS,NL
      N2=N1+NS*NS                    ! A(N1) for matrix S(NS,NS)
      N3=N2+NS*NL                    ! M(N2) for matrix L(NS,NL)
      N4=N3+NS                       ! A(N3) for vector P(NS)
      IF(N4.GT.NDIM+1) STOP 'Dimension too small'
      CALL SUB1(A(N1),M(N2),A(N3),NS,NL,...)
      ...
C     N4=N2          (Reset N4=N2 to re-use the location used by L & P)
      N5=N4+NS*2                     ! A(N4) for matrix Q(NS,2)
      IF(N5.GT.NDIM+1) STOP 'Dimension too small'
      CALL SUB2(A(N1),A(N4),NS,...)
      ...
      END
*************************************************************************
      SUBROUTINE SUB1(S,L,P,NS,NL,...)      SUBROUTINE SUB2(S,Q,NS,...)
      DIMENSION S(NS,NS),L(NS,NL),P(NS)     DIMENSION S(NS,NS),Q(NS,2)
      ...                                   ...
      END                                   END
*************************************************************************
```

若使用FORTRAN 90，則可利用其動態記憶分配函數，以取得所需大小之儲存記憶位置，如[表五(b)]所示。動態記憶分配函數亦可直接用在

副程式中之臨時運算用之陣列。例如SUB1中，若L與P為臨時運算用之陣列，可按[表五(c)]所示二種方式之一修改。

[表五(b)] 動態矩陣分配用動態記憶分配函數

```
*************************************************************************
      REAL,    ALLOCATABLE:: S(:,:),P(:),Q(:,:)
      INTEGER, ALLOCATABLE:: L(:,:)       ! For FORTRAN-90
C**   REAL     S[ALLOCATABLE](:,:),P[ALLOCATABLE](:),Q[ALLOCATABLE](:,:)
C**   INTEGER  L[ALLOCATABLE](:,:)        ! For MS-FORTRAN
      ...
      READ(*,'(10I5)') NS,NL,...          ! Read (or calculate) : NS,NL
      ALLOCATE (S(NS,NS),L(NS,NL),P(NS), STAT=IERROR)
C**   ALLOCATE (S(1:NS,1:NS),L(1:NS,1:NL),P(1:NS)) ! can be any ranges
      IF(IERROR.NE.0) STOP 'Array too large'
      CALL SUB1(S,L,P,NS,NL,...)
      ...
C     DEALLOCATE (L,P, STAT=IERROR)       ! Free spaces used by L & P
      ALLOCATE (Q(NS,2), STAT=IERROR)
      IF(IERROR.NE.0) STOP 'Array too large'
      CALL SUB2(S,Q,NS,...)
      ...
      IF(ALLOCATED(L)) DEALLOCATE (L,P, STAT=IERROR)
      DEALLOCATE (S,Q, STAT=IERROR)       ! Free spaces used by S & Q
      END
*************************************************************************
```

[表五(c)] 臨時運算用陣列之動態記憶分配

```
*************************************************************************
      SUBROUTINE SUB1(S,NS,NL,...)     SUBROUTINE SUB1(S,Ldy,Pdy,NS,NL,...)
      DIMENSION S(NS,NS)               DIMENSION S(NS,NS)
      ALLOCATABLE L(:,:),P(:)          ALLOCATABLE :: L(:,:),P(:)
C**   DIMENSION L [ALLOCATABLE](:,:), P [ALLOCATABLE](:)
      ALLOCATE (L(NS,NL),P(NS))        ALLOCATE(L(NS,NL),P(NS),STAT=IERROR)
      ...                              ...
      DEALLOCATE (L,P)                 DEALLOCATE (L,P, STAT=IERROR)
      RETURN                           RETURN
      END                              END
*************************************************************************
```

5.3 帶狀矩陣之儲存法

圖一(a)示一8x8之帶狀矩陣，其下半帶寬MA=1，上半帶寬MB=2 (均不計對角線)。在有些應用(如[A]矩陣為聯立方程式之係數矩陣)上，可不必儲存帶寬以外之零元素，矩陣為對稱時，亦可僅存上三角或下三角部分之非零元素，如此即可節省大量之儲存位置。一般方法係將對角元素調整後排在同一行或同一列如圖一(b)與(c)所示。

$(a)\ DIMENSION\ \ A(8,8)$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & a_{46} & 0 & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & a_{57} & 0 \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} & a_{67} & a_{68} \\ 0 & 0 & 0 & 0 & 0 & a_{76} & a_{77} & a_{78} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{87} & a_{88} \end{bmatrix}$$

$(b)\ DIMENSION\ \ A(8,4)$

$$\begin{bmatrix} . & a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{32} & a_{33} & a_{34} & a_{35} \\ a_{43} & a_{44} & a_{45} & a_{46} \\ a_{54} & a_{55} & a_{56} & a_{57} \\ a_{65} & a_{66} & a_{67} & a_{68} \\ a_{76} & a_{77} & a_{78} & . \\ a_{87} & a_{88} & . & . \end{bmatrix}$$

$(c)\ DIMENSION\ \ A(4,8)$

$$\begin{bmatrix} . & . & a_{13} & a_{24} & a_{35} & a_{46} & a_{57} & a_{68} \\ . & a_{22} & a_{23} & a_{34} & a_{45} & a_{56} & a_{67} & a_{78} \\ a_{11} & a_{32} & a_{33} & a_{44} & a_{55} & a_{66} & a_{77} & a_{88} \\ a_{21} & a_{42} & a_{43} & a_{54} & a_{65} & a_{76} & a_{87} & . \end{bmatrix}$$

圖一

圖一(b)中元素a_{ij}係存於宣告階數為8x4之矩陣[A]之A(i, MA+j–i+1)中，程式設計時須將行指標j換算為位置指標MA+j–i+1，增加程式設計之複雜度。同理圖一(c)中元素a_{ij}係存於矩陣[A]之A(MB+i–j+1, j)中。

以下介紹本書程式之另一處理方法。該法係將矩陣(或對稱矩陣之上或下三角部分)之非零元素"直接"存在一個宣告階數為DIMENSION A(3,10)之矩陣中，此矩陣僅可儲存3x10個非零元素。

圖一 $(d)\ DIMENSION\ \ A(3,10)$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{43} & a_{44} & a_{45} & a_{46} & a_{76} & a_{77} & a_{78} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{54} & a_{55} & a_{56} & a_{57} & a_{87} & a_{88} \\ . & a_{32} & a_{33} & a_{34} & a_{35} & a_{65} & a_{66} & a_{67} & a_{68} & . \end{bmatrix}$$

其中元素a_{43}應位於第3行之第4列(或稱第4個位置)，但因矩陣[A]每行只留3列(3個位置)，故從第3行第1列算起之第4個位置應該在第4行之第1位置，即A(1,4)之位置，但a_{14}在上半帶寬之範圍以外不必儲存，正好留給a_{43}使用。

同理，a_{44}原為第4行第4個位置，現須在第5行第1個位置，而a_{54}則須在第5行第2個位置，最後元素a_{88}應從第8行第1列算起的第8個位置，即A(2,10)之位置，但$a_{2,10}$不在矩陣8x8之範圍仍可被a_{88}使用。

依此方式，經仔細推算，可知矩陣[A]之帶寬內所有非零元素，於DIMENSION A(3,10)中皆有一唯一位置儲存之。值得注意的是各元素仍可用其原來之"行列指標"直接引用之(即A(4,6)代表a_{46})不須經過指標換算。因對宣告階數為A(3,10)之矩陣而言，A(2,5)=A(5,4)=A(8,3)，因其儲存位置均相同為14=(5–1)*3+2=(4–1)*3+5=(3–1)*3+8。即A(2,5),A(5,4)及A(8,3)等所表示之儲存位置皆相同，而各元素僅A(5,4)須要儲存。由於指標不必換算，可使運算程式之設計與一般儲存全部矩陣元素之方式相同。

5.4 變寬帶矩陣之儲存法

下表為一對稱矩陣，其非零元素大都集中在對角線附近，但其半帶寬並不相等。此種矩陣之儲存，原則上可僅儲存其上三角部分或下三角部分之非零元素。但此等矩陣於實際問題中常為線性聯立方程式之係數矩陣，目的在用以求解方程式未知數。

$$
\begin{array}{rc}
J = & \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\
LL(J) = & \begin{array}{ccccccccc} 1 & 2 & 4 & 6 & 9 & 10 & 14 & 17 & 22 \end{array} \\
[A] = & \begin{bmatrix}
a_{11} & a_{12} & a_{13} & & & & & & \\
* & a_{22} & a_{23} & a_{24} & a_{25} & & & & \\
* & * & a_{33} & a_{34} & a_{35} & & a_{37} & & \\
 & * & * & a_{44} & a_{45} & & 0 & & a_{49} \\
 & * & * & * & a_{55} & a_{56} & a_{57} & a_{58} & 0 \\
 & & & & * & a_{66} & a_{67} & a_{68} & 0 \\
 & & * & & * & * & a_{77} & a_{78} & a_{79} \\
 & & & & * & * & * & a_{88} & a_{89} \\
 & & & * & & & * & * & a_{99}
\end{bmatrix}
\end{array}
$$

在第三章討論過，如將上列對稱矩陣分解為$[A] = \bar{L}\bar{U}$，其中$\bar{L} = \bar{U}^T$，或分解為$[A] = U^T D U$時，$\bar{U}$或U均為上三角矩陣，其每行上半帶寬與原係數矩陣之上半帶寬相同(注意u_{47}, u_{59}, u_{69}均可能不等於零)。故只要儲存

每行之上半帶寬以內之元素即可。因此，元素可以濃縮排列如下表所示。

$A(K)$	a_{11}	a_{12}	a_{22}	a_{13}	a_{23}	a_{33}	a_{24}	a_{34}	a_{44}	a_{25}	a_{35}	a_{45}	a_{55}	a_{56}	a_{66}
K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$A(K)$	a_{37}	0	a_{57}	a_{67}	a_{77}	a_{58}	a_{68}	a_{78}	a_{88}	a_{49}	0	0	a_{79}	a_{89}	a_{99}
K	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

為了便予由元素之行列指標獲得該元素之正確儲存位置，每行須設一位置指標 LL(J), J=1,2,...,N ， N 為係數矩陣[A]之行數。按第5.2節之原理，若矩陣元素欲以雙指標表示，其宣告為DIMENSION A(1,1) ，而元素 a_{ij} 欲以A(i,LL(j))表示，則上述變寬帶矩陣之位置指標LL(1)至LL(9)應為1,2,4,6,9,10,14,17,22 。第一行之位置指標等於1 ，其餘各行之位置指標為該行之上半帶寬加前一行之位置指標。例如：
第2行之上半帶寬等於1 ，加第1行之位置指標LL(1)=1 ，得LL(2)=2 。
第3行之上半帶寬等於2 ，加第2行之位置指標LL(2)=2 ，得LL(3)=4 。
第4行之上半帶寬等於2 ，加第3行之位置指標LL(3)=4 ，得LL(4)=6 。
餘類推。根據最後一行(第N行)之位置指標LL(N)可算得全部儲存位置LL(N)+N–1。如此則第I列第J行之元素可以A(I,JA)表示，其中JA=LL(J)。例如，矩陣元素 a_{24} ，因LL(4)=6 ，故以A(2,6)表示。而程式因[A]之宣告為DIMENSION A(1,1) ，故A(2,6)之儲存位置為(6–1)*1+2=6–1+2=7 ，與上表所示者相同。

如果程式之宣告改為DIMENSION A(1) ，亦即以向量表示，則第I列第J行之元素改用A(I+JA)表示，其中JA=LL(J)–LL(1)。在此情況下，行位置指標亦可照前述之值減1 ，即LL(1:9)={0,1,3,5,8,9,13,16,21} ，則JA=LL(J)。但本書程式仍以JA=LL(J)–LL(1)計算，故二種LL(*)值均可適用。

如果矩陣不對稱而須儲存全部帶寬內之非零元素，請參考前一章中之說明。該章對本節所述之處理方法亦做介紹。

5.5 資料長度不等之處理

前節所述儲存變寬帶矩陣之處理方法可以應用在儲存每組有不同長度之資料之情形。例如，在有限元素分析中，不同元素之結點數目不一定會相同：如三角形線性元素有三個結點；三角形二次元素有六個結點；

四邊形線性元素有四個結點；四邊形二次元素有八或九個結點等。如以二維陣列儲存這些結點號碼：例如，設第J元素有4個結點號碼將其存於NODE(1:4,J)，則陣列之列數須為元素之可能最多結點數。對於結點較少之元素即有多餘儲址閒置不用而造成浪費。以下介紹一種不浪費儲存位置之簡易辦法。

用下表所示各元素之結點號碼為例：設有7個元素，各元素之結點數之和為28，則可宣告一個長度至少為28之陣列NODE(1,28)，用以依序存放各元素之結點號碼；再宣告一個長度至少比元素數多1之一維陣列LL(8)，用以存放各元素結點資料之位置指標，如下表所示。則第J元素之4個結點號碼即儲存在NODE(1:4,LL(J))。表中位置指標數比元素數多1可便予計算元素之結點數，如第J元素之結點數為LL(J+1)–LL(J)。

元素號碼	元素之結點號碼	結點數	LL(N)	
1	4 8 5	3	1	
2	1 4 5 2	4	4	=3+1
3	5 8 12	3	8	=4+4
4	2 5 6 3	4	11	=3+8
5	4 10 12 11 8 7	6	15	=4+11
6	5 12 13 6	4	21	=6+15
7	12 14 15 13	4	25	=4+21
8			29	=4+25

$NODE(1,K)$	4	8	5	1	4	5	2	5	8	12	2	5	6	3
K	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$NODE(1,K)$	4	10	12	11	8	7	5	12	13	6	12	14	15	13
K	15	16	17	18	19	20	21	22	23	24	25	26	27	28

5.6 副程式$CUDECP$用於濃縮帶狀矩陣之用例

[表六]主程式主要用以顯示帶狀矩陣之三種儲存方式在計算機內之實際安排情形。陣列AGEN儲存未經濃縮之帶狀矩陣。陣列APCN儲存經部分濃縮之帶狀矩陣。陣列AFCN儲存經完全濃縮之帶狀矩陣。儲存完全濃縮帶狀矩陣之陣列所需之宣告列數MM，等於帶狀矩陣之上半帶寬MB與

下半帶寬MA之和。所需宣告行數NN ，等於原矩陣總行數N加上(原矩陣總列數M–1)/宣告列數，亦即： MM=MA+MB ， NN=N+(M–1)/MM 。

注意三種儲存方式之指標對同一元素均相同，即其指標不必因濃縮儲存而修改，且在呼叫CUDECP及CUSOLX時，僅需告以不同之宣告列數，其餘參數完全相同。

［表六］ 濃縮帶狀矩陣用例

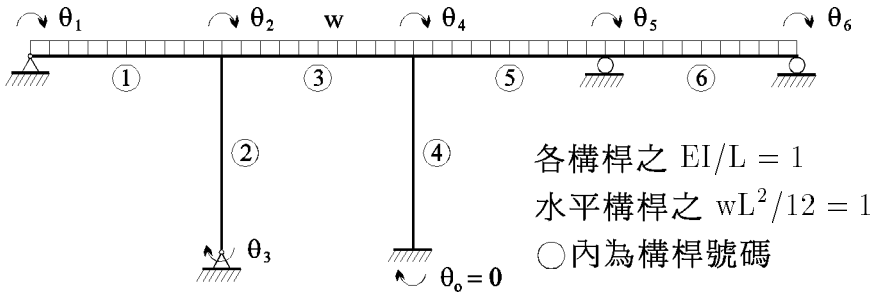
```
************************************************************************
      PROGRAM CUDMAN
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AGEN(8,8),BG(8),APCD(4,9),BP(8),AFCD(3,10),BF(8)
      DATA MAG/8/,MAP/4/,MAF/3/,APCD/36*0.0/,AFCD/30*0.0/
C**   ============================================================    **
C**   Program for decomposition of an unsymmetric banded matrix       **
C**           and solve the equation by substitution                  **
C**   Using general, partial condensed and full condensed matrices    **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Read :  Order N, Lower bandwidth MA, upper bandwidth MB,  |    **
C**   |         Matrix & constant vector                         |    **
C**   | Compute NAP,NAF = # of columns reqd for condensed matrix |    **
C**   +----------------------------------------------------------+    **
      READ(*,'(3I4)') N,MA,MB
      NAP=N+(N-1)/MAP
      NAF=N+(N-1)/MAF
      DO 20 I=1,N
   20 READ(*,'(10F8.0)') (AGEN(I,J),J=1,N)
      READ(*,'(10F8.0)') (BG(J),J=1,N)
C**   +----------------------------------------------------------+    **
C**   | Setup Partial & Full condensed matrices                  |    **
C**   | Note that : the subscripts are the same                  |    **
C**   +----------------------------------------------------------+    **
      DO 40 J=1,N
      DO 30 I=MAX0(J-MB,1),MIN0(J+MA,N)
      APCD(I,J)=AGEN(I,J)
   30 AFCD(I,J)=AGEN(I,J)
      BP(J)=BG(J)
   40 BF(J)=BG(J)
C**   +----------------------------------------------------------+    **
C**   | Write the given matrix & constant vector                 |    **
C**   +----------------------------------------------------------+    **
      CALL PRINTA(AGEN,N,N,MAG,'Given matrix in general form')
      CALL PRINTA(APCD,MAP,NAP,MAP,'In partial condensed form')
      CALL PRINTA(AFCD,MAF,NAF,MAF,'In full condensed form')
      WRITE(*,'(/'' Constant vector''//(5X,10F6.2))') (BG(J),J=1,N)
C**   +----------------------------------------------------------+    **
C**   | Perform Decomposition & substitutions                    |    **
C**   +----------------------------------------------------------+    **
      CALL CUDECP(AGEN,N,MA,MB,MAG)
      CALL CUSOLX(AGEN,BG,N,MA,MB,MAG)
C**   ------------------------------------------------------------    **
      CALL CUDECP(APCD,N,MA,MB,MAP)
      CALL CUSOLX(APCD,BP,N,MA,MB,MAP)
C**   ------------------------------------------------------------    **
```

```
      CALL CUDECP(AFCD,N,MA,MB,MAF)
      CALL CUSOLX(AFCD,BF,N,MA,MB,MAF)
C**   +-------------------------------------------------------------+    **
C**   | Write the decomposed matrix & result vector                 |    **
C**   +-------------------------------------------------------------+    **
      CALL PRINTA(AGEN,N,N,MAG,'Decomposed matrix in general form')
      WRITE(*,'(/'' Result vector''//(5X,10F6.2))') (BG(J),J=1,N)
      CALL PRINTA(APCD,MAP,NAP,MAP,'In partial condensed form')
      WRITE(*,'(/'' Result vector''//(5X,10F6.2))') (BP(J),J=1,N)
      CALL PRINTA(AFCD,MAF,NAF,MAF,'In full condensed form')
      WRITE(*,'(/'' Result vector''//(5X,10F6.2))') (BF(J),J=1,N)
      STOP
      END
*****************************************************************************
      SUBROUTINE PRINTA(A,M,N,MM,NOTE)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(MM,N)
      CHARACTER*(*) NOTE
      DATA NN/10/
C**   ============================================================    **
C**   Print matrix A(M,N)                                             **
C**   ============================================================    **
      WRITE(*,'(/1X,A)') NOTE
      DO 20 JL=1,N,NN
      WRITE(*,'(/5X,10I6)') (J,J=JL,MIN0(JL+NN-1,N))
      DO 10 I=1,M
      WRITE(*,'(I5,10F6.2)') I,(A(I,J),J=JL,MIN0(JL+NN-1,N))
   10 CONTINUE
   20 CONTINUE
      RETURN
      END
*****************************************************************************
   8   1   2
    2.00    1.00    2.00       0       0       0       0       0
    1.00    2.00    1.30    0.60       0       0       0       0
       0    2.00    3.40    2.60    0.90       0       0       0
       0       0    1.50    3.30    0.93    1.20       0       0
       0       0       0    0.80    1.36    0.88    0.36       0
       0       0       0       0    0.30    0.62    0.34    0.10
       0       0       0       0       0    0.40    2.50    1.46
       0       0       0       0       0       0    1.10    3.86
    1.10    0.94    4.66    6.51    0.94    0.84   -1.29    2.21

=============================================================================
Given matrix in general form

         1     2     3     4     5     6     7     8
   1  2.00  1.00  2.00   .00   .00   .00   .00   .00
   2  1.00  2.00  1.30   .60   .00   .00   .00   .00
   3   .00  2.00  3.40  2.60   .90   .00   .00   .00
   4   .00   .00  1.50  3.30   .93  1.20   .00   .00
   5   .00   .00   .00   .80  1.36   .88   .36   .00
   6   .00   .00   .00   .00   .30   .62   .34   .10
   7   .00   .00   .00   .00   .00   .40  2.50  1.46
   8   .00   .00   .00   .00   .00   .00  1.10  3.86

In partial condensed form

         1     2     3     4     5     6     7     8     9
```

```
   1  2.00  1.00  2.00   .00   .80  1.36   .88   .36   .00
   2  1.00  2.00  1.30   .60   .00   .30   .62   .34   .10
   3   .00  2.00  3.40  2.60   .90   .00   .40  2.50  1.46
   4   .00   .00  1.50  3.30   .93  1.20   .00  1.10  3.86

In full condensed form

         1     2     3     4     5     6     7     8     9    10
   1  2.00  1.00  2.00  1.50  3.30   .93  1.20   .40  2.50  1.46
   2  1.00  2.00  1.30   .60   .80  1.36   .88   .36  1.10  3.86
   3   .00  2.00  3.40  2.60   .90   .30   .62   .34   .10   .00

Constant vector

      1.10   .94  4.66  6.51   .94   .84 -1.29  2.21

Decomposed matrix in general form

         1     2     3     4     5     6     7     8
   1  2.00   .50  1.00   .00   .00   .00   .00   .00
   2  1.00  1.50   .20   .40   .00   .00   .00   .00
   3   .00  2.00  3.00   .60   .30   .00   .00   .00
   4   .00   .00  1.50  2.40   .20   .50   .00   .00
   5   .00   .00   .00   .80  1.20   .40   .30   .00
   6   .00   .00   .00   .00   .30   .50   .50   .20
   7   .00   .00   .00   .00   .00   .40  2.30   .60
   8   .00   .00   .00   .00   .00   .00  1.10  3.20

Result vector

      -.50  -.30  1.20   .80 -1.00  2.50 -1.50  1.00

In partial condensed form

         1     2     3     4     5     6     7     8     9
   1  2.00   .50  1.00   .00   .80  1.20   .40   .30   .00
   2  1.00  1.50   .20   .40   .00   .30   .50   .50   .20
   3   .00  2.00  3.00   .60   .30   .00   .40  2.30   .60
   4   .00   .00  1.50  2.40   .20   .50   .00  1.10  3.20

Result vector

      -.50  -.30  1.20   .80 -1.00  2.50 -1.50  1.00

In full condensed form

         1     2     3     4     5     6     7     8     9    10
   1  2.00   .50  1.00  1.50  2.40   .20   .50   .40  2.30   .60
   2  1.00  1.50   .20   .40   .80  1.20   .40   .30  1.10  3.20
   3   .00  2.00  3.00   .60   .30   .30   .50   .50   .20   .00

Result vector

      -.50  -.30  1.20   .80 -1.00  2.50 -1.50  1.00
==========================================================================
```

5.7 結構分析應用－勁度矩陣以帶狀矩陣儲存

在上一節之主程式裡，係首先讀入一個未經濃縮之帶狀矩陣AGEN，然後再將其濃縮儲存至一個較小之濃縮矩陣APCN及AFCD中。這其間似乎有點多此一舉，而且，計算機之容量既然能容納整個矩陣AGEN，又何必再將其擠進一個小矩陣AFCD中。其實，該主程式目的僅在於顯示濃縮儲存之具體型式及強調行列指標不必因濃縮而改變之特性。在實際應用時，常需先根據問題之大小算出矩陣之階數及帶寬，再利用動態矩陣分配法分配一大小適當之儲址以備存放該濃縮矩陣。然後將該矩陣直接建立在此一預留之濃縮矩陣內，而不是建立於一未濃縮之大矩陣內再搬至濃縮之小矩陣中。以下將以一個簡化的結構分析程式為例，舉出濃縮矩陣之實際用法。



圖二 範例結構

例中之結構分析程式採用勁度分析法，能分析各種幾何形狀之平面結構，但每個結點最多只允許有一個自由度。茲以圖二之結構為例，該結構共有7個結點，其中除一個結點為固端外，其餘各結點均有一轉角自由度，因此共有6個轉角自由度，亦即有六個未知轉角$\theta_1, \theta_2, \ldots, \theta_6$。利用6個結點之彎矩平衡關係，可得下列方程式：

$$\begin{bmatrix} 4 & 2 & 0 & 0 & 0 & 0 \\ 2 & 12 & 2 & 2 & 0 & 0 \\ 0 & 2 & 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 12 & 2 & 0 \\ 0 & 0 & 0 & 2 & 8 & 2 \\ 0 & 0 & 0 & 0 & 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} - \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{5.1}$$

對每一構桿而言，各構桿之桿端彎矩與桿端轉角有如下關係：

$$\text{構桿}1：\quad \begin{Bmatrix} M_1^1 \\ M_2^1 \end{Bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \end{Bmatrix} + \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \tag{5.2}$$

$$\text{構桿}2：\quad \begin{Bmatrix} M_2^2 \\ M_3^2 \end{Bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_2 \\ \theta_3 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{5.3}$$

$$\text{構桿}3：\quad \begin{Bmatrix} M_2^3 \\ M_4^3 \end{Bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_2 \\ \theta_4 \end{Bmatrix} + \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \tag{5.4}$$

$$\text{構桿}4：\quad \begin{Bmatrix} M_o^4 \\ M_4^4 \end{Bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_o \\ \theta_4 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{5.5}$$

$$\text{構桿}5：\quad \begin{Bmatrix} M_4^5 \\ M_5^5 \end{Bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_4 \\ \theta_5 \end{Bmatrix} + \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \tag{5.6}$$

$$\text{構桿}6：\quad \begin{Bmatrix} M_5^6 \\ M_6^6 \end{Bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_5 \\ \theta_6 \end{Bmatrix} + \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \tag{5.7}$$

以上6個方程式可擴充至與式(5.1)一樣大小之階數，如式(5.8)至式(5.13)所示，請注意構桿4中，$\theta_o = 0$，在式(5.1)中之矩陣並無對應於$\mathrm{M_o}$之列，故予去掉。

$$\text{構桿}1：\quad \begin{Bmatrix} M_1^1 \\ M_2^1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{bmatrix} 4 & 2 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} + \begin{Bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{5.8}$$

$$\text{構桿}2：\quad \begin{Bmatrix} 0 \\ M_2^2 \\ M_3^2 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{5.9}$$

構桿3 ：
$$\begin{Bmatrix} 0 \\ M_2^3 \\ 0 \\ M_4^3 \\ 0 \\ 0 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} + \begin{Bmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \end{Bmatrix} \tag{5.10}$$

構桿4 ：
$$\begin{Bmatrix} 0 \\ 0 \\ 0 \\ M_4^4 \\ 0 \\ 0 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{5.11}$$

構桿5 ：
$$\begin{Bmatrix} 0 \\ 0 \\ 0 \\ M_4^5 \\ M_5^5 \\ 0 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 & 0 \\ 0 & 0 & 0 & 2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \end{Bmatrix} \tag{5.12}$$

構桿6 ：
$$\begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ M_5^6 \\ M_6^6 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 2 & 4 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \end{Bmatrix} \tag{5.13}$$

將以上六式相加，並引用結點$i = 1, 2, \ldots, 6$之彎矩平衡條件$\sum_{k=1}^{6} M_i^k = 0$，即可得式(5.1)之平衡方程式。此處有下列數點應予注意：

(1)式(5.1)中結構勁度矩陣之帶寬等於以上六式中各構桿勁度矩陣之帶寬之最大者。構桿勁度矩陣之半帶寬MD等於與該構桿有關之未知轉角編號之最大與最小者之差值。例如構桿3，有關之未知轉角θ_2與θ_4之編號為2與4，其最大與最小者之差為4–2=2，故MD=2。同法可得

構桿1,2,5,6之構桿勁度矩陣之MD=1，而構桿4者為零。因此，結構勁度矩陣之半帶寬為2。副程式CBAND即係用以計算各構桿勁度矩陣之MD，並比較保留其最大者於MM，最後所得者即為結構勁度矩陣之半帶寬。主程式再根據算得之半帶寬預留適當的儲址(MM x NN)，以備在副程式FORMEQ中建立結構勁度矩陣及荷重向量之用。

(2) 在將各構桿勁度矩陣及固端力素合併(即擴大後加入)至結構勁度矩陣及荷重向量時，不必先將式(5.2)之矩陣擴大改存於如式(5.8)之矩陣再加到式(5.1)之矩陣，而可直接由式(5.2)之矩陣，根據其所須對應於式(5.1)矩陣之行列，直接加到式(5.1)矩陣之對應行列之元素。詳見副程式FORMEQ。其中KK(I,K)即用以儲存對應行列之號碼。

(3) 程式之輸入資料為：

NE	=	構桿總數。
N	=	結構自由度數(即未知轉角數)。
$KK(I,K)$	=	構桿K之勁度矩陣[SM](如式(5.2))之第I行(或列)所對應之結構勁度矩陣[S]之行(或列)。
$SM(2,2,K)$	=	構桿K之勁度矩陣，如式(5.2)中之方陣。
$PM(2,K)$	=	構桿K之固端力素，如式(5.2)中之行向量。

[表六]程式之後所列者為圖二結構之輸入資料與分析結果。其中結構勁度矩陣經分解為：

$$\begin{bmatrix} 4 & 2 & 0 & 0 & 0 & 0 \\ 2 & 12 & 2 & 2 & 0 & 0 \\ 0 & 2 & 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 12 & 2 & 0 \\ 0 & 0 & 0 & 2 & 8 & 2 \\ 0 & 0 & 0 & 0 & 2 & 4 \end{bmatrix} =$$

$$\begin{bmatrix} 2 & & & & & \\ 1 & 3.32 & & & & \\ 0 & .603 & 1.91 & & & \\ 0 & .603 & -.191 & 3.41 & & \\ 0 & 0 & 0 & .587 & 2.77 & \\ 0 & 0 & 0 & 0 & .723 & 1.86 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ & 3.32 & .603 & .603 & 0 & 0 \\ & & 1.91 & -.191 & 0 & 0 \\ & & & 3.41 & .587 & 0 \\ & & & & 2.77 & .723 \\ & & & & & 1.86 \end{bmatrix}$$

算得之轉角如下，最後並印出由式(5.2)算得之各桿端彎矩。

$$\begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} = \begin{Bmatrix} 0.2746 \\ -0.0492 \\ 0.0246 \\ -0.0039 \\ 0.0725 \\ -0.2863 \end{Bmatrix} \tag{5.14}$$

［表六］ 結構分析－帶狀矩陣儲存

```
************************************************************************
      PROGRAM STRACB
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(5000),K(2,5000)
      EQUIVALENCE (A,K)
      NDIM=5001
C**   ============================================================    **
C**   Program for structural analysis using constant banded matrix    **
C**   ============================================================    **
   10 READ(5,'(2I4)') NE,N
      IF(NE.EQ.0) STOP
C**   +----------------------------------------------------------+    **
C**   | Dynamic allocate the space for arrays                    |    **
C**   +----------------------------------------------------------+    **
      N1=1
      N2=N1+2*2*NE               ! SM(2,2,NE)
      N3=N2+2*NE                 ! PM(2,NE)
      N4=N3+2*NE                 ! KK(2,NE)
      N5=N4                      ! Dummy L(0) unused
      IF(N5.GT.NDIM) STOP
      CALL INPUT(A(N1),A(N2),K(1,N3),NE)
      CALL CBAND(K(1,N3),NE,N,MM,NN)
C**   +----------------------------------------------------------+    **
C**   | Allocate space for Constant banded S(MM,NN) & P(N)       |    **
C**   +----------------------------------------------------------+    **
      N6=N5+MM*NN
      N7=N6+N
      IF(N7.GT.NDIM) STOP
      CALL FORMEQ(A(N1),A(N2),K(1,N3),A(N5),A(N6),K(1,N4),NE,N,MM,NN)
      CALL PRINTV(A(N5),K(1,N4),N,MM,NN,'Stiffness con. banded matrix')
C**   ------------------------------------------------------------    **
      CALL CBDECP(A(N5),MM,N,MM)
      CALL CBSOLX(A(N5),A(N6),MM,N,MM)
C**   ------------------------------------------------------------    **
      CALL PRINTV(A(N5),K(1,N4),N,MM,NN,'Decomposed con. banded matrix')
      CALL OUTPUT(A(N1),A(N2),K(1,N3),A(N6),NE,N)
      GO TO 10
      END
************************************************************************
      SUBROUTINE CBAND(KK,NE,N,MM,NN)
C**   ============================================================    **
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION KK(2,NE)
C**   ============================================================    **
C**   Compute MM = Half-bandwidth of the stiffness matrix             **
C**           NN = N+(N-1)/MM = Number of columns for the matrix      **
C**   ============================================================    **
      MM=2           ! To be different to variable banded matrix
      DO 50 K=1,NE
C**   +------------------------------------------------------------+    **
C**   | MAX/MIN = Maximum/Minimum unknown # of member K           |    **
C**   | MD = MAX-MIN = Half-bandwidth of member stiffness matrix  |    **
C**   | MM = Maximum of MD(1:NE)                                 |    **
C**   +------------------------------------------------------------+    **
      MAX=0
      MIN=N
      DO 30 I=1,2
      J=KK(I,K)
      IF(J.EQ.0.OR.J.GT.N) GO TO 30
      IF(MAX.LT.J) MAX=J
      IF(MIN.GT.J) MIN=J
   30 CONTINUE
      MD=MAX-MIN
      IF(MD.GT.MM) MM=MD
   50 CONTINUE
C**   ------------------------------------------------------------    **
      NN=N+(N-1)/MM
      RETURN
      END
*************************************************************************
      SUBROUTINE INPUT(SM,PM,KK,NE)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION SM(2,2,NE),PM(2,NE),KK(2,NE)
C**   ============================================================    **
C**   READ KK(2,NE)   = Unknown # of member end displacements         **
C**        SM(2,2,NE) = Member stiffness matrix                       **
C**        PM(2,NE)   = Member fixed end forces                       **
C**   ============================================================    **
      DO 40 K=1,NE
      READ(*,'(2I4)')   (KK(I,K),I=1,2)
   40 READ(*,'(6F8.0)') ((SM(I,J,K),J=1,2),I=1,2),(PM(I,K),I=1,2)
      RETURN
      END
*************************************************************************
      SUBROUTINE FORMEQ(SM,PM,KK,S,P,L,NE,N,MM,NN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION SM(2,2,NE),PM(2,NE),KK(2,NE)
      DIMENSION S(MM,NN),P(N),L(N)
C**   ============================================================    **
C**   Form S(N,N) = Structural stiffness matrix                       **
C**    and  P(N)  = Loading vector for S(N,N)*D(N) = P(N)             **
C**   ------------------------------------------------------------    **
C**   For constant banded : MM = Half-bandwidth; NN = N+(N-1)/MM      **
C**   For variable banded : MM = 1;              NN = L(N)+N-1        **
C**   ============================================================    **
      DO 10 JJ=1,NN
      DO 10 II=1,MM
   10 S(II,JJ)=0.0
      DO 15 JJ=1,N
```

```
   15 P(JJ)=0.0
C**   ------------------------------------------------------------    **
      DO 30 K=1,NE
      DO 30 J=1,2
      JJ=KK(J,K)
      IF(JJ.EQ.0.OR.JJ.GT.N) GO TO 30
      IF(MM.NE.1) THEN
C**     +-----------------------------------------------------+    **
C**     | For constant banded matrix (L(N) is dummy and unused)  |    **
C**     +-----------------------------------------------------+    **
        JS=JJ
      ELSE
C**     +-----------------------------------------------------+    **
C**     | For variable banded matrix (L(N) is location index)    |    **
C**     +-----------------------------------------------------+    **
        JS=L(JJ)
      ENDIF
      DO 20 I=1,2
      II=KK(I,K)
      IF(II.EQ.0.OR.II.GT.JJ) GO TO 20
      S(II,JS)=S(II,JS)+SM(I,J,K)
   20 CONTINUE
      P(JJ)=P(JJ)-PM(J,K)
   30 CONTINUE
C**   ------------------------------------------------------------    **
      WRITE(*,'(/'' Loading vector of strucutre''//(1X,1P6E11.4))') P
      RETURN
      END
*************************************************************************
      SUBROUTINE PRINTV(S,L,N,MM,NN,NOTE)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION S(MM,NN),L(N)
      CHARACTER*(*) NOTE
C**   ============================================================    **
C**   Write S(MM,NN)                                                  **
C**   ============================================================    **
      WRITE(*,'(/1X,A/)') NOTE
      DO 40 I=1,MM
   40 WRITE(*,'(1X,1P8E9.2)') (S(I,J),J=1,NN)
      IF(MM.NE.1) THEN
C**     +-----------------------------------------------------+    **
C**     | For constant banded matrix (L(N) is dummy and unused)  |    **
C**     +-----------------------------------------------------+    **
        DO 50 J=1,N
        JM=MAX0(J-MM,1)
   50   WRITE(*,'(1X,1P8E9.2)') (0.0,I=1,JM-1),(S(I,J),I=JM,J)
      ELSE
C**     +-----------------------------------------------------+    **
C**     | For variable banded matrix (L(N) is location index)    |    **
C**     +-----------------------------------------------------+    **
        JM=1
        DO 60 J=1,N
        JS=L(J)
        IF(J.GT.1) JM=J-L(J)+L(J-1)
   60   WRITE(*,'(1X,1P8E9.2)') (0.0,I=1,JM-1),(S(I,JS),I=JM,J)
      ENDIF
      RETURN
      END
*************************************************************************
```

```
      SUBROUTINE OUTPUT(SM,PM,KK,D,NE,N)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION SM(2,2,NE),PM(2,NE),KK(2,NE),D(N),DM(2),AM(2)
C**   ============================================================    **
C**   Write D(N)  = Nodal displacements of the structure              **
C**         AM(2) = Member end forces = SM(2,2)*DM(2) + PM(2)         **
C**   where DM(i) = D(KK(i,K)), if KK(i,K)>0   (else DM(i)=0)         **
C**   ============================================================    **
      WRITE(*,'(/'' Displacements of strucutre''//(1X,1P6E11.4))') D
      WRITE(*,'(/1X,A)')  'Member end forces'
      WRITE(*,'(/1X,A/)') 'Member   Left-moment   Right-moment'
      DO 50 K=1,NE
      DO 30 J=1,2
      JJ=KK(J,K)
      DM(J)=0.0
      IF(JJ.NE.0.AND.JJ.LE.N) DM(J)=D(JJ)
   30 CONTINUE
      DO 40 I=1,2
      AM(I)=PM(I,K)
      DO 40 J=1,2
   40 AM(I)=AM(I)+SM(I,J,K)*DM(J)
   50 WRITE(*,'(I6,1P2E15.5)') K,(AM(I),I=1,2)
      RETURN
      END
*************************************************************************
   6   6
   1   2
     4.0     2.0     2.0     4.0     -1.     1.0
   3   2
     4.0     2.0     2.0     4.0     0.0     0.0
   2   4
     4.0     2.0     2.0     4.0     -1.     1.0
   0   4
     4.0     2.0     2.0     4.0     0.0     0.0
   4   5
     4.0     2.0     2.0     4.0     -1.     1.0
   5   6
     4.0     2.0     2.0     4.0     -1.     1.0

=========================================================================
Loading vector of strucutre

 1.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00-1.0000E+00

Stiffness con. banded matrix

 4.00E+00 2.00E+00 0.00E+00 4.00E+00 0.00E+00 0.00E+00 8.00E+00 2.00E+00
 0.00E+00 1.20E+01 2.00E+00 2.00E+00 1.20E+01 2.00E+00 0.00E+00 4.00E+00
 4.00E+00
 2.00E+00 1.20E+01
 0.00E+00 2.00E+00 4.00E+00
 0.00E+00 2.00E+00 0.00E+00 1.20E+01
 0.00E+00 0.00E+00 0.00E+00 2.00E+00 8.00E+00
 0.00E+00 0.00E+00 0.00E+00 0.00E+00 2.00E+00 4.00E+00

Decomposed con. banded matrix

 2.00E+00 1.00E+00 0.00E+00 1.91E+00-1.91E-01 0.00E+00 2.77E+00 7.23E-01
 0.00E+00 3.32E+00 6.03E-01 6.03E-01 3.41E+00 5.87E-01 0.00E+00 1.86E+00
```

```
 ...
Displacements of strucutre

 2.7461E-01-4.9223E-02 2.4611E-02-3.8860E-03 7.2539E-02-2.8627E-01

Member end forces

Member   Left-moment   Right-moment

    1   -9.71445E-17    1.35233E+00
    2   -1.38778E-17   -1.47668E-01
    3   -1.20466E+00    8.86010E-01
    4   -7.77202E-03   -1.55440E-02
    5   -8.70466E-01    1.28238E+00
    6   -1.28238E+00   -2.22045E-16
================================================================================
```

5.8 結構分析應用－勁度矩陣以變寬帶矩陣儲存

前節所述之結構分析程式可以改變其勁度矩陣之儲存方式，由帶狀矩陣改成變寬帶矩陣。修改後之程式與分析結果列於[表七]，但省略二者相同之部分。輸入資料與帶狀矩陣者相同。由於二者程式之差異不大，部分副程式與前者共用，但有少數指令不同，請自行比較其相異處。

前節程式中勁度矩陣按帶狀矩陣儲存，只須計算並保留半帶寬MM即可。本節程式中勁度矩陣按變寬帶矩陣儲存，故須保留每行之半帶寬或每行之位置指標L(J)。而且，同樣需要在建立勁度矩陣之前先予求出。在本章已說明過變寬帶矩陣之位置指標可以由半帶寬累積求得。即第J行位置指標等於前一行位置指標加第J行之半帶寬。而各行之半帶寬決定於該行由上而下第一個非零元素所在列之號碼，簡稱為非零頂列。如第J行由上而下第一個非零元素為S(I,J)，則第J行之非零頂列為I，該行半帶寬則為(J–I)。

結構勁度矩陣各行之非零頂列須由各構桿擴大勁度矩陣之各行的非零頂列決定之。通常假定未擴大之構桿勁度矩陣中之元素皆不為零，故擴大之構桿勁度矩陣之有關各行之非零頂列均相等，其值為有關變數號碼中之最小者。在此所謂有關即與該構桿有關連之轉角號碼。例如構桿3(參考式(5.10))，其擴大矩陣之非零元素所佔之最小列號為2，因此與其有關之行(即第2行與第4行)之非零頂列均為2。其餘無關之行則均為零元素而不影響結構勁度矩陣之半帶寬。將構桿擴大矩陣有關各行之非零頂列與未加入該構桿勁度之前結構勁度矩陣之對應各行之非零頂列做比較，而保留其較小者，即得加入該構桿勁度之後結構勁度矩陣各行之非零頂列

。對每一構桿均做類似比較後，即可得結構勁度矩陣每行之非零頂列。

仍以圖二結構之勁度矩陣為例，在未加入各構桿之勁度矩陣時，令第1行至第6行之非零頂列分別為可能最大值，即(1,2,3,4,5,6)。

(1)加入構桿1之勁度矩陣後，各行之非零頂列變為(1,1,3,4,5,6)。因其有關之行及列為1,2。其中1為最小，故第1,2行之非零頂列均為1。

(2)加入構桿2之勁度矩陣後，各行之非零頂列變為(1,1,2,4,5,6)。因其有關之行及列為2,3。其中2為最小。故第2,3行之非零頂列均為2。但第2行保留原有之較小之非零頂列1。同理：

(3)加入構桿3(有關行為2,4)後，各行之非零頂列變為(1,1,2,2,5,6)。

(4)加入構桿4(有關行為4)後，各行之非零頂列不變仍為(1,1,2,2,5,6)。

(5)加入構桿5(有關行為4,5)後，各行之非零頂列變為(1,1,2,2,4,6)。最後

(6)加入構桿6(有關行為5,6)後，即得結構勁度矩陣，其各行之非零頂列為(1,1,2,2,4,5)。因此各行之半帶寬等於行號減非零頂列即(0,1,1,2,1,1)。其位置指標則為(1,2,3,5,6,7)。須予儲存之元素總數為7+6−1=12。

上述計算過程詳見程式COLID，其中向量L(N)首先儲存各行之非零頂列，然後再儲存由非零頂列算得之各行之位置指標，做為建立結構勁度矩陣時濃縮儲存之依據。有關勁度矩陣之建立見副程式FORMEQ，注意其中僅一行指令不同於帶狀矩陣者。

［表七］ 結構分析−變寬帶矩陣儲存

```
*************************************************************************
      PROGRAM STRAVB
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(5000),K(2,5000)
      EQUIVALENCE (A,K)
      NDIM=5001
C**   ================================================================    **
C**   Program for structural analysis using variable banded matrix       **
C**   ================================================================    **
   10 READ(5,'(2I4)') NE,N
      IF(NE.EQ.0) STOP
C**   +--------------------------------------------------------------+    **
C**   | Dynamic allocate the space for arrays                        |    **
C**   +--------------------------------------------------------------+    **
      N1=1
      N2=N1+2*2*NE               ! SM(2,2,NE)
      N3=N2+2*NE                 ! PM(2,NE)
      N4=N3+2*NE                 ! KK(2,NE)
      N5=N4+N                    ! L(N)
      IF(N5.GT.NDIM) STOP
      CALL INPUT(A(N1),A(N2),K(1,N3),NE)
      CALL COLID(K(1,N3),K(1,N4),NE,N,NN)
C**   +--------------------------------------------------------------+    **
C**   | Allocate space for Variable banded S( 1,NN) & P(N)           |    **
```

```
C**   +-------------------------------------------------------+    **
      N6=N5+NN
      N7=N6+N
      IF(N7.GT.NDIM) STOP
      CALL FORMEQ(A(N1),A(N2),K(1,N3),A(N5),A(N6),K(1,N4),NE,N, 1,NN)
      CALL PRINTV(A(N5),K(1,N4),N, 1,NN,'Stiffness var. banded matrix')
C**   ---------------------------------------------------------    **
      CALL VBDECP(A(N5),K(1,N4),N)
      CALL VBSOLX(A(N5),A(N6),K(1,N4),N)
C**   ---------------------------------------------------------    **
      CALL PRINTV(A(N5),K(1,N4),N, 1,NN,'Decomposed var. banded matrix')
      CALL OUTPUT(A(N1),A(N2),K(1,N3),A(N6),NE,N)
      GO TO 10
      END
*************************************************************************
      SUBROUTINE COLID(KK,L,NE,N,NN)
C**   =========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION KK(2,NE),L(N)
C**   =========================================================    **
C**   Compute L(N) = Location index for variable banded matrix      **
C**            NN  = L(N)+N-1 = Total number of elements to store   **
C**   =========================================================    **
      DO 10 J=1,N
   10 L(J)=J
C**   +-------------------------------------------------------+    **
C**   | MIN  = Min.KK(1:2,K) = Minimum unknown # of member K    |    **
C**   | L(J) = Row number of top non-zero element at column J   |    **
C**   |      = Minimum of MIN and old.L(J) for J=KK(1:2,K)      |    **
C**   +-------------------------------------------------------+    **
      DO 50 K=1,NE
      MIN=N
      DO 30 I=1,2
      J=KK(I,K)
      IF(J.EQ.0.OR.J.GT.N) GO TO 30
      IF(MIN.GT.J) MIN=J
   30 CONTINUE
      DO 40 I=1,2
      J=KK(I,K)
      IF(J.EQ.0.OR.J.GT.N) GO TO 40
      IF(MIN.LT.L(J)) L(J)=MIN
   40 CONTINUE
   50 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | L(J) = Location index of column J                      |    **
C**   +-------------------------------------------------------+    **
      DO 60 J=2,N
   60 L(J)=L(J-1)+(J-L(J))
      NN=L(N)+N-1
      RETURN
      END
*************************************************************************
Loading vector of strucutre

 1.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00-1.0000E+00

Stiffness var. banded matrix

 4.00E+00 2.00E+00 1.20E+01 2.00E+00 4.00E+00 2.00E+00 0.00E+00 1.20E+01
 2.00E+00 8.00E+00 2.00E+00 4.00E+00
```

```
 ...
Decomposed var. banded matrix

 2.00E+00 1.00E+00 3.32E+00 6.03E-01 1.91E+00 6.03E-01-1.91E-01 3.41E+00
 5.87E-01 2.77E+00 7.23E-01 1.86E+00
 2.00E+00
 1.00E+00 3.32E+00
 0.00E+00 6.03E-01 1.91E+00
 0.00E+00 6.03E-01-1.91E-01 3.41E+00
 0.00E+00 0.00E+00 0.00E+00 5.87E-01 2.77E+00
 0.00E+00 0.00E+00 0.00E+00 0.00E+00 7.23E-01 1.86E+00
 ...
================================================================================
```

習題

1. 試寫副程式CBMPY(A,B,C,N,M,L,LA,LB,LC,MA,MB,MC,NA,NB,NC)用以計算帶狀矩陣之乘積。其中
   [A]為NxM之已知矩陣，下半帶寬為LA，上半帶寬為MA，宣告列數為NA≥min(LA+MA, N)；
   [B]為MxL之已知矩陣，下半帶寬為LB，上半帶寬為MB，宣告列數為NB≥min(LB+MB, M)；
   [C]=[A][B]由副程式算出，下半帶寬為LC=min(LA+LB, N–1)，上半帶寬為MC=min(MA+MB, L–1)，宣告列數為NC≥min(LC+MC, N)。

2. 寫一主程式安排下列[A]與[B]矩陣，並呼叫上題副程式，計算[C]矩陣並予印出以便核對(印出形式不拘)

$$[C]=[A][B]=\begin{bmatrix}2&1&0&0&0\\1&2&1&0&0\\0&1&2&1&0\\0&0&1&2&1\\0&0&0&1&2\\0&0&0&0&1\end{bmatrix}\begin{bmatrix}4&2&0&0&0&0\\2&4&2&0&0&0\\1&2&4&2&0&0\\0&1&2&4&2&0\\0&0&1&2&4&2\end{bmatrix}$$

第六章

對稱矩陣之特徵值問題

6.1 前言

在許多工程問題上常須解下列型式之線性方程式：

$$\begin{bmatrix} a_{11}-\lambda & a_{12} & a_{13} \\ a_{21} & a_{22}-\lambda & a_{23} \\ a_{31} & a_{32} & a_{33}-\lambda \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \tag{6.1}$$

或

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \lambda \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} \tag{6.2}$$

式(6.1)及(6.2)可分別寫成如下矩陣式：

$$[[A]-\lambda[I]]\{X\} = \{0\}$$
$$[A]\{X\} = \lambda\{X\}$$

上列方程式只有在λ等於某些特定值時才有解答存在。若$[A]$為$n \times n$之實數對稱矩陣，則使上式有解答之λ值有n個，且皆為實數。若令各值分別為$\lambda_1, \lambda_2, \ldots, \lambda_n$，則對應每一$\lambda_i$，將可由上式求得對應之$\{X\}$，令其為$\{\Phi_i\}$，則$\lambda_i$與$\{\Phi_i\}$應滿足下列條件(以$n=3$為例)：

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} \phi_{1i} \\ \phi_{2i} \\ \phi_{3i} \end{Bmatrix} = \lambda_i \begin{Bmatrix} \phi_{1i} \\ \phi_{2i} \\ \phi_{3i} \end{Bmatrix}, \quad i = 1,2,3 \tag{6.3}$$

以上共n個類似等式，可合併寫成：

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{bmatrix}$$

或寫成

$$[A][\Phi] = [\Phi][\Lambda] \tag{6.4}$$

上式兩邊若前乘$[\Phi]^{-1}$(假設其存在)，或後乘$[\Phi]^{-1}$可分別得：

$$[\Phi]^{-1}[A][\Phi] = [\Lambda] \tag{6.5}$$

$$[A] = [\Phi][\Lambda][\Phi]^{-1} \tag{6.6}$$

上兩式為矩陣$[A]$與其特徵值$[\Lambda]$及其特徵向量$[\Phi]$間之典型關係。
式(6.5)及式(6.6)兩邊各自乘r次可得：

$$[\Phi]^{-1}[A]^r[\Phi] = [\Lambda]^r \tag{6.7}$$

$$[A]^r = [\Phi][\Lambda]^r[\Phi]^{-1} \tag{6.8}$$

由上列關係可知$[A]^r$矩陣之特徵值為$[A]$矩陣之特陣值之r次方，但$[A]$與$[A]^r$之特徵向量相同。

6.2 相似轉換

考慮下列特徵值問題： $[A]\{X\} = \lambda\{X\}$

上式中若令

$$\{X\} = [P]\{\bar{X}\} \tag{6.9}$$

代入可得

$$[A][P]\{\bar{X}\} = \lambda[P]\{\bar{X}\}$$

上式兩邊前乘$[P]^{-1}$可得

$$[P]^{-1}[A][P]\{\bar{X}\} = \lambda\{\bar{X}\} \tag{6.10}$$

或

$$[\bar{A}]\{\bar{X}\} = \lambda\{\bar{X}\} \tag{6.11}$$

式中

$$[\bar{A}] = [P]^{-1}[A][P] \tag{6.12}$$

則$[\bar{A}]$稱為$[A]$的相似轉換，相似轉換的特點為：

(1) $[A]$與$[\bar{A}]$間所有特徵值皆相等。
(2) 特徵向量間保持式(6.9)之關係。
注意$[P]$須為非奇異方陣，即其行列式值不得為零。相似轉換在特徵值問題中十分重要，大部分特徵值及其特徵向量之求法都是利用相似轉換，將原來矩陣轉換為型式較特殊或較容易計算特徵值及特徵向量之矩陣。

解特徵值及特徵向量的方法很多，各法皆有其特點與適用場合。本章將只介紹二種較常用於工程分析之解法，一為賈柯比 (Jacobi)法；一為次空間法 (Subspace iteration)。

6.3 對角矩陣之特徵值及特徵向量

若矩陣$[A]$由式(6.12)做相似轉換後之矩陣$[\bar{A}]$正好為對角矩陣，即式(6.11)可寫成

$$\begin{bmatrix} \bar{a}_{11} & 0 & 0 \\ 0 & \bar{a}_{22} & 0 \\ 0 & 0 & \bar{a}_{33} \end{bmatrix} \begin{Bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{Bmatrix} = \lambda \begin{Bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{Bmatrix} \tag{6.13}$$

由上式可明顯看出其一特徵值為$\lambda_1 = \bar{a}_{11}$，對應之$[\bar{A}]$矩陣之特徵向量為$\{\bar{\Phi}_1\} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$，而對應之$[A]$矩陣之特徵向量為$\{\Phi_1\} = [P]\{\bar{\Phi}_1\} = \begin{Bmatrix} p_{11} \\ p_{21} \\ p_{31} \end{Bmatrix}$，同理可得$\lambda_2 = \bar{a}_{22}$，$\{\bar{\Phi}_2\} = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}$，$\{\Phi_2\} = [P]\{\bar{\Phi}_2\} = \begin{Bmatrix} p_{12} \\ p_{22} \\ p_{32} \end{Bmatrix}$；$\lambda_3 = \bar{a}_{33}$，$\{\bar{\Phi}_3\} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}$，$\{\Phi_3\} = [P]\{\bar{\Phi}_3\} = = \begin{Bmatrix} p_{13} \\ p_{23} \\ p_{33} \end{Bmatrix}$。

或合併寫成

$$[\Lambda] = [\bar{\Lambda}] = \begin{bmatrix} \bar{a}_{11} & 0 & 0 \\ 0 & \bar{a}_{22} & 0 \\ 0 & 0 & \bar{a}_{33} \end{bmatrix}, \quad [\bar{\Phi}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[\Phi] = [P][\bar{\Phi}] = [P] = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{6.14}$$

6.4 賈柯比法

由上節說明，知若能找到一矩陣$[P]$，使$[\bar{A}]=[P]^{-1}[A][P]$為對角矩陣，則得該矩陣對角線上之值為$[A]$矩陣之特徵值及$[P]$矩陣內各行為$[A]$矩陣之特徵向量。賈柯比法係利用一系列的轉換矩陣$[T_1],[T_2],\ldots,[T_m]$，這些轉換矩陣須為正交矩陣，即其逆矩陣等於其轉置矩陣，$[T_k]^{-1}=[T_k]^T$。並計算其相似轉換$[A_1]=[T_1]^T[A][T_1]$，$[A_2]=[T_2]^T[A_1][T_2]$，...直到$[A_m]=[T_m]^T[A_{m-1}][T_m]$等於(或接近)對角矩陣為止。即得$[A]$矩陣之特徵向量等於

$$[\Phi]=[P]=[T_1][T_2]\cdots[T_m] \tag{6.15}$$

且因$[\Phi]^{-1}=[\Phi]^T$，故式(6.5)與式(6.6)可改寫成

$$[\Phi]^T[A][\Phi]=[\Lambda] \tag{6.16}$$

$$[A]=[\Phi][\Lambda][\Phi]^T \tag{6.17}$$

賈柯比法係以下列正交矩陣做為轉換矩陣

$$[T_k]=\begin{bmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & \cos\theta & 0 & \cdots & 0 & -\sin\theta & & \\ & & & 0 & 1 & & & 0 & & \\ & & & \vdots & & \ddots & & \vdots & & \\ & & & 0 & & & 1 & 0 & & \\ & & & \sin\theta & 0 & \cdots & 0 & \cos\theta & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ i\text{ 列} \\ \\ \\ \\ j\text{ 列} \\ \\ \\ \end{matrix} \tag{6.18}$$

$$\begin{matrix} & & & i\text{ 行} & & & & j\text{ 行} & & \end{matrix}$$

注意$[T_k]^T[T_k]=[I]$，即$[T_k]^{-1}=[T_k]^T$。故$[T_k]$為正交矩陣。若(以$n=6$為例)：

$$[A_{k-1}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix} \tag{6.19}$$

則(以$i=3,\ j=5$為例)：

$$[A_k] = [T_k]^T\,[A_{k-1}]\,[T_k] = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccc} & & i & & j & \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \bar{a}_{13} & a_{14} & \bar{a}_{15} & a_{16} \\ a_{21} & a_{22} & \bar{a}_{23} & a_{24} & \bar{a}_{25} & a_{26} \\ \bar{a}_{31} & \bar{a}_{32} & \bar{a}_{33} & \bar{a}_{34} & \bar{a}_{35} & \bar{a}_{36} \\ a_{41} & a_{42} & \bar{a}_{43} & a_{44} & \bar{a}_{45} & a_{46} \\ \bar{a}_{51} & \bar{a}_{52} & \bar{a}_{53} & \bar{a}_{54} & \bar{a}_{55} & \bar{a}_{56} \\ a_{61} & a_{62} & \bar{a}_{63} & a_{64} & \bar{a}_{65} & a_{66} \end{bmatrix} \end{array} \begin{array}{l} \\ \\ i \\ \\ j \\ \\ \end{array} \tag{6.20}$$

注意在$[A_k]$矩陣中僅第i,j兩列與第i,j兩行內之元素與$[A_{k-1}]$矩陣中者不同，其餘元素仍與$[A_{k-1}]$者相同，其中不同之元素由下列各式計算。

$$\bar{a}_{ii} = a_{ii}\cos^2\theta + 2a_{ij}\sin\theta\cos\theta + a_{jj}\sin^2\theta \tag{6.21}$$

$$\bar{a}_{jj} = a_{ii}\sin^2\theta - 2a_{ij}\sin\theta\cos\theta + a_{jj}\cos^2\theta \tag{6.22}$$

$$\bar{a}_{ij} = a_{ij}(\cos^2\theta - \sin^2\theta) + (a_{jj} - a_{ii})\sin\theta\cos\theta \tag{6.23}$$

$$\bar{a}_{li} = a_{li}\cos\theta + a_{lj}\sin\theta\ ,\quad l = 1,2,\ldots,n\ ,\quad \text{但}\neq i,j \tag{6.24}$$

$$\bar{a}_{lj} = -a_{li}\sin\theta + a_{lj}\cos\theta\ ,\quad l = 1,2,\ldots,n\ ,\quad \text{但}\neq i,j \tag{6.25}$$

選擇式(6.18)之$[T_k]$之目的為使$\bar{a}_{ij}=0$，因此須使θ滿足下列條件。

$$\bar{a}_{ij} = a_{ij}\,(\cos^2\theta - \sin^2\theta) + (a_{jj} - a_{ii})\,\sin\theta\,\cos\theta = 0 \tag{6.26}$$

除以$\cos^2\theta$得

$$a_{ij}\,\tan^2\theta + (a_{ii} - a_{jj})\,\tan\theta - a_{ij} = 0$$

故得

$$\tan\theta_{1,2} = \frac{-(a_{ii}-a_{jj}) \pm \sqrt{(a_{ii}-a_{jj})^2 + 4\,a_{ij}^2}}{2\,a_{ij}} \tag{6.27}$$

$$= \frac{2\,a_{ij}}{(a_{ii}-a_{jj}) \pm \sqrt{(a_{ii}-a_{jj})^2 + 4\,a_{ij}^2}} \tag{6.28}$$

當a_{ij}很小時，$|a_{ii}-a_{jj}|$與$\sqrt{(a_{ii}-a_{jj})^2+4\,a_{ij}^2}$之值將非常接近，應避免兩者相減，以減少誤差(亦詳另章誤差分析)。因此：
(1)當$a_{ii}>a_{jj}$時，θ_1宜以式(6.28)計算，θ_2宜以式(6.27)計算；
(2)當$a_{ii}<a_{jj}$時，θ_1宜以式(6.27)計算，θ_2宜以式(6.28)計算。
同時，若要使$\bar{a}_{ii}>\bar{a}_{jj}$，必須選擇$\theta=\theta_1$，因為：
(1)若$a_{ii}>a_{jj}$時，$|\theta_1|<\pi/4$，$\cos^2\theta_1>\sin^2\theta_1$，$2\sin\theta_1\cos\theta_1>0$，故選擇$\theta=\theta_1$時，由式(6.21)及式(6.22)可知$\bar{a}_{ii}>\bar{a}_{jj}$；反之：
(2)若$a_{ii}<a_{jj}$時，$|\theta_1|>\pi/4$，$\cos^2\theta_1<\sin^2\theta_1$，$2\sin\theta_1\cos\theta_1<0$，故選擇$\theta=\theta_1$時，由式(6.21)及式(6.22)可知$\bar{a}_{ii}>\bar{a}_{jj}$。
因此，若要$\bar{a}_{ii}>\bar{a}_{jj}$，則以下式計算

$$\tan\theta = \frac{|a_{ii}-a_{jj}|+\sqrt{(a_{ii}-a_{jj})^2+4\,a_{ij}^2}}{2\,a_{ij}}, \quad \text{當} a_{ii}<a_{jj} \tag{6.29}$$

$$\tan\theta = \frac{2\,a_{ij}}{|a_{ii}-a_{jj}|+\sqrt{(a_{ii}-a_{jj})^2+4\,a_{ij}^2}}, \quad \text{當} a_{ii}\geq a_{jj} \tag{6.30}$$

然後再由下列兩式計算$\cos\theta$及$\sin\theta$。

$$\cos\theta=1/\sqrt{1+\tan^2\theta}\,, \quad \sin\theta=\cos\theta\tan\theta \tag{6.31}$$

另外式(6.21)可改用$\bar{a}_{ii}=a_{ii}+a_{ij}\tan\theta$計算。此式可由$\cos^2\theta=1-\sin^2\theta$，並由式(6.26)之$(a_{jj}-a_{ii})=a_{ij}(\sin^2\theta-\cos^2\theta)/\sin\theta\cos\theta$代入而得。

注意賈柯比法雖每次轉換能使其中一個非對角線上元素轉變為零，但亦可能同時使位於與該元素同行或同列之零元素轉變成非零元素。一般若每次選擇轉換為零之非對角元素均為絕對值最大者，可使轉換次數較少，而所需轉換次數約為非對角元素總數$(n-1)(n-2)/2$之兩倍。

6.5 一般特徵值問題

在工程問題上常遇到下列特徵值問題：

$$[A]\{X\}=\lambda[B]\{X\} \tag{6.32}$$

其中$[A]$與$[B]$皆為對稱矩陣，而且$[B]$為恒正矩陣。欲解上式問題，可先計算下列標準特徵值問題：

$$[B]\{Y\}=\mu\{Y\} \tag{6.33}$$

上式可用前節賈柯比法求得$[B]$矩陣之特徵向量方陣$[U]$及特徵值對角矩陣$[D]$，並由式(6.16)與(6.17)之關係得

$$[U]^T[B][U]=[D] \tag{6.34}$$

$$[B]=[U][D][U]^T \tag{6.35}$$

將式(6.35)代入(6.32)得

$$[A]\{X\}=\lambda[U][D][U]^T\{X\} \tag{6.36}$$

上式前乘$[U]^T$，並令 $$\{X\}=[U]\{\bar{X}\} \tag{6.37}$$

得

$$[U]^T[A][U]\{\bar{X}\}=\lambda[D]\{\bar{X}\} \tag{6.38}$$

上式再前乘$[D]^{-1/2}$，並令 $$\{\bar{X}\}=[D]^{-1/2}\{\bar{\bar{X}}\} \tag{6.39}$$

得 $$[D]^{-1/2}[U]^T[A][U][D]^{-1/2}\{\bar{\bar{X}}\}=\lambda\{\bar{\bar{X}}\} \tag{6.40}$$

再令上式中，$[D]^{-1/2}[U]^T[A][U][D]^{-1/2}=[\bar{\bar{A}}]$，即得下列標準特徵值問題：

$$[\bar{\bar{A}}]\{\bar{\bar{X}}\}=\lambda\{\bar{\bar{X}}\} \tag{6.41}$$

上式仍可用節賈柯比法求得$[\bar{\bar{A}}]$矩陣之特徵向量方陣$[Z]$，及特徵值對角矩陣$[\Lambda]$。由式(6.37)及式(6.39)得

$$\{X\}=[U]\{\bar{X}\}=[U][D]^{-1/2}\{\bar{\bar{X}}\} \tag{6.42}$$

故滿足式(6.32)之特徵向量$\{X\}$之方陣$[V]$為

$$[V]=[U][D]^{-1/2}\{Z\} \tag{6.43}$$

注意$[U]$與$[Z]$雖皆為正交矩陣，而$[D]^{-1/2}$並非正交矩陣，故$[V]$即非正交矩陣。但$[V]$，$[\Lambda]$，$[B]$及$[A]$間仍有式(6.44)與式(6.45)之關係。該等關係可由式(6.43)代入，並引用式(6.34)及$[U]^T[U]=[I]$與$[Z]^T[Z]=[I]$之關係即得式(6.44)。由式(6.32)可知$[A][V]=[B][V][\Lambda]$代入式(6.45)即得該式之關係。

$$[V]^T[B][V]=[I] \tag{6.44}$$

$$[V]^T[A][V]=[V]^T[B][V][\Lambda]=[\Lambda] \tag{6.45}$$

若下列特徵值問題中，$[B]$為恒正對稱矩陣，$[A]$為對稱矩陣。

$$[B]\{X\} = \bar{\lambda}[A]\{X\} \tag{6.46}$$

則上式可改寫為

$$[A]\{X\} = \frac{1}{\bar{\lambda}}[B]\{X\} = \lambda[B]\{X\} \tag{6.47}$$

然後照式(6.32)至式(6.45)之分析過程求得

$$[V] = [U][D]^{-1/2}[Z]$$
$$[V]^T[B][V] = [I]$$
$$[\Lambda] = [V]^T[A][V]$$

因此得式(6.46)之特徵值$\bar{\lambda}$之對角矩陣為$[\bar{\Lambda}] = [\Lambda]^{-1}$，其特徵向量方陣仍為$[V]$，但亦可改為$[\bar{V}] = [V][\Lambda]^{-1/2}$，即$[V] = [\bar{V}][\Lambda]^{1/2}$，代入上列各式可得$[\bar{V}]$，$[\bar{\Lambda}]$，$[B]$與$[A]$之間的關係如下：

$$[\bar{V}]^T[A][\bar{V}] = [I] \tag{6.48}$$

$$[\bar{V}]^T[B][\bar{V}] = [\bar{\Lambda}] \tag{6.49}$$

其中

$$[\bar{V}] = [U][D]^{-1/2}[Z][\Lambda]^{-1/2} \tag{6.50}$$

若矩陣$[A]$與$[B]$均非恒正矩陣，即式(6.38)中之對角矩陣$[D]$之對角元素有負值也有零值，則式(6.39)之$[D]^{-1/2}$即非實數或不存在。對角元素有負值時其處理方式會使矩陣變成不對稱，詳下一章之說明。以下說明對角元素為零時之處理方法：式(6.38)中，令$[\tilde{A}] = [\tilde{D}]^{-1/2}[U]^T[A][U][\tilde{D}]^{-1/2}$，$\{\bar{X}\} = [\tilde{D}]^{-1/2}\{\tilde{X}\}$，其中$[\tilde{D}]$為$[D]$中之0對角元素改為1後之對角矩陣，以$n = 4$為例，設$d_{33} = 0$，則$\tilde{d}_{33} = 1$，可得$[\tilde{A}]\{\tilde{X}\} = \lambda[J]\{\tilde{X}\}$，即

$$\begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} & \tilde{a}_{13} & \tilde{a}_{14} \\ \tilde{a}_{21} & \tilde{a}_{22} & \tilde{a}_{23} & \tilde{a}_{24} \\ \tilde{a}_{31} & \tilde{a}_{32} & \tilde{a}_{33} & \tilde{a}_{34} \\ \tilde{a}_{41} & \tilde{a}_{42} & \tilde{a}_{43} & \tilde{a}_{44} \end{bmatrix} \begin{Bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{Bmatrix} = \lambda \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & & 1 \end{bmatrix} \begin{Bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{Bmatrix} \tag{6.51}$$

注意上式與標準特徵值問題之差異為$[J]$矩陣中有0對角元素。令

$$[G] = \begin{bmatrix} 1 & & & \\ & 1 & & \\ -\frac{\tilde{a}_{31}}{\tilde{a}_{33}} & -\frac{\tilde{a}_{32}}{\tilde{a}_{33}} & 1 & -\frac{\tilde{a}_{34}}{\tilde{a}_{33}} \\ & & & 1 \end{bmatrix} \tag{6.52}$$

並令$[\hat{A}]=[G]^T[\tilde{A}][G]$，$\{\tilde{X}\}=[G]\{\hat{X}\}$，可得

$$[\hat{A}]=[G]^T[\tilde{D}]^{-1/2}[U]^T[A][U][\tilde{D}]^{-1/2}[G] \tag{6.53}$$

$$\{X\}=[U][\tilde{D}]^{-1/2}[G]\{\hat{X}\} \tag{6.54}$$

$$[\hat{A}]\{\hat{X}\}=\lambda[J]\{\hat{X}\} \tag{6.55}$$

即

$$\begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} & 0 & \hat{a}_{14} \\ \hat{a}_{21} & \hat{a}_{22} & 0 & \hat{a}_{24} \\ 0 & 0 & \tilde{a}_{33} & 0 \\ \hat{a}_{41} & \hat{a}_{42} & 0 & \hat{a}_{44} \end{bmatrix} \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{Bmatrix} = \lambda \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & & 1 \end{bmatrix} \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{Bmatrix} \tag{6.56}$$

上式可將$\hat{x}_3$去除，即成為標準特徵值問題。亦可照下述做法保持矩陣階數不變：將上式中$[J]$之對角元素為0者改為1，即成為$[I]$，對應$[\hat{A}]$之對角元素改為∞（相當於將0視為接近0之非零值），令其為$[\bar{\bar{A}}]$，即得$[\bar{\bar{A}}]\{\hat{X}\}=\lambda\{\hat{X}\}$，或

$$\begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} & 0 & \hat{a}_{14} \\ \hat{a}_{21} & \hat{a}_{22} & 0 & \hat{a}_{24} \\ 0 & 0 & \infty & 0 \\ \hat{a}_{41} & \hat{a}_{42} & 0 & \hat{a}_{44} \end{bmatrix} \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{Bmatrix} = \lambda \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{Bmatrix} \tag{6.57}$$

上式即與式(6.41)類似為標準特徵值問題，可由賈柯比法得其特徵向量方陣$[Z]$與特徵值對角矩陣$[\Lambda]$。所得之$[\Lambda]$即為式(6.32)之特徵值。注意其中之$\lambda_3=\infty$。而式(6.32)之特徵向量方陣$[V]$可由下式計算

$$[V]=[U][\tilde{D}]^{-1/2}[G][Z] \tag{6.58}$$

本節所述之賈柯比法必須同時求出所有之特徵值。但一般工程問題常無此必要，且若矩陣階數甚大時，所需儲存位置及計算時間皆為數甚巨。又賈柯比法亦無法利用帶狀矩陣或變寬帶矩陣之特性以節省儲存位置，因為賈柯比法雖逐步消去非對角元素，但每一非對角元素於每一運算過程中仍隨時有從零值轉換為非零值的可能，故不能像高斯消去法用於解方程式時，僅儲存定寬帶內或變寬帶內之元素。本章另將介紹之次空間法 (Subspace iteration)對於一般常用之帶狀矩陣或變寬帶矩陣可僅儲存帶內元素，且可僅計算所需個數之特徵值及特徵向量，因此在儲存位置上及時間上均甚節省。

在介紹次空間法之前擬先介紹三個與該法有關之方法，分別為芮萊(Rayleigh)法，芮萊瑞茲 (Rayleigh–Ritz)法及乘冪法 (Power method)。

6.6 芮萊法

考慮下列特徵問題

$$[K]\{X\} = \omega^2 [M]\{X\} \tag{6.59}$$

該方程式常見於一般動力分析中，其中$[K]$為勁度矩陣，$[M]$為質量矩陣，ω為自然振動頻率，$\{X\}$為振態向量。假設自然振態向量為$\{V\}$，即該系統按$\{X\} = \{V\}$之方式振動，則$\{R(t)\} = \{V\}\cos\omega(t-\phi)$，因此可得該系統之最大應變能$V_{max}$，及最大動能$T_{max}$，分別為

$$\begin{aligned} V_{max} &= \frac{1}{2}\{R\}^T_{max}[K]\{R\}_{max} \\ &= \frac{1}{2}\{V\}^T[K]\{V\} \end{aligned} \tag{6.60}$$

$$\begin{aligned} T_{max} &= \frac{1}{2}\{\dot{R}\}^T_{max}[M]\{\dot{R}\}_{max} \\ &= \frac{1}{2}\omega^2\{V\}^T[M]\{V\} \end{aligned} \tag{6.61}$$

由能量轉換關係$V_{max} = T_{max}$，可得

$$\omega^2 = \frac{\{V\}^T[K]\{V\}}{\{V\}^T[M]\{V\}} \tag{6.62}$$

因假設之$\{V\}$非正確之振態，故由上式算得之ω^2僅為近似值，但該近似值通常均十分接近正確值，即根據芮萊原理，當$\{V\}$為正確之振態向量時，由上式算得之正確ω^2為局部極小值。

6.7 芮萊瑞茲法

前節利用一個假設之向量$\{V\}$可以求得近似之特徵值，如果進一步假設振態向量係由數個(設為M個，$M<<N$)向量$\{V_1\},\{V_2\},\ldots\{V_M\}$之線性組合，即

$$\begin{aligned} \{X\} &= \{V_1\}y_1 + \{V_2\}y_2 + \cdots + \{V_M\}y_M \\ &= [V]\{Y\} \end{aligned} \tag{6.63}$$

將上式代入式(6.60)及式(6.61)可得

$$\begin{aligned} V_{max} &= \frac{1}{2}\{R\}^T_{max}[K]\{R\}_{max} \\ &= \frac{1}{2}\{Y\}^T[V]^T[K][V]\{Y\} \qquad (6.64) \\ T_{max} &= \frac{1}{2}\{\dot{R}\}^T_{max}[M]\{\dot{R}\}_{max} \\ &= \frac{1}{2}\omega^2\{Y\}^T[V]^T[M][V]\{Y\} \qquad (6.65) \end{aligned}$$

由$V_{max} = T_{max}$之關係，可得

$$\begin{aligned} \omega^2 &= \frac{\{Y\}^T[V]^T[K][V]\{Y\}}{\{Y\}^T[V]^T[M][V]\{Y\}} \\ &= \frac{\{Y\}^T[\bar{K}]\{Y\}}{\{Y\}^T[\bar{M}]\{Y\}} \qquad (6.66) \end{aligned}$$

根據芮萊原理，由上式算得之ω^2若為正確值，則亦應為局部最小值，在$\{X\}$為$\{V_1\},\{V_2\},\ldots\{V_M\}$之線性組合下，欲得最小之$\omega^2$，只能調整各向量之組合係數$y_i$。因此，由上式對各$y_i$偏微分可得$\omega^2$為極小值之$M$個條件式如下

$$\begin{aligned} \left\{\frac{\partial\omega^2}{\partial y_i}\right\} &= \frac{\{Y\}^T[\bar{M}]\{Y\}\{\frac{\partial(\{Y\}^T[\bar{K}]\{Y\})}{\partial y_i}\} - \{Y\}^T[\bar{K}]\{Y\}\{\frac{\partial(\{Y\}^T[\bar{M}]\{Y\})}{\partial y_i}\}}{(\{Y\}^T[\bar{M}]\{Y\})^2} \\ &= \{0\} \qquad (6.67) \end{aligned}$$

由式(6.66)知$\{Y\}^T[\bar{K}]\{Y\} = \omega^2\{Y\}^T[\bar{M}]\{Y\}$，代入得

$$\{\frac{\partial(\{Y\}^T[\bar{K}]\{Y\})}{\partial y_i}\} - \omega^2\{\frac{\partial(\{Y\}^T[\bar{M}]\{Y\})}{\partial y_i}\} = \{0\}$$

或

$$[\bar{K}]\{Y\} - \omega^2[\bar{M}]\{Y\} = \{0\} \qquad (6.68)$$

上式與式(6.59)相似，同為特徵值問題，但上式中

$$[\bar{K}] = [V]^T[K][V] \qquad (6.69)$$

$$[\bar{M}] = [V]^T[M][V] \qquad (6.70)$$

皆僅為$M\times M$之矩陣，可以利用賈柯比法或其他方法計算M個特徵值ω_i^2及其對應之M個特徵向量$\{\Psi_i\}$，因此而得M個最佳向量組合$[\bar{V}]_{N\times M} = [V]_{N\times M}[\Psi]_{M\times M}$。

6.8 乘冪法

現在再回來考慮下列標準特徵值問題

$$[A]\{X\} = \lambda\{X\}$$

設$[A]$為$n\times n$之方矩陣，特徵值為$\lambda_1, \lambda_2, \ldots, \lambda_n$，並假設$\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n$，對應之特徵向量為$\{\phi_1\}, \{\phi_2\}, \ldots \{\phi_n\}$，且為互相正交之向量。則任一向量$\{X_o\}$均可寫成$\{\phi_1\}$至$\{\phi_n\}$之$n$個特徵向量之線性組合，即

$$\{X_o\} = C_1\{\phi_1\} + C_2\{\phi_2\} + \cdots + C_n\{\phi_n\}$$

式中$C_i = \{\phi_i\}^T\{X_o\}/(\{\phi_i\}^T\{\phi_i\})$，上式前乘矩陣$[A]$，可得

$$\begin{aligned}[A]\{X_o\} &= C_1[A]\{\phi_1\} + C_2[A]\{\phi_2\} + \cdots + C_n[A]\{\phi_n\} \\ &= C_1\lambda_1\{\phi_1\} + C_2\lambda_2\{\phi_2\} + \cdots + C_n\lambda_n\{\phi_n\}\end{aligned}$$

依此將式(6.71)連續前乘m個$[A]$矩陣，可得

$$\begin{aligned}\{X_m\} &= [A]^m\{X_o\} \\ &= C_1[A]^m\{\phi_1\} + C_2[A]^m\{\phi_2\} + \cdots + C_n[A]^m\{\phi_n\} \\ &= C_1\lambda_1^m\{\phi_1\} + C_2\lambda_2^m\{\phi_2\} + \cdots + C_n\lambda_n^m\{\phi_n\} \\ &= \lambda_1^m\left(C_1\{\phi_1\} + C_2(\frac{\lambda_2}{\lambda_1})^m\{\phi_2\} + \cdots + C_n(\frac{\lambda_n}{\lambda_1})^m\{\phi_n\}\right) \\ &\approx C_1\lambda_1^m\{\phi_1\} \qquad (6.71)\end{aligned}$$

由式(6.71)可知由任意向量$\{X_o\}$連續前乘數個$[A]$矩陣時，將漸漸趨近於最大特徵值之特徵向量$\{\phi_1\}$乘一常數，而連續二個向量長度之比等於最大特徵值，即$|X_m|/|X_{m-1}| \approx \lambda_1$。

6.9 次空間法

次空間法係對於下列特徵值問題交替利用芮萊瑞茲法與乘冪法的一種反覆求解過程。利用乘冪法時，須將該式改寫成式(6.73)之標準特徵值問題，以便求得實際振動分析所需之較小之ω^2值(即較大之$1/\omega^2$)。

$$[K]\{X\} = \omega^2[M]\{X\} \qquad (6.72)$$

$$[K]^{-1}[M]\{X\} = \frac{1}{\omega^2}\{X\} \qquad (6.73)$$

現將次空間法之運算過程分為下列步驟，其中步驟(2)至(3)為乘冪法。步驟(4)至(7)為芮萊瑞茲法。利用乘冪法之目的為使$[V]$向量轉向至較小ω值之特徵向量。

(1)假設M個向量$[V^o]$。
(2)計算 $[U^o]=[M][V^o]$。
(3)計算 $[V^1]=[K]^{-1}[U^o]$。
(4)計算 $[\bar{K}]=[V^1]^T[K][V^1]$。
(5)計算 $[\bar{M}]=[V^1]^T[M][V^1]$。
(6)計算 $[\bar{K}]\{Y\}=\omega^2[\bar{M}]\{Y\}$之特徵值對角矩陣$[\Lambda]$及特徵向量方陣$[\Psi]$。
(7)計算 $[V^2]=[V^1][\Psi]$。
(8)比較步驟(6)算得之$[\Lambda]$是否已收斂，若未收斂則重新假設$[V^o]=[V^2]$，回至步驟(2)重覆計算。
(9)若已收斂，則得M個特徵值$[\Lambda]$之特徵向量為$[\Phi]=[V^2]$。

於實際求解過程中，常將上列計算步驟改為：

(1)假設M個向量$[U^o]$ $(=[M][V^o])$。
(2)計算 $[V^1]=[K]^{-1}[U^o]$。
(3)計算 $[\bar{K}]=[V^1]^T[U^o]$ $(=[V^1]^T[K][V^1])$。
(4)計算 $[U^1]=[M][V^1]$。
(5)計算 $[\bar{M}]=[U^1]^T[V^1]$ $(=[V^1]^T[M][V^1])$。
(6)計算 $[\bar{K}]\{Y\}=\omega^2[\bar{M}]\{Y\}$之特徵值對角矩陣$[\Lambda]$及特徵向量方陣$[\Psi]$。
(7)計算 $[U^2]=[U^1][\Psi]$ $(=[M][V^1][\Psi]=[M][V^2])$。
(8)比較步驟(6)算得之$[\Lambda]$是否已收斂，若未收斂則重新假設$[U^o]=[U^2]$，回至步驟(2)重覆計算。
(9)若已收斂，則計算$[V^2]=[V^1][\Psi]$，而得M個特徵值$[\Lambda]$之特徵向量為$[\Phi]=[V^2]$。

注意步驟(2)計算$[V^1]=[K]^{-1}[U^o]$時，可利用$[K]$矩陣之克雷斯基分解矩陣$[L]$對$[U^o]$先做前進代入再做反向代入而得$[V^1]$，而$[L]$之計算可在開始次空間法之前先求出，而且可利用變寬帶矩陣之儲存方式。故即使$[K]$矩陣階數甚大，若只需求很少數之特徵值及特徵向量時利用次空間法可以很有效而快速求出。通常上述反覆計算約需十數次。若M值取較所需之特徵值個數為多時，其收斂較快，但每次運算量增多。一般常取$M=\min(2p, p+5)$為宜，其中p為所需之特徵值個數。

6.10 副程式 $JACOBI$

副程式係利用賈柯比法計算$[A]\{X\}=\lambda\{X\}$之特徵值對角矩陣$[\Lambda]$及特徵向量方陣$[\Phi]$。$[A]$須為實數對稱矩陣。因此僅須已知上三角部分之元素。其儲存方式有二：

(1)儲存於$DIMENSION\ A(NA,NA)$之上三角部分，即a_{ij}存於$A(i,j)$，但程式中JA及JB之計算須改用!後所示者，且須增加一呼叫參數NA。
(2)儲存於$DIMENSION\ A(N*(N+1)/2)$，令a_{ij}存於$A(i+(j-1)*j/2)$，寫成$A(i+(j-1)*j/2)=a_{ij}$，即$A(1)=a_{11}$，$A(2)=a_{12}$，$A(3)=a_{22}$，$A(4)=a_{13}$，$A(5)=a_{23}$，$A(6)=a_{33}$，餘類推。

$[A]$矩陣經副程式運算後轉換為對角矩陣，其對角元素即為特徵值，其對應特徵向量則分別存於$DIMENSION\ V(NV,NV)$之第1行至第N行。

$A(NA,NA)$或$A(N(N+1)/2)$	=	已知之$[A]$矩陣，經運算後對角線為特徵值$[\Lambda]$。
$V(NV,NV)$	=	(1)用以存放特徵向量方陣$[\Phi]$ $(JEVC=+1)$；或 (2)用以前乘特徵向量方陣$[\Phi]$ $(JEVC=-1)$。
N	=	矩陣$[A]$之階數。
$X(N)$	=	臨時運算位置。
$IH(N)$	=	臨時運算位置。
$JEVC$	=	0，不計算特徵向量，可不留$[V]$矩陣之位置。
	=	1，計算特徵向量$[\Phi]$，結果存於$[V]$。
	=	-1，計算特徵向量$[\Phi]$，並後乘於$[V]$。
$EPSI$	=	若$\lvert a_{ij}\rvert<\lvert a_{kk}\rvert*EPSI$，則認為$[A]$矩陣已轉換成對角矩陣$[\Lambda]$，而不再轉換。其中$a_{ij},a_{kk}$分別為轉換過程中$[A]$矩陣之絕對值最大非對角元素及絕對值最小對角元素。
NV	=	矩陣$[V]$宣告列數。
NA	=	矩陣$[A]$宣告列數。

[表一] 副程式$JACOBI$

```
************************************************************************
      SUBROUTINE JACOBI(A,V,N,X,IH,JEVC,M,EPSI,NV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),V(1),X(N),IH(N)
C**   ============================================================    **
C**   DIMENSION A(N*(N-1)/2),V(NV,N),X(N),IH(N) ! A(NA,N)             **
```

```
C**   ------------------------------------------------------------    **
C**   [A] {X} = d {X} --> [A] [V] = [V] [D]                           **
C**   ------------------------------------------------------------    **
C**   Input : A,V,N,JEVC,M,EPSI,NV,NA      Output : A,V,X,M           **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = [A] matrix required upper triangular part only         **
C*O          = [D] diagonal matrix                                    **
C*I   V(N,N) = Matrix to be post-multiplied by eigenvector matrix     **
C*O          = EiVec(JEVC=1); None(JEVC=0); input.V*EiVec(JEVC=-1)    **
C*I   N      = Order of matrix [A]                                    **
C*O   X(N)   = Eigenvalues (also in the diagonal of [A])              **
C*W   IH(N)  = Working array for row I of max.A(I,J) in col J         **
C*I   JEVC   = 1 : Find eigenvectors and return in V(N,N)             **
C*I          = 0 : Do not find eigenvectors and V(N,N) is dummy       **
C*I          =-1 : Find eigenvectors and post-multiply to V(N,N)      **
CIO   M      = Iteration count                                        **
C*I   EPSI   = Convergence if max.|A(I,J)| < min.A(K,K)*EPSI          **
C*I   NV,NA  = Row dimensions of array V(NV,N) and A(NA,N)            **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Initialize : Set M=0, [V]=[I] (JEVC=1), X(N) and IH(N)    |    **
C**   +----------------------------------------------------------+    **
C**   M=0
      IF(JEVC.GT.0) THEN
        DO 20 J=1,N
        JV=(J-1)*NV
          DO 10 I=1,N
          V(I+JV)=0.0
   10     CONTINUE
        V(J+JV)=1.0
   20   CONTINUE
      ENDIF
      IF(N.LE.1) RETURN
C**   +----------------------------------------------------------+    **
C**   | X(JX) = max.|A(IH(JX),JX)| in col JX  (IH(JX) < JX)      |    **
C**   | AMIN  = min.A(K,K) (K=1:N)                               |    **
C**   +----------------------------------------------------------+    **
      AMIN=DABS(A(1))
      DO 40 JX=2,N
      JA=(JX-1)*JX/2 ! JA=(JX-1)*NA
      AMIN=DMIN1(AMIN,DABS(A(JX+JA)))
      X(JX)=-1.0
      DO 40 II=1,JX-1
      IF(DABS(A(II+JA)).LE.X(JX)) GO TO 40
        X(JX)=DABS(A(II+JA))
        IH(JX)=II
   40 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Begin loops : Return if converged                        |    **
C**   +----------------------------------------------------------+    **
C**   | Find IP,JP for max.|A(IP,JP)| from X(N) and IH(N)         |    **
C**   +----------------------------------------------------------+    **
   50 M=M+1
C**   CALL PRINTT(A,N,'Matrix A')
      AIJ=-1.0
      DO 60 JX=2,N
      IF(X(JX).LE.AIJ) GO TO 60
        AIJ=X(JX)
        IP=IH(JX)
        JP=JX
```

```
   60 CONTINUE
      IF(AIJ.LE.AMIN*EPSI) RETURN
C**   ------------------------------------------------------------    **
      IA=IP
      IB=JP
      JA=(IP-1)*IP/2 ! JA=(IP-1)*NA
      JB=(JP-1)*JP/2 ! JB=(JP-1)*NA
C**   +----------------------------------------------------------+    **
C**   | Compute the rotation angle                               |    **
C**   +----------------------------------------------------------+    **
      AIJ=2.*A(IA+JB)
      AII=A(IA+JA)
      AJJ=A(IB+JB)
      TANG=AIJ/(DABS(AJJ-AII)+DSQRT((AJJ-AII)**2+AIJ**2))
      AIJ=A(IA+JB)*TANG
      A(IA+JB)=0.0
      IF(AII.GE.AJJ) THEN
        A(IA+JA)=AII+AIJ
        A(IB+JB)=AJJ-AIJ
      ELSE
        A(IA+JA)=AJJ+AIJ
        A(IB+JB)=AII-AIJ
        TANG=1.0/TANG
      ENDIF
      COSS=1.0/(1.0+TANG*TANG)
      COSN=DSQRT(COSS)
      SINE=TANG*COSN
C**   AMIN=DMIN1(AMIN,DABS(A(IB+JB)))
C**   +----------------------------------------------------------+    **
C**   |                  IP                    JP                |    **
C**   |    / .         *                     *                \ |    **
C**   |    |   .   A(IA+JA)              A(IB+JB)               | |    **
C**   |    |     .    *                     *                  | |    **
C**   | IP |       A(IP,IP) * A(IA+JA) * A(IP,JP) * A(IA+JA) * | |    **
C**   |    |                .             *                    | |    **
C**   |    |                     .       A(IB+JB)             | |    **
C**   |    |                          .    *                  | |    **
C**   | JP |                            A(JP,JP) * A(IB+JB) * | |    **
C**   |    |                                       .          | |    **
C**   |    |                                            .     | |    **
C**   |    \                                                . / |    **
C**   +----------------------------------------------------------+    **
C**   | Reset X(IP),IH(IP) and X(JP),IH(JP) for the changes      |    **
C**   +----------------------------------------------------------+    **
      JX=IP
      JY=JP
      X(JX)=0.0
      X(JY)=0.0
      DO 250 JJ=1,N
      IF(JJ.LT.IP) THEN
C**     JX=IP
C**     JY=JP
        IA=JJ
        IB=JJ
C**     JA=(IP-1)*IP/2 ! JA=(IP-1)*NA
C**     JB=(JP-1)*JP/2 ! JB=(JP-1)*NA
      ELSE IF(JJ.EQ.IP) THEN
        IA=IP
        GO TO 250
      ELSE
```

```
        JX=JJ
        JA=JA+JJ-1 ! JA=JA+NA
        IF(JJ.LT.JP) THEN
C**       JX=JJ
C**       JY=JP
C**       IA=IP
          IB=JJ
C**       JA=(JJ-1)*JJ/2 ! JA=(JJ-1)*NA
C**       JB=(JP-1)*JP/2 ! JB=(JP-1)*NA
        ELSE IF(JJ.EQ.JP) THEN
          IB=JP
          GO TO 250
        ELSE
C**       JX=JJ
          JY=JX
C**       IA=IP
C**       IB=JP
C**       JA=(JJ-1)*JJ/2 ! JA=(JJ-1)*NA
          JB=JA
        ENDIF
      ENDIF
      AIJ=A(IA+JA)
      A(IA+JA)= AIJ*COSN+A(IB+JB)*SINE
      A(IB+JB)=-AIJ*SINE+A(IB+JB)*COSN
C**   +-----------------------------------------------------------+    **
C**   | Reset X(JX),IH(JX) if the max.X(JX) has been changed      |    **
C**   +-----------------------------------------------------------+    **
      IF(IH(JX).EQ.IP.OR.IH(JY).EQ.JP) THEN
        X(JX)=-1.0
        DO 180 II=1,JJ-1
        IF(X(JX).GE.ABS(A(II+JA))) GO TO 180
          X(JX)=ABS(A(II+JA))
          IH(JX)=II
  180   CONTINUE
        IF(JJ.GE.JP) GO TO 250
      ENDIF
C**   +-----------------------------------------------------------+    **
C**   | Update X(JX),IH(JX) and X(JY),IH(JY)                      |    **
C**   +-----------------------------------------------------------+    **
      IF(DABS(A(IA+JA)).GT.X(JX)) THEN
        X(JX)=DABS(A(IA+JA))
        IH(JX)=IA
      ENDIF
      IF(DABS(A(IB+JB)).GT.X(JY)) THEN
        X(JY)=DABS(A(IB+JB))
        IH(JY)=IB
      ENDIF
  250 CONTINUE
C**   +-----------------------------------------------------------+    **
C**   | V(N,N) post-multiply by the rotation matrix               |    **
C**   +-----------------------------------------------------------+    **
      IF(JEVC.NE.0) THEN
        JU=(IP-1)*NV
        JV=(JP-1)*NV
        DO 310 I=1,N
          AIJ=V(I+JU)
          V(I+JU)= AIJ*COSN+V(I+JV)*SINE
          V(I+JV)=-AIJ*SINE+V(I+JV)*COSN
  310   CONTINUE
      ENDIF
```

```
      GO TO 50
      END
*************************************************************************
      SUBROUTINE PRINTT(A,N,NOTE)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1)
      CHARACTER*(*) NOTE
C**   ============================================================    **
C**   Print transpose of upper triangle matrix                        **
C**   ============================================================    **
      WRITE(6,'(/1X,A/)') NOTE
      DO 60 J=1,N
   60 WRITE(*,'(1X,1P8E10.3)') (A(I),I=(J-1)*J/2+1,(J+1)*J/2)
      RETURN
      END
*************************************************************************
```

6.11 副程式 $AVEXBV$

副程式係利用二次$JOCOBI$法計算$[A]\{X\}=\lambda[B]\{X\}$之特徵值對角矩陣$[\Lambda]$及特徵向量方陣$[V]$。$[A]$與$[B]$均為對稱矩陣，$[A]$與$[B]$均照副程式$JOCOBI$中所述第二種方式儲存。照第一種方式儲存者僅須略加修改即可。

$A(N(N+1)/2)$	=	$[A]$之上三角部分；運算後，對角線為特徵值$[\Lambda]$。
$B(N(N+1)/2)$	=	$[B]$之上三角部分，運算後，原來元素不保留。
$V(NV,NV)$	=	用以存放特徵向量方陣$[Z]$。
$X(N)$	=	用以存放特徵對角矩陣$[\Lambda]$。
$IH(N)$	=	臨時運算位置。
$IH(N)$	=	臨時運算位置。
$EPSI$	=	轉叫副程式JACOBI時使用。
NV	=	矩陣$[V]$宣告列數。
N	=	矩陣$[A]$與$[B]$之階數。

［表二］ 副程式$AVEXBV$與$BVEXAV$

```
*************************************************************************
      SUBROUTINE AVEXBV(A,B,V,N,X,IH,M,EPSI,NV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),V(1),X(N),IH(N)
C**   ============================================================    **
C**   DIMENSION A,B(N*(N-1)/2),V(NV,N),X(N),IH(N) ! A,B(NA,N)         **
C**   ------------------------------------------------------------    **
C**   [A] {X} = d [B] {X}   --->   [A] [V] = [B] [V] [D]              **
```

```
C**   Normalize as : [V]'[A] [V] = [D],      [V]'[B] [V] = [I]        **
C**   ------------------------------------------------------------    **
C**   Input : A,B,V,N,M,EPSI,NV,NA       Output : A,V,X,M             **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = [A] matrix required upper triangular part only         **
C*O          = [D] diagonal matrix                                    **
CIW   B(N,N) = [B] matrix required upper triangular part only         **
C*O   V(N,N) = Eigen vector matrix                                    **
C*I   N      = Order of matrix [A]                                    **
C*O   X(N)   = Eigenvalues (also in the diagonal of [A])              **
C*W   IH(N)  = Working array for row I of max.A(I,J) in col J         **
C*O   M      = Iteration count                                        **
C*I   EPSI   = Convergence if max.|A(I,J)| < min.A(K,K)*EPSI          **
C*I   NV,NA  = Row dimensions of array V(NV,N) and A(NA,N)            **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Solve [B] {Y} = e {Y}  for  [V] & [E]                     |    **
C**   +----------------------------------------------------------+    **
      M=0
      CALL JACOBI(B,V,N,X,IH,1,M,EPSI,NV)
C**   +----------------------------------------------------------+    **
C**   | Compute : [V] [E]^{-1/2} ---> [V]                         |    **
C**   +----------------------------------------------------------+    **
      DO 20 J=1,N
      JV=(J-1)*NV
      JA=(J-1)*J/2 ! JA=(J-1)*NA
      IF(B(J+JA).GT.0.0) THEN
        BB=1.0/DSQRT(B(J+JA))
        DO 10 I=1,N
        V(I+JV)=V(I+JV)*BB
   10   CONTINUE
      ENDIF
   20 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Compute : [V]'[A] [V] ---> [A]                            |    **
C**   +----------------------------------------------------------+    **
      CALL MAV(A,B,A,V,X,N,N,NV)
      CALL MAV(A,B,B,V,X,N,0,NV)
C**   +----------------------------------------------------------+    **
C**   | Compute : [G]'[A] [G] ---> [A];  [V] [G] ---> [V]         |    **
C**   +----------------------------------------------------------+    **
      CALL GAG(A,B,V,N,X,EPSI,NV)
C**   +----------------------------------------------------------+    **
C**   | Solve [A] {X} = d {X}  for  [V] & [D]                     |    **
C**   +----------------------------------------------------------+    **
      CALL JACOBI(A,V,N,X,IH,-1,M,EPSI,NV)
C**   +----------------------------------------------------------+    **
C**   | Save eigen values in X(N)                                 |    **
C**   +----------------------------------------------------------+    **
      DO 95 J=1,N
      JA=(J-1)*J/2 ! JA=(J-1)*NA
      X(J)=A(J+JA)
   95 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE BVEXAV(A,B,V,N,X,IH,M,EPSI,NV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),V(1),X(N),IH(N)
```

```
C**   ============================================================    **
C**   DIMENSION A,B(N*(N-1)/2),V(NV,N),X(N),IH(N) ! A,B(NA,N)        **
C**   ------------------------------------------------------------    **
C**   [B] {X} = d [A] {X}   --->   [B] [V] = [A] [V] [D]              **
C**   Normalize as : [V]'[A] [V] = [I],      [V]'[B] [V] = [D]        **
C**   ------------------------------------------------------------    **
C**   Input : A,B,V,N,M,EPSI,NV,NA       Output : A,V,X,M             **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = [A] matrix required upper triangular part only         **
C*O          = [D] diagonal matrix                                    **
CIW   B(N,N) = [B] matrix required upper triangular part only         **
C*O   V(N,N) = Eigen vector matrix                                    **
C*I   N      = Order of matrix [A]                                    **
C*O   X(N)   = Eigenvalues (also in the diagonal of [A])              **
C*W   IH(N)  = Working array for row I of max.A(I,J) in col J         **
C*O   M      = Iteration count                                        **
C*I   EPSI   = Convergence if max.|A(I,J)| < min.A(K,K)*EPSI          **
C*I   NV,NA  = Row dimensions of array V(NV,N) and A(NA,N)            **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Solve [B] {Y} = e {Y}  for  [V] & [E]                     |    **
C**   +----------------------------------------------------------+    **
      M=0
      CALL JACOBI(B,V,N,X,IH,1,M,EPSI,NV)
C**   +----------------------------------------------------------+    **
C**   | Compute : [V] [E]^{-1/2} ---> [V]                         |    **
C**   +----------------------------------------------------------+    **
      DO 20 J=1,N
      JV=(J-1)*NV
      JA=(J-1)*J/2 ! JA=(J-1)*NA
      IF(B(J+JA).GT.0.0) THEN
        BB=1.0/DSQRT(B(J+JA))
        DO 10 I=1,N
        V(I+JV)=V(I+JV)*BB
   10   CONTINUE
      ENDIF
   20 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Compute : [V]'[A] [V] ---> [A]                            |    **
C**   +----------------------------------------------------------+    **
      CALL MAV(A,B,A,V,X,N,N,NV)
      CALL MAV(A,B,B,V,X,N,0,NV)
C**   +----------------------------------------------------------+    **
C**   | Compute : [G]'[A] [G] ---> [A];  [V] [G] ---> [V]         |    **
C**   +----------------------------------------------------------+    **
      CALL GAG(A,B,V,N,X,EPSI,NV)
C**   +----------------------------------------------------------+    **
C**   | Solve [A] {X} = 1/d {X}  for  [V] & [D]^{-1}              |    **
C**   +----------------------------------------------------------+    **
      CALL JACOBI(A,V,N,X,IH,-1,M,EPSI,NV)
C**   +----------------------------------------------------------+    **
C**   | Before norm. : [V]'[A] [V] = [D]^{-1}, [V]'[B] [V] = [I]  |    **
C**   | Normalize as : [V]'[A] [V] = [I],      [V]'[B] [V] = [D]  |    **
C**   +----------------------------------------------------------+    **
      DO 95 J=1,N
      JV=(J-1)*NV
      JA=(J-1)*J/2 ! JA=(J-1)*NA
      IF(A(J+JA).EQ.0.0) THEN
        X(J)=1.0D30
      ELSE
```

```
        X(J)=1.0/A(J+JA)
        BB=DSQRT(ABS(X(J)))
        DO 90 I=1,N
        V(I+JV)=V(I+JV)*BB
   90   CONTINUE
        ENDIF
   95 CONTINUE
        RETURN
        END
************************************************************************
        SUBROUTINE MAV(A,B,C,V,X,N,M,NV)
C**     ================================================================    **
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION A(1),B(1),C(1),V(1),X(N)
C**     ================================================================    **
C**     DIMENSION A,B,C(N*(N-1)/2),V(NV,N),X(N) ! A,B,C(NA,N)               **
C**     ----------------------------------------------------------------    **
C**     CALL MAV(A,B,A,V,X,N,N,NV) :     [A] [V] --> [A]                    **
C**     CALL MAV(A,B,B,V,X,N,0,NV) : [V]'[A]'[V] --> [A]                    **
C**     ----------------------------------------------------------------    **
C**     (CT+A) *    V   --> (AT+B) ; B UNUSED IF M=0                        **
C**     ----------------------------------------------------------------    **
C**     A A A     V V V     A B B                                           **
C**     C A A  *  V V V  =  A A B                                           **
C**     C C A     V V V     A A A                                           **
C**     ================================================================    **
        DO 50 I=1,N
C**       +--------------------------------------------------------+    **
C**       | Set <X> = the I-th row of [A]                          |    **
C**       +--------------------------------------------------------+    **
        IA=(I-1)*I/2 ! IA=(I-1)*NA
        DO 10 J=1,I-1
        X(J)=C(J+IA)
   10   CONTINUE
        DO 20 J=I,N
        JA=(J-1)*J/2 ! JA=(J-1)*NA
        X(J)=A(I+JA)
   20   CONTINUE
C**     +--------------------------------------------------------+    **
C**     | Compute left part of the I-th row of ([A][V])          |    **
C**     +--------------------------------------------------------+    **
        DO 30 J=1,I
        JV=(J-1)*NV
        AV=0.0
        DO 25 K=1,N
        AV=AV+X(K)*V(K+JV)
   25   CONTINUE
        A(J+IA)=AV
   30   CONTINUE
C**     +--------------------------------------------------------+    **
C**     | Compute right part of the I-th row of ([A][V])         |    **
C**     +--------------------------------------------------------+    **
        DO 40 J=I+1,M
        JV=(J-1)*NV
        JB=(J-1)*J/2 ! JB=(J-1)*NA
        AV=0.0
        DO 35 K=1,N
        AV=AV+X(K)*V(K+JV)
   35   CONTINUE
        B(I+JB)=AV
```

```
   40   CONTINUE
   50 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE GAG(A,B,V,N,GK,EPSI,NV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),V(1),GK(N)
C**   ============================================================    **
C**   DIMENSION A,B(N*(N-1)/2),V(NV,N),GK(N) ! A,B(NA,N)              **
C**   ------------------------------------------------------------    **
C**   Static condensation : [G]'[A(N,N)] [G] --> [A(N,N)]             **
C**                         [V(N,N)] [G] --> [V(N,N)]             **
C**   ------------------------------------------------------------    **
C**   Input : A,B,V,N,EPSI,NV;   Output : A,V;   Working : GK         **
C**   ============================================================    **
      DO 80 K=1,N
      KA=(K-1)*K/2 ! KA=(K-1)*NA
      IF(B(K+KA).EQ.0.0) THEN
        DO 30 I=1,K                 !     /   1                     \
        GK(I)=A(I+KA)               !     |         1               |
        A(I+KA)=0.0                 !     |             1           |
   30   CONTINUE                    ! [G]=| -A41 -A42 -A43     -A45 |
        DO 40 J=K+1,N               !     | ---- ---- ----  1  ---- |
        JA=(J-1)*J/2 ! JA=(J-1)*NA  !     |  A44  A44  A44      A44 |
        GK(J)=A(K+JA)               !     \                      1  /
        A(K+JA)=0.0
   40   CONTINUE
        A(K+KA)=1.0D30
C**     ------------------------------------------------------------    **
        IF(GK(K).NE.0.0) THEN
          KV=(K-1)*NV
          DO 60 J=1,N
          IF(J.EQ.K) GO TO 60
          GKJ=GK(J)/GK(K)
          JV=(J-1)*NV                 !     / A11 A12 A13  0  A15 \
          JA=(J-1)*J/2 ! JA=(J-1)*NA  !     | A21 A22 A23  0  A25 |
          DO 50 I=1,N                 ! [A]=| A31 A32 A33  0  A35 |
          V(I+JV)=V(I+JV)-V(I+KV)*GKJ !     |  0   0   0   M   0  |
          IF(I.GT.J.OR.I.EQ.K) GO TO 50 !   \ A51 A52 A53  0  A55 /
          A(I+JA)=A(I+JA)-GK(I)*GKJ
   50     CONTINUE
   60     CONTINUE
        ELSE
          DO 70 I=1,N
          IF(DABS(GK(I)).GT.EPSI) STOP 'Not a well posed problem'
   70     CONTINUE
        ENDIF
      ELSE IF(B(K+KA).LT.0.0) THEN
        STOP '[B] is not semi-positive'
      ENDIF
   80 CONTINUE
      RETURN
      END
*************************************************************************
```

[表三]之主程式係用以讀入下列算例中之矩陣$[A]$及$[B]$，然後呼叫副程式$AVEXBV$算出所有之特徵值及特徵向量。程式後為輸入資料及計算結果。

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \lambda \begin{bmatrix} 0.96726 & 0.33482 & 0 & 0 \\ 0.33482 & 1.93452 & 0.33482 & 0 \\ 0 & 0.33482 & 1.93452 & 0.33482 \\ 0 & 0 & 0.33482 & 1.93452 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \lambda \begin{bmatrix} 1.30208 & 0 & 0 & 0 \\ 0 & 2.60417 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.60417 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix}$$

[表三] 一般特徵值用例

```
**************************************************************************
C**   PROGRAM FOR GENERAL EIGENVALUE PROBLEMS
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(210),B(210),V(20,20),X(20),IH(20)
      NMAX=20
C**   ============================================================    **
   10 READ(*,'(I4,F8.6)') N,EPS
      IF(N.EQ.0.OR.N.GT.NMAX) STOP
      READ(*,'(10F8.0)') (A(K),K=1,N*(N+1)/2)
      READ(*,'(10F8.0)') (B(K),K=1,N*(N+1)/2)
      CALL PRINTT(A,N,'Lower part of Matrix A :')
      CALL PRINTT(B,N,'Lower part of Matrix B :')
C**   ------------------------------------------------------------    **
      CALL AVEXBV(A,B,V,N,X,IH,M,EPS,NMAX)
C**   ------------------------------------------------------------    **
      WRITE(*,'(/1X,A,I4,A,$)') 'After',M,' Jacobian iterations,'
      WRITE(*,'(A/)') ' the eigenvalues are'
      WRITE(*,'(1X,1P8E10.3)') (X(I),I=1,N)
      WRITE(6,'(/1X,A/)') 'the eigenvector matrix is'
      DO 75 I=1,N
   75 WRITE(*,'(1X,1P8E10.3)') (V(I,J),J=1,N)
      GO TO 10
      END
**************************************************************************
   4 0.0001
 1.0     1.0     2.0     0.0   1.0     2.0     0.0   0.0   1.0     2.0
 0.96726 0.33482 1.93452 0.0   0.33482 1.93452 0.0   0.0   0.33482 1.93452
   4 0.0001
 1.0     1.0     2.0     0.0   1.0     2.0     0.0   0.0   1.0     2.0
 1.30208 0.0     2.60417 0.0   0.0     0.      0.0   0.0   0.0     2.60417
```

```
========================================================================
Lower part of Matrix A :

 1.000E+00
 1.000E+00 2.000E+00
 0.000E+00 1.000E+00 2.000E+00
 0.000E+00 0.000E+00 1.000E+00 2.000E+00

Lower part of Matrix B :

 9.673E-01
 3.348E-01 1.935E+00
 0.000E+00 3.348E-01 1.935E+00
 0.000E+00 0.000E+00 3.348E-01 1.935E+00

After  30 Jacobian iterations, the eigenvalues are

 1.507E+00 1.262E+00 7.357E-01 1.157E-01

the eigenvector matrix is

 4.425E-01-4.777E-01 5.458E-01-6.164E-01
 4.088E-01-1.828E-01-2.089E-01 5.695E-01
 3.129E-01 3.378E-01-3.860E-01-4.359E-01
 1.693E-01 4.414E-01 5.043E-01 2.359E-01
------------------------------------------------------------------------
Lower part of Matrix A :

 1.000E+00
 1.000E+00 2.000E+00
 0.000E+00 1.000E+00 2.000E+00
 0.000E+00 0.000E+00 1.000E+00 2.000E+00

Lower part of Matrix B :

 1.302E+00
 0.000E+00 2.604E+00
 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 0.000E+00 0.000E+00 2.604E+00

After   8 Jacobian iterations, the eigenvalues are

 1.247E+00 5.971E-01 1.000E+30 7.606E-02

the eigenvector matrix is

 6.459E-01-2.874E-01 0.000E+00-5.179E-01
 4.027E-01 6.396E-02 0.000E+00 4.666E-01
-1.437E-01 2.590E-01 1.000E+00-3.229E-01
-1.153E-01-5.819E-01 0.000E+00 1.792E-01
========================================================================
```

6.12 副程式 $SUBSPC$

本程式係利用次空間法計算

$$[K]\{X\} = \omega^2 [M]\{X\}$$

之 MT 個較小 ω^2 值及對應前 MT 個特徵向量 $V(N,MT)$。

$LL(N)$	=	已知之變寬帶勁度矩陣之位置指標。
$AKA(LL(N)+N-1)$	=	變寬帶勁度矩陣 $[K]$，運算後為 $[L]$。
$AMA(LL(N)+N-1)$	=	非對角質量矩陣 $[M]$，其儲存方式與 AKA 者同，須令 ID=1。或
$AMA(N)$	=	質量矩陣 $[M]$，為對角矩陣，須令 ID=0。
$AV(N,MT)$	=	求得之前 MT 個特徵向量。
$W(MT)$	=	求得之前 MT 個由小至大排列的較小 ω_i^2。
$VV(N)$	=	臨時運算位置。
$SKS(MT*(MT-1)/2)$	=	臨時運算位置，存 $[\bar{K}]$ 用。
$SMS(MT*(MT-1)/2)$	=	臨時運算位置，存 $[\bar{M}]$ 用。
$SV(MT,MT)$	=	臨時運算位置，存 $[\Psi]$ 用。
$Y(MT)$	=	求得之前 MT 個較小 ω_i^2 之前一次近似值。
ID	=	質量矩陣型式指標：ID=0 為對角矩陣；ID=1 為非對角矩陣。
$NMAX$	=	次空間法反覆運算之最高允許次數。
EPS	=	判斷 $W(MT)$ 是否收斂用。若 $\lvert W(I)-Y(I)\rvert \le W(I)*EPS$，$I=1,2,\ldots,MW$，則認為 MW 個較小 ω_i^2 均已收斂，而可停止運算。
N	=	矩陣 AKA 之階數。

［表四］ 變寬帶矩陣特徵值問題-次空間法

```
************************************************************************
      SUBROUTINE SUBSPC
     * (LL,AKA,AMA,AV,W,N,MT,MW,VV,SKS,SMS,SV,Y,ID,NMAX,EPS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION LL(N),AKA(1),AMA(1),AV(1),W(MT)
      DIMENSION VV(N),SKS(1),SMS(1),SV(1),Y(MT)
C**   ============================================================    **
C*I   LL(N)     = Input variable band matrix stored index             **
C*I   AKA(NN)   = Input stiffness matrix [K]                          **
C*I   AMA(NN)   = Input mass matrix [M] (if id=1)                     **
```

```
C*I   AMA(N)    = Input diagonal mass matrix [M] (if id=0)           **
C*O   AV(N,MT)  = Output eigenvectors [V]                           **
C*O   W(MT)     = Eigenvalues (omaga squares)                       **
C*W   VV(N)     = Working array                                     **
C*W   SKS(MT*(MT+1)/2) = Working arry to store [V]'[K] [V]          **
C*W   SMS(MT*(MT+1)/2) = Working arry to store [V]'[M] [V]          **
C*W   SV(MT,MT) = Working array                                     **
C*O   Y(MT)     = Working array                                     **
C*I   N         = Order of AKA and AMA                              **
C*I   MT        = Total no. of eigenvectors used in iteraton        **
C*I   MW        = Reqd. no. of eigenvectors                         **
C*I   ID        = 0, AMA is a diagonal matrix                       **
C*I             = 1, AMA is stored as AKA                           **
C*I   NMAX      = Allowed max. no. of subspace iterations           **
C*I   EPS       = Tolerence of eigenvales W(MW)                     **
C*I   NN        = LL(N)+N-1                                         **
C**   ================================================================**
C**   +----------------------------------------------------------+  **
C**   | Initialize                                               |  **
C**   +----------------------------------------------------------+  **
      NPT=1
      M=MT
      MMM=NMAX
      NNN=0
      KKK=0
C**   +----------------------------------------------------------+  **
C**   | Decompose [K]=[L]'[D] [L]; (1) AV=[Uo], {W}={0}          |  **
C**   +----------------------------------------------------------+  **
      CALL VBDECP(AKA,LL,N)
      CALL SETMVO(LL,AKA,AMA,AV,N,M,ID,VV)
      DO 20 J=1,M
   20 W(J)=0.0
C**   +----------------------------------------------------------+  **
C**   | (2) AV=[V1]=inv.[K][Uo]; (3) SKS=[V1]'[Uo]=[V1]'[K] [V1] |  **
C**   +----------------------------------------------------------+  **
   30 NNN=NNN+1
      DO 50 J=1,M
      Y(J)=W(J)
      JV=(J-1)*N
      DO 40 I=1,N
   40 VV(I)=AV(I+JV)
      CALL VBSOLX(AKA,AV(1+JV),LL,N)
   50 CALL CMPYAT(AV,VV,SKS((J-1)*J/2+1),J,N)
      IF(NNN/NPT*NPT.EQ.NNN.OR.NNN.EQ.MMM) THEN
        WRITE(*,'(/1X,A,I4)') 'Subspace iteration No.',NNN
        CALL PRINTT(SKS,M,'Matrix SKS')
      ENDIF
C**   +----------------------------------------------------------+  **
C**   | (4) If NNN<MMM then AV=[U1]=[M][V1] else AV=[V1]          |  **
C**   | (5) SMS=[U1]'[V1]=[V1]'[M] [V1]                          |  **
C**   +----------------------------------------------------------+  **
      DO 80 J=1,M
      JV=(J-1)*N
      IF(NNN.LT.MMM) THEN
        DO 70 I=1,N
        VV(I)=AV(I+JV)
   70   CONTINUE
        CALL VBCMPY(AMA,VV,AV(1+JV),LL,N,ID)
      ELSE
        CALL VBCMPY(AMA,AV(1+JV),VV,LL,N,ID)
```

```
      ENDIF
      CALL CMPYAT(AV,VV,SMS((J-1)*J/2+1),J,N)
   80 CONTINUE
      IF(NNN/NPT*NPT.EQ.NNN.OR.NNN.EQ.MMM) THEN
        CALL PRINTT(SMS,M,'Matrix SMS')
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | (6) Solve [SKS] [SV] = [SMS] [SV] [W]                  |    **
C**   | (7) If NNN<MMM then AV=[U2]=[U1][SV] else AV=[V2]=[V1][SV]    **
C**   +--------------------------------------------------------+    **
   90 CALL BVEXAV(SMS,SKS,SV,M,W,VV,KKK,1.0E-5,M)
      CALL ORDERX(W,SV,VV,M)
      CALL AMPY(AV,SV,VV,N,M)
C**   +--------------------------------------------------------+    **
C**   | (8) Convergence test (If so, set MMM=0 for last iter.) |    **
C**   +--------------------------------------------------------+    **
      IF(NNN.LT.MMM) THEN
        DO 120 J=1,MW
        IF(DABS(Y(J)-W(J)).GT.EPS*W(J)) GO TO 30
  120   CONTINUE
        MMM=0
        GO TO 30
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | (9) Output the eigenvalues and return                  |    **
C**   +--------------------------------------------------------+    **
      IF(NNN.EQ.MMM) WRITE(*,'(/A,1PE11.4)') ' Not converged within',EPS
      WRITE(*,'(/1X,A,I4,A)') 'Subspace performs',NNN,' iterations.'
      WRITE(*,'(1X,A,I4,A)')  'Jacobian performs',KKK,' rotations.'
      WRITE(*,'(1X,A)') 'The eigenvalues of the last two iterartions :'
      WRITE(*,'(/(1X,1P10E13.6))') (Y(J),J=1,M)
      WRITE(*,'(/(1X,1P10E13.6))') (W(J),J=1,M)
      RETURN
      END
*************************************************************************
      SUBROUTINE SETMVO(LL,AKA,AMA,AV,N,M,ID,VV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION LL(1),AKA(1),AMA(1),AV(11),VV(1)
      DATA IX/1/
C**   ============================================================    **
      DO 15 J=1,M
      JV=(J-1)*N
      DO 10 I=1,N
   10 AV(I+JV)=RANDOM(IX)
   15 AV(J+JV)=AV(J+JV)+1.0
      RETURN
      END
*************************************************************************
      FUNCTION RANDOM(IX)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C**   Random number by linear congruential method
C**   For 32 bits Integer*4 51603,2147483647
C**   For 16 bits Integer*2 403,32767
C**   For 1's complement, cancel +1
C**   ============================================================    **
      IX=MOD(403*IX+3,32768)
C**   IF(IX.LT.0) IX=IX+32767+1
```

```
   20 RANDOM=IX/32767.0
      RETURN
      END
*************************************************************************
      SUBROUTINE VBCMPY(A,X,Y,LL,N,ID)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),X(1),Y(1),LL(1)
C**   ================================================================    **
C**   [A(N,N)] {X(N)} --> {Y(N)}  (ID=0: A[N] diagonal)                   **
C**   ================================================================    **
      IF(ID.EQ.0) THEN
        DO 50 I=1,N
   50   Y(I)=A(I)*X(I)
        RETURN
      ENDIF
      JA=0
      DO 20 J=1,N
      YJ=0.0
      XJ=X(J)
      JM=J-(LL(J)-LL(1)-JA)
      JA=LL(J)-LL(1)
      DO 10 I=JM,J-1
      Y(I)=Y(I)+A(I+JA)*XJ
   10 YJ=YJ+A(I+JA)*X(I)
   20 Y(J)=YJ+A(J+JA)*XJ
      RETURN
      END
*************************************************************************
      SUBROUTINE CMPYAT(A,B,C,M,N)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),C(1)
C**   ================================================================    **
C**   [A(N,M)]'[B(N)] --> [C(M)]                                          **
C**   ================================================================    **
      DO 20 J=1,M
      JA=(J-1)*N
      CJ=0.0
      DO 10 I=1,N
   10 CJ=CJ+A(I+JA)*B(I)
   20 C(J)=CJ
      RETURN
      END
*************************************************************************
      SUBROUTINE AMPY(A,B,AI,N,M)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),AI(1)
C**   ================================================================    **
C**   [A(N,M)] [B(M,M)] --> [A(N,M)]                                      **
C**   ================================================================    **
      DO 30 I=1,N
      DO 10 K=1,M
      AI(K)=A(I+(K-1)*N)
   10 CONTINUE
      DO 30 J=1,M
      JA=(J-1)*N
      JB=(J-1)*M
      AIJ=0.0
```

```
      DO 20 K=1,M
      AIJ=AIJ+AI(K)*B(K+JB)
   20 CONTINUE
      A(I+JA)=AIJ
   30 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE ORDERX(X,V,L,N)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(1),V(1),L(1)
C**   ============================================================    **
C**   Order V(N,1:N) & X(1:N) to decreasing X(1:N)                    **
C**   ============================================================    **
      DO 10 I=1,N
   10 L(I)=I
      M=N
C**   ------------------------------------------------------------    **
   20 NE=M-1
      DO 50 I=1,NE
      J=L(I)
      K=L(I+1)
      IF(X(J).LE.X(K)) GO TO 50
        L(I)=K
        L(I+1)=J
        M=I
   50 CONTINUE
      IF(M.LE.NE) GO TO 20
C**   ------------------------------------------------------------    **
      DO 90 J=1,N-1
      I=J
   70 I=L(I)
      IF(I-J) 70,90,80
   80 II=(I-1)*N
      JJ=(J-1)*N
      T=X(I)
      X(I)=X(J)
      X(J)=T
      DO 85 KK=1,N
      T=V(KK+II)
      V(KK+II)=V(KK+JJ)
   85 V(KK+JJ)=T
   90 CONTINUE
      RETURN
      END
*************************************************************************
```

[表五]主程式用以讀入前節範例之AKA與AMA矩陣，然後呼叫副程式$SUBSPC$算出前2個特徵值及特徵向量。利用3個向量做次空間法運算，程式後為輸入資料及計算結果(例二結果省略)。

[表五] 變寬帶矩陣特徵值問題用例

```
*************************************************************************
      PROGRAM SUBSMN
C**   ============================================================    **
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(5000),LL(5000)
      EQUIVALENCE (A,LL)
      NDIM=5001
C**   ================================================================    **
   10 READ(5,'(5I4,F8.6)') N,MW,MT,ID,NMAX,EPS
      IF(N.EQ.0) STOP
      IF(MT.EQ.0) MT=MIN0(2*MW,MW+5,N)
      IF(NMAX.EQ.0) NMAX=20
      IF(EPS.EQ.0.0) EPS=0.000001
      READ(*,'(20I4)') (LL(J),J=1,N)
      NN=LL(N)+N-LL(1)
C**   +--------------------------------------------------------------+    **
C**   | Dynamic allocates the arrays                                 |    **
C**   +--------------------------------------------------------------+    **
      N1=1
      N2=N1+N                ! LL(N)
      N3=N2+NN               ! AKA(NN)
      N4=N3+NN*ID+N*(1-ID)   ! AMA(NN) or AMA(N)
      N5=N4+N*MT             ! AV(N,MT)
      N6=N5+MT               ! W(MT)
      N7=N6+N                ! VV(N)
      N8=N7+MT*(MT+1)/2      ! SKS(MT*(MT+1)/2)
      N9=N8+MT*(MT+1)/2      ! SMS(MT*(MT+1)/2)
      N10=N9+MT*MT           ! SV(MT,MT)
      N11=N10+MT             ! Y(MT)
      IF(N11.GT.NDIM) STOP
C**   +--------------------------------------------------------------+    **
C**   | Input AKA(N,N),AMA(N,N)                                      |    **
C**   | Compute and Output AV(N,MW),W(MW)                            |    **
C**   +--------------------------------------------------------------+    **
      CALL INPUT(A(N1),A(N2),A(N3),N,ID)
      CALL SUBSPC(A(N1),A(N2),A(N3),A(N4),A(N5),N,MT,MW
     *           ,A(N6),A(N7),A(N8),A(N9),A(N10),ID,NMAX,EPS)
      CALL OUTPUT(A(N4),A(N5),N,MW)
      GO TO 10
      END
*************************************************************************
      SUBROUTINE INPUT(LL,AKA,AMA,N,ID)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION LL(N),AKA(1),AMA(1)
C**   ================================================================    **
C**   READ(*,'(20I4)') (LL(J),J=1,N)
      NN=LL(N)+N-LL(1)
      READ(*,'(10F8.0)') (AKA(K),K=1,NN)
      NN=NN*ID+N*(1-ID)
      READ(*,'(10F8.0)') (AMA(K),K=1,NN)
      RETURN
      END
*************************************************************************
      SUBROUTINE OUTPUT(AV,W,N,MW)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AV(1),W(MW)
C**   ================================================================    **
C**   Output eigenvalues W(MW) and eigenvectors A(N,MW)                   **
C**   ================================================================    **
      DO 10 J=1,MW
   10 WRITE(*,25) W(J),(AV(I+(J-1)*N),I=1,N)
```

```
      RETURN
   25 FORMAT(/' Eigenvalue =',1PE11.4,', Eigenvector :'//(1X,10E11.4))
      END
*************************************************************************
   4   2   3   1  15 .0001
   1   2   3   4
      1.      1.      2.      1.      2.      1.      2.
 0.96726 0.33482 1.93452 0.33482 1.93452 0.33482 1.93452
   4   2   3   0  15 .0001
   1   2   3   4
      1.      1.      2.      1.      2.      1.      2.
 1.30208 2.60417 0.      2.60417

=========================================================================

Subspace iteration No.   1
Matrix SKS                          Matrix SMS

 1.207E+00                           1.250E+00
 1.165E-01 1.270E+00                -1.013E+00 5.203E+00
 1.117E+00-1.318E+00 8.105E+00       3.897E+00-1.639E+01 6.381E+01

Subspace iteration No.   2
Matrix SKS                          Matrix SMS

 8.643E+00                           7.470E+01
-3.806E-03 1.347E+00                -4.107E-02 1.822E+00
 1.104E-03-2.406E-03 6.679E-01       1.118E-02-6.778E-03 4.466E-01

Subspace iteration No.   3
Matrix SKS                          Matrix SMS

 8.643E+00                           7.471E+01
-2.081E-04 1.355E+00                -2.245E-03 1.839E+00
 1.193E-04-1.665E-03 6.696E-01       1.206E-03-4.686E-03 4.491E-01

Subspace iteration No.   4
Matrix SKS                          Matrix SMS

 8.643E+00                           7.471E+01
-1.109E-05 1.358E+00                -1.196E-04 1.845E+00
 1.274E-05-1.135E-03 6.720E-01       1.287E-04-3.196E-03 4.527E-01

Subspace iteration No.   5
Matrix SKS                          Matrix SMS

 8.643E+00                           7.471E+01
-5.833E-07 1.359E+00                -6.291E-06 1.847E+00
 1.345E-06-7.652E-04 6.753E-01       1.358E-05-2.154E-03 4.574E-01

Subspace iteration No.   6
Matrix SKS                          Matrix SMS

 8.643E+00                           7.471E+01
-3.038E-08 1.359E+00                -3.276E-07 1.847E+00
 1.398E-07-5.087E-04 6.797E-01       1.412E-06-1.432E-03 4.638E-01

Subspace iteration No.   7
Matrix SKS                          Matrix SMS
```

```
 8.643E+00                              7.471E+01
-1.565E-09 1.359E+00                   -1.687E-08 1.848E+00
 1.427E-08-3.323E-04 6.854E-01          1.441E-07-9.354E-04 4.721E-01

Subspace iteration No.   8
Matrix SKS                             Matrix SMS

 8.643E+00                              7.471E+01
-7.955E-11 1.359E+00                   -8.567E-10 1.848E+00
 1.421E-09-2.122E-04 6.926E-01          1.435E-08-5.975E-04 4.826E-01

Subspace performs   8 iterations.     Jacobian performs  20 rotations.
The eigenvalues of the last two iterartions :

 1.156975E-01 7.356775E-01 1.451798E+00

 1.156975E-01 7.356674E-01 1.435195E+00

Eigenvalue = 1.1570E-01, Eigenvector :

 6.1643E-01-5.6950E-01 4.3588E-01-2.3590E-01

Eigenvalue = 7.3567E-01, Eigenvector :

 5.4657E-01-2.0887E-01-3.8721E-01 5.0291E-01
==========================================================================
```

習題

1. 計算下列特徵方程式之特徵值及特徵向量。

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \lambda \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix}$$

2. 簡化副程式$AVEXBV$。將其$[B]$矩陣改為對角矩陣，而僅用一向量儲存其對角線元素，假設對角元素皆大於零。

第七章

一般矩陣之特徵值問題

7.1 前言

特徵值問題從相關矩陣之個數可分為：

(1)標準特徵值問題：$[A]\{X\} = \lambda\{X\}$ 。

(2)一般特徵值問題：$[A]\{X\} = \lambda[B]\{X\}$ 。

(3) λ 型特徵值問題：$([C_r]\lambda^r - \cdots - [C_2]\lambda^2 - [C_1]\lambda - [C_o])\{X\} = \{0\}$ 。

λ 型可按如下方式改為一般型，以 $r = 3$ 為例，注意由第三與第二組等式可知其中：$\{X_1\} = \lambda\{X_o\}$ ，$\{X_2\} = \lambda\{X_1\} = \lambda^2\{X_o\}$ 。

$$\begin{bmatrix} [C_2] & [C_1] & [C_o] \\ [I] & [0] & [0] \\ [0] & [I] & [0] \end{bmatrix} \begin{Bmatrix} \{X_2\} \\ \{X_1\} \\ \{X_o\} \end{Bmatrix} = \lambda \begin{bmatrix} [C_3] & [0] & [0] \\ [0] & [I] & [0] \\ [0] & [0] & [I] \end{bmatrix} \begin{Bmatrix} \{X_2\} \\ \{X_1\} \\ \{X_o\} \end{Bmatrix} \tag{7.1}$$

一般型之 $[A]$ 或 $[B]$ 如至少有一個為非奇異矩陣，則可由下列方式之一改為標準型：(1) $[B]^{-1}[A]\{X\} = \lambda\{X\}$ ；或(2) $[A]^{-1}[B]\{X\} = \lambda^{-1}\{X\}$ 。但如 $[A]$ 與 $[B]$ 均為對稱矩陣時，此方式不能保住其對稱特性。又如 $[A]$ 與 $[B]$ 均為奇異矩陣時，亦不能採用。因此本章另提供一種間接解法，以做二次標準特徵值問題而求得一般型之解。該法不但可解 $[A]$ 與 $[B]$ 均為奇異之問題，亦可保住對稱矩陣之對稱性，效率也較佳，故亦用於第六章之對稱矩陣之特徵值問題。特徵值問題之解法為數頗多，本章將僅介紹常用而有效之基本解法，並提供四套程式如[表一]至[表四]所示。亦可參考著名之套裝程式集 $EISPACK$ 。

7.2 複數係數矩陣之處理

上述三種類型之特徵值問題其矩陣係數又可分為實數與複數二種。雖然很多方法同時適用於實數與複數，但大部分程式僅針對實數而寫，因此有必要將複數係數之特徵值問題改為如下之實數係數之特徵值問題(λ型類同)，則向量$\{X\}=\{U\}+i\{V\}$，與特徵值λ會滿足原特徵方程式(因由下式可得$[A_r]\{U\}-[A_i]\{V\}=\lambda\{U\}$與$[A_i]\{U\}+[A_r]\{V\}=\lambda\{V\}$，後式乘$i$與前式相加即得原式：$([A_r]+i\,[A_i])\{X\}=\lambda\,\{X\}$，另二型類同)，故若$\{U\}+i\{V\}\neq\{0\}$，此向量即為原式之特徵向量。

$$\begin{bmatrix} [A_r] & -[A_i] \\ [A_i] & [A_r] \end{bmatrix} \begin{Bmatrix} \{U\} \\ \{V\} \end{Bmatrix} = \lambda \begin{Bmatrix} \{U\} \\ \{V\} \end{Bmatrix} \tag{7.2}$$

$$\begin{bmatrix} [A_r] & -[A_i] \\ [A_i] & [A_r] \end{bmatrix} \begin{Bmatrix} \{U\} \\ \{V\} \end{Bmatrix} = \lambda \begin{bmatrix} [B_r] & -[B_i] \\ [B_i] & [B_r] \end{bmatrix} \begin{Bmatrix} \{U\} \\ \{V\} \end{Bmatrix} \tag{7.3}$$

上述問題之形式特殊，故有下列二項特性：

(1)若$\{\{U\},\{V\}\}$為特徵值λ之特徵向量，則$\{-\{V\},\{U\}\}$亦為特徵值λ之特徵向量。此關係可直接代入上式証知。

(2)若$\{\{U\},\{V\}\}$為特徵值λ之特徵向量，則其共軛向量$\{\{\bar{U}\},\{\bar{V}\}\}$為其共軛特徵值$\bar{\lambda}$之特徵向量。此亦為所有實係數矩陣之特性。此關係可由上式兩邊取共軛証知。

注意新特徵值問題有$2N$組特徵值與特徵向量，為原問題之二倍，可見有一半為重複或多餘，以下說明如何分辨出原問題之特徵值與特徵向量：

(1)若原問題之某一特徵值為實數，則新問題之特徵向量$\{\{U\},\{V\}\}$必為實數，且由上述特性(1)知另一組特徵向量為$\{-\{V\},\{U\}\}$，而這二組向量為互相獨立且相互正交，因此新問題之對應特徵值為二重根。但其對應原問題之特徵向量$\{X_1\}=\{U\}+i\{V\}$與$\{X_2\}=-\{V\}+i\{U\}$僅差一比例常數i，即$\{X_2\}=i\{X_1\}$。

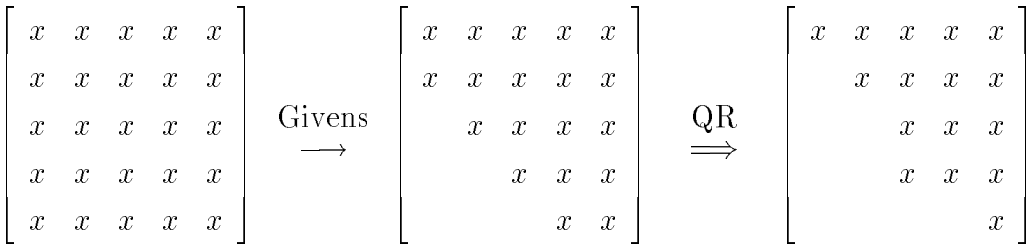
(2)若原問題之某一特徵值為複數，則新問題除該特徵值外另有一特徵值與其互為共軛，由特性(2)知其對應之二特徵向量亦互為共軛。而其對應原問題之特徵向量可分別由$\{X_1\}=\{U\}+i\{V\}$與$\{X_2\}=\{\bar{U}\}+i\{\bar{V}\}$求得，但其中會有一個$\{X\}$等於零，則其所對應之新問題之特徵值與特徵向量即為多餘的。因$\{X\}=\{0\}$雖滿足原特徵方程式，但因全為零，而不能算是其特徵向量。

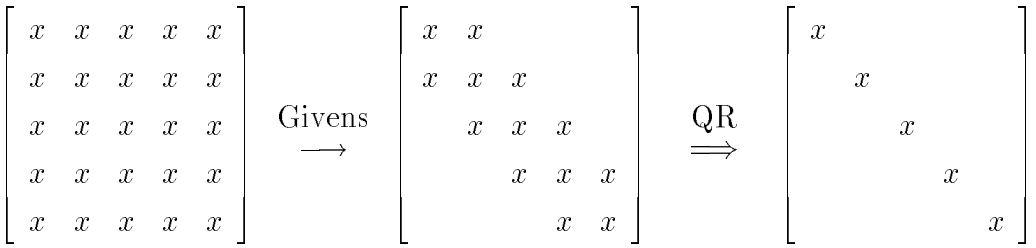
7.3 解標準特徵值問題之二階段

解標準特徵值問題時，通常會將問題分為如下之二階段：

(1)第一段先將矩陣轉成赫申伯格(Hessenberg)矩陣或三對角矩陣；

(2)第二段再將矩陣轉成準上三角矩陣或對角矩陣。





其原因可由下列數點看出：

(1)第一段僅用一回合的相似轉換即可達成。

(2)第二段理論上須用無限回合的相似轉換才可達成。

(3)第二段的轉換過程中每一回合之相似矩陣均保持赫申伯格矩陣或三對角矩陣之型式，僅其上次對角元素會逐漸趨近於零。

(4)前項每一回合之運算量均為有限，而正比於n^3、n^2或n：第一段每一回合之運算量與n^3成正比；第二段每一回合之運算量，不對稱矩陣與n^2成正比，對稱矩陣與n成正比。

(5)如不分二段而直接由滿矩陣轉換為準上三角矩陣或對角矩陣時，每一回合之運算量亦與n^3成正比，而此運算回合理論上亦為無限次，顯然運算效率比分二段者差很多。

7.4 相似轉換

若$[P]$為非奇異方陣，則$[\bar{A}]=[P]^{-1}[A][P]$與$[A]$之特徵值相同，其特徵向量矩陣$[\bar{X}]=[P]^{-1}[X]$，稱$[\bar{A}]$與$[A]$相似，$[P]$為轉換矩陣。若$[A]$為對稱，則一般亦希望$[\bar{A}]$對稱，因此須$[P]^{-1}=[P]^T$，即矩陣$[P]$之逆矩陣與轉置矩陣相等，稱為正交矩陣，一般以$[Q]$代替$[P]$表示轉換矩陣為正交矩陣，則對稱相似矩陣寫成$[\bar{A}]=[Q]^T[A][Q]$。

不對稱矩陣之轉換矩陣可以不用正交矩陣，但是為了數值之穩定，亦常採用正交矩陣。常用之正交矩陣基本上有二種類型：(1)Givens型；(2)Householder型。Givens型可用於上節之二段中；而Householder型較常用於其中之第一段。以下先說明Givens型在二段中之運算方式：
Givens型為如下型式(以n=6, i=4, j=2為例)之正交矩陣，記成$[P_{ik}^j]$，式中$c=\cos\theta$, $s=\sin\theta$，若θ由下式求得，則$[A]$之前乘以$[P_{ik}^j]$可使$\bar{a}_{ik}=0$，稱為以a_{jk}做為配對消去a_{ik}。

$$\bar{a}_{ik}=a_{ik}\cos\theta-a_{jk}\sin\theta=0 \tag{7.4}$$

$$[P_{4k}^2]=\begin{bmatrix} 1 & & & & \\ & c & & s & \\ & & 1 & & \\ & -s & & c & \\ & & & & 1 \end{bmatrix} \tag{7.5}$$

第一段之相似轉換，係依$k=1,\cdots,n-2$之順序將第k行之下次對角以下之元素消去，即得上赫申伯格矩陣或三對角矩陣，以下為消去第2行之情形。x之下標為消去之順序。$'x$元素表示受前乘矩陣之影響，x'元素表示受後乘矩陣之影響，$'x'$元素則受二者之影響。

$$[P_{42}^3][P_{52}^4]\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & x_2 & x & x & x \\ & x_1 & x & x & x \end{bmatrix}[P_{52}^4]^T[P_{42}^3]^T=\begin{bmatrix} x & x & x' & x' & x' \\ x & x & x' & x' & x' \\ & 'x & 'x' & 'x' & 'x' \\ & 0 & 'x' & 'x' & 'x' \\ & 0 & 'x' & 'x' & 'x' \end{bmatrix} \tag{7.6}$$

第二段之相似轉換，係先將上赫申伯格矩陣做如下之QR分解，其中$[Q]^T = [P^4_{54}][P^3_{43}][P^2_{32}][P^1_{21}]$，即下式為$[Q]^T[A] = [R]$，故得$[A] = [Q][R]$。

$$[P^4_{54}][P^3_{43}][P^2_{32}][P^1_{21}]\begin{bmatrix} x & x & x & x & x \\ x_1 & x & x & x & x \\ & x_2 & x & x & x \\ & & x_3 & x & x \\ & & & x_4 & x \end{bmatrix} = \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix} \tag{7.7}$$

再由下式計算$[R][Q]$等於$[\bar{A}]$，因此$[\bar{A}] = [R][Q] = [Q]^T[A][Q]$，即為$[A]$之相似轉換。如此即為一回合之運算。

$$\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix}[P^1_{21}]^T[P^2_{32}]^T[P^3_{43}]^T[P^4_{54}]^T = \begin{bmatrix} x & x & x & x & x \\ x_1 & x & x & x & x \\ & x_2 & x & x & x \\ & & x_3 & x & x \\ & & & x_4 & x \end{bmatrix} \tag{7.8}$$

第一段之相似轉換，亦可依$k = n, n-1, \cdots, 3$之順序將第k行之上次對角以上之元素消去，而得下赫申伯格矩陣或三對角矩陣，以下為消去第4行之情形。

$$[P^3_{24}][P^2_{14}]\begin{bmatrix} x & x & x & x_1 & \\ x & x & x & x_2 & \\ x & x & x & x & \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}[P^2_{14}]^T[P^3_{24}]^T = \begin{bmatrix} 'x' & 'x' & 'x' & 0 & \\ 'x' & 'x' & 'x' & 0 & \\ 'x' & 'x' & 'x' & 'x & \\ x' & x' & x' & x & x \\ x' & x' & x' & x & x \end{bmatrix} \tag{7.9}$$

對應之第二段之相似轉換，則將下赫申伯格矩陣做如下之QL分解，其中$[Q]^T = [P^2_{12}][P^3_{23}][P^4_{34}][P^5_{45}]$，即下式為$[Q]^T[A] = [L]$，故得$[A] = [Q][L]$。

$$[P^2_{12}][P^3_{23}][P^4_{34}][P^5_{45}]\begin{bmatrix} x & x_4 & & & \\ x & x & x_3 & & \\ x & x & x & x_2 & \\ x & x & x & x & x_1 \\ x & x & x & x & x \end{bmatrix} = \begin{bmatrix} x & & & & \\ x & x & & & \\ x & x & x & & \\ x & x & x & x & \\ x & x & x & x & x \end{bmatrix} \tag{7.10}$$

再由下式計算$[L][Q]$等於$[\bar{A}]$，因此$[\bar{A}]=[L][Q]=[Q]^T[A][Q]$，即為$[A]$之相似轉換。如此即完成一回合之運算。

$$\begin{bmatrix} x & & & & \\ x & x & & & \\ x & x & x & & \\ x & x & x & x & \\ x & x & x & x & x \end{bmatrix} [P_{45}^5]^T[P_{34}^4]^T[P_{23}^3]^T[P_{12}^2]^T = \begin{bmatrix} x & x_4 & & & \\ x & x & x_3 & & \\ x & x & x & x_2 & \\ x & x & x & x & x_1 \\ x & x & x & x & x \end{bmatrix} \tag{7.11}$$

以下介紹Householder型在第一段之運算方式：
Householder型為如下型式(以n=5,k=2為例)之正交矩陣，即$[P_k]=[I]-2\{W_k\}\langle W_k\rangle$，式中$\{W_k\}$為單位向量，即$\langle W_2\rangle\{W_2\}=w_{32}^2+w_{42}^2+w_{52}^2=1$。為簡化符號，以後將省略$W_k$之下標$k$。則由$[P_k][P_k]^T=([I]-2\{W\}\langle W\rangle)([I]-2\{W\}\langle W\rangle)=[I]-4\{W\}\langle W\rangle+4\{W\}(\langle W\rangle\{W\})\langle W\rangle=[I]$，可証知$[P_k]$為正交矩陣，注意$[P_k]$亦為對稱。

$$[P_2]=\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} - 2\begin{Bmatrix} 0 \\ 0 \\ w_{32} \\ w_{42} \\ w_{52} \end{Bmatrix} \langle\ 0 \quad 0 \quad w_{32} \quad w_{42} \quad w_{52}\ \rangle \tag{7.12}$$

上式之$\{W\}$可任取為$\{W\}=\{U\}/|\{U\}|$，如取$\{U\}=\{0,\,0,\,a_{32}{+}s,\,a_{42},\,a_{52}\}$，其中$s^2=a_{32}^2{+}a_{42}^2{+}a_{52}^2$，取$s$與$a_{32}$同號（否則$a_{32}+s$有可能近值相減而產生較大之誤差），則$|\{U\}|^2=2s(a_{32}{+}s)$，故上式可寫成：

$$[P_2]=\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} - \frac{2}{2s(a_{32}{+}s)}\begin{Bmatrix} 0 \\ 0 \\ a_{32}{+}s \\ a_{42} \\ a_{52} \end{Bmatrix} \langle\ 0 \quad 0 \quad a_{32}{+}s \quad a_{42} \quad a_{52}\ \rangle \tag{7.13}$$

注意$[P_k][A]$之前k列與$[A]$者相同，而其第k行為：$\{a_{12},\,a_{22},\,a_{32},\,a_{42},\,a_{52}\}-\{0,\,0,\,a_{32}{+}s,\,a_{42},\,a_{52}\}=\{a_{12},\,a_{22},\,-s,\,0,\,0\}$。又$[P_k][A]$後乘$[P_k]$亦不改前$k$

行之值。故得$[\bar{A}]=[P_2][A][P_2]$中第2行之下次對角元素以下之值均為零。

$$[P_2]\left[\begin{array}{cc|ccc} a_{11} & a_{12} & x & x & x \\ a_{21} & a_{22} & x & x & x \\ \hline & a_{32} & x & x & x \\ & a_{42} & x & x & x \\ & a_{52} & x & x & x \end{array}\right][P_2]=\left[\begin{array}{cc|ccc} a_{11} & a_{12} & x' & x' & x' \\ a_{21} & a_{22} & x' & x' & x' \\ \hline & -s & 'x' & 'x' & 'x' \\ & 0 & 'x' & 'x' & 'x' \\ & 0 & 'x' & 'x' & 'x' \end{array}\right] \tag{7.14}$$

若$[A]$對稱，可由式(7.15)至式(7.18)先計算$\{P\},\ K,\ \{Q\}$，再計算$[\bar{A}]$。

$$\begin{aligned} [\bar{A}] &= [P_2][A][P_2]=([I]-\frac{2}{|\{U\}|^2}\{U\}\langle U\rangle)\,[A]\,([I]-\frac{2}{|\{U\}|^2}\{U\}\langle U\rangle) \\ &= [A]-\{U\}\langle Q\rangle-\{Q\}\langle U\rangle \qquad (7.15) \\ \{P\} &= \frac{2}{|\{U\}|^2}\,[A]\{U\} \qquad (7.16) \\ K &= \frac{1}{|\{U\}|^2}\,\langle U\rangle\{P\} \qquad (7.17) \\ \{Q\} &= \{P\}-K\{U\} \qquad (7.18) \end{aligned}$$

Householder型亦可用於第二段：例如取$\{U\}=\{0,\ a_{22}+s,\ a_{32},\ a_{42},\ a_{52}\}$，其中$s^2=a_{22}^2+a_{32}^2+a_{42}^2+a_{52}^2$，可用以消去$[P_2][A]$第2行對角以下之元素。

7.5 相似轉換之探討

注意用於第一段之式(7.6)與用於第二段之式(7.7)之轉換有一點不同，為前者用次對角元素做配對，而後者用主對角元素做配對。用次對角元素做配對時，其對應之後乘矩陣不會影響消去行之各值。用主對角元素做配對時，其對應之後乘矩陣則會影響消去行之值，因此某元素之值雖經前乘矩陣消去，但於後乘矩陣運算後又會變成非零元素，只是其絕對值會愈變愈小，這也就是第二段須要做無限多回合運算的原因。另外注意式(7.6)中之前乘矩陣與後乘矩陣可以交替運算，即$([P_{42}^3]([P_{52}^4][A][P_{52}^4]^T)[P_{42}^3]^T)$與$(([P_{42}^3][P_{52}^4][A])[P_{52}^4]^T[P_{42}^3]^T)$二種運算順序均可。而式(7.7)與式(7.8)分列之原因主要在強調：前乘矩陣須先做完後，再做後乘矩陣之運算。若按$([P_{54}^4]([P_{43}^3]([P_{32}^2]([P_{21}^1][A][P_{21}^1]^T)[P_{32}^2]^T)[P_{43}^3]^T)[P_{54}^4]^T)$之順序運算，雖然其最後結果會相同，但中間過程會有下次對角元素變為非零之值，運算較不便。注意第六章之賈柯比法以對角元素做配對亦有類似現象。

第二段每一回合之運算，以QL法為例，將使上次對角元素a_{ij}約以a_{ii}/a_{jj}之比率遞減。因此為使該元素加速收斂為零，可利用下節介紹之特徵值推移法將所有特徵值減掉σ，σ為約略等於最小之特徵值（QL法可取a_{11}做近似，詳次節），則上述之比率變成較小之$(a_{ii}-\sigma)/(a_{jj}-\sigma)$。在第一個上次對角元素消去後，可暫不考慮第一行第一列之存在，σ可選約略等於其餘之最小特徵值，即第二小之特徵值，以快速消去第二個上次對角元素，餘類推至全部上次對角元素消去為止。不過當有複數特徵值時，對應之上次對角元素無法消去，應予跳過。

由上節之第一段相似轉換之做法，可知必存在某一正交矩陣$[Q]$，能夠將對稱矩陣相似轉換為三對角矩陣，或將不對稱矩陣相似轉換為赫申伯格矩陣。而由第二段相似轉換之做法，可知該正交矩陣$[Q]$有無窮之多。不過有趣的是：如果由某一正交矩陣$[Q]$做相似轉換所得之三對角矩陣或(上)赫申伯格矩陣之次(下)對角元素均不為零，則只要知道該$[Q]$矩陣之最後(或最前)一行之值，則可確定該正交矩陣及其對應之三對角矩陣或(上)赫申伯格矩陣(有2^{n-1}種可能之正負號組合)。以$[H]=[Q]^T[A][Q]$為例証明如下：以$n=4$為例，將$[H][Q]^T=[Q]^T[A]$寫成：

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ & h_{32} & h_{33} & h_{34} \\ & & h_{43} & h_{44} \end{bmatrix} \begin{bmatrix} \langle q_1 \rangle \\ \langle q_2 \rangle \\ \langle q_3 \rangle \\ \langle q_4 \rangle \end{bmatrix} = \begin{bmatrix} \langle q_1 \rangle \\ \langle q_2 \rangle \\ \langle q_3 \rangle \\ \langle q_4 \rangle \end{bmatrix} [A] \tag{7.19}$$

因$[Q]$之最後一行$\{q_4\}$已知，故由式(7.20)及正交特性即$\langle q_i \rangle \{q_j\} = \delta_{ij}$，可得$h_{44}$，因此得式(7.22)之$\langle u_3 \rangle$，及式(7.23)之$h_{43}$。如$h_{43} \neq 0$，即得式(7.24)之$\{q_3\}$(取正負$h_{43}$而有2種可能)。同理由第3式可依次得$h_{34}$, h_{33}, h_{32}各值，如$h_{32} \neq 0$，即得式(7.30)之$\{q_2\}$(取正負h_{32}亦有2種可能)。餘類推。注意若h_{43}變號，則h_{34}僅正負號跟著改變，絕對值不變。同樣若h_{32}變號，則h_{23}, h_{24}亦僅正負號跟著改變，絕對值不變。如已知$[Q]$之第一行，可用$[\{q_1\},\{q_2\},\cdots,\{q_n\}][H]=[A][\{q_1\},\{q_2\},\cdots,\{q_n\}]$証明。

$$h_{43}\langle q_3 \rangle + h_{44}\langle q_4 \rangle = \langle q_4 \rangle [A] \tag{7.20}$$

$$h_{44} = \langle q_4 \rangle [A] \{q_4\} \tag{7.21}$$

$$h_{43}\langle q_3 \rangle = \langle q_4 \rangle [A] - h_{44}\langle q_4 \rangle = \langle u_3 \rangle \tag{7.22}$$

$$h_{43}^2 = \langle u_3 \rangle \{u_3\} \tag{7.23}$$

$$\langle q_3 \rangle = \langle u_3 \rangle / h_{43} \tag{7.24}$$

$$h_{32}\langle q_2\rangle + h_{33}\langle q_3\rangle + h_{34}\langle q_4\rangle = \langle q_3\rangle[A] \tag{7.25}$$

$$h_{34} = \langle q_3\rangle[A]\{q_4\} \tag{7.26}$$

$$h_{33} = \langle q_3\rangle[A]\{q_3\} \tag{7.27}$$

$$h_{32}\langle q_2\rangle = \langle q_3\rangle[A] - h_{33}\langle q_3\rangle - h_{34}\langle q_4\rangle = \langle u_2\rangle \tag{7.28}$$

$$h_{32}^2 = \langle u_2\rangle\{u_2\} \tag{7.29}$$

$$\langle q_2\rangle = \langle u_2\rangle / h_{32} \tag{7.30}$$

由以上之分析可知若已有$[Q]$之最後一行，則用任何方法計算（以上亦為方法之一，但一般不採用），均會求得2^{n-1}種可能正負號組合之一的$[Q]$與$[H]$，但其中各元素之絕對值為固定。若$h_{i+1,i}=0$，則矩陣可分為二個獨立部分個別處理：即1至i間之$i\times i$矩陣，與$i{+}1$至n間之$(n{-}i)\times(n{-}i)$矩陣。

7.6 對角元素之大小順序與非對角元素之收斂

本節將說明QL法之對角元素會傾向於由小至大之順序，即最小值為左上角之a_{11}；而QR法之對角元素會傾向於由大至小之順序，即最小值為右下角之a_{nn}。首先寫出相似轉換前後之係數關係如下：

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} a_{11}c{+}a_{21}s & a_{12}c{+}a_{22}s \\ -a_{11}s{+}a_{21}c & -a_{12}s{+}a_{22}c \end{bmatrix}\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

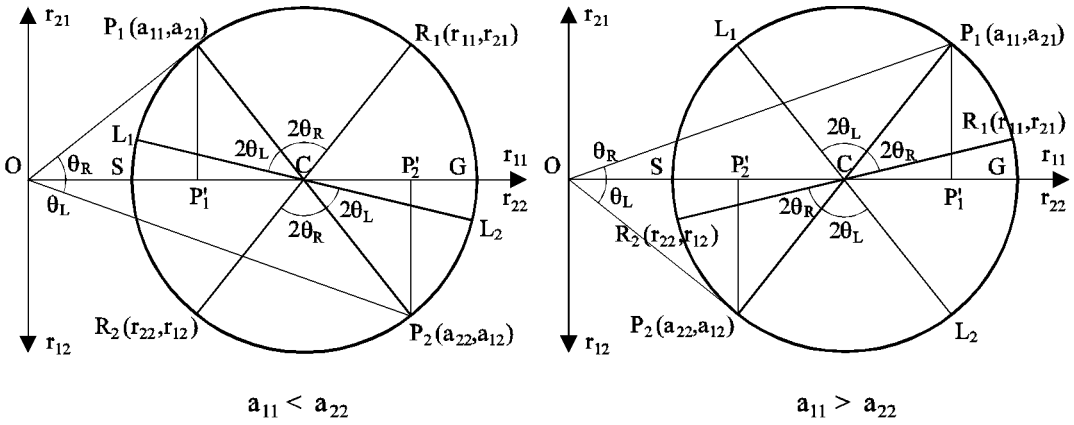
$$= \begin{bmatrix} a_{11}c^2{+}a_{22}s^2{+}(a_{21}{+}a_{12})cs & a_{12}c^2{-}a_{21}s^2{+}(a_{22}{-}a_{11})cs \\ a_{21}c^2{-}a_{12}s^2{+}(a_{22}{-}a_{11})cs & a_{22}c^2{+}a_{11}s^2{-}(a_{21}{+}a_{12})cs \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}$$

對於對稱之情形，即$a_{12}=a_{21}$，可以用莫爾圓 (Mohr circle) 討論較為清楚。莫爾圓係根據以下關係而得（將上列關係代入下式即可証得）：

$$(r_{11}-t)^2 + r_{21}^2 = r^2\ , \qquad (r_{22}-t)^2 + r_{12}^2 = r^2 \tag{7.31}$$

$$t = (a_{11}+a_{22})/2\ , \qquad r = \sqrt{((a_{11}-a_{22})/2)^2 + a_{12}^2} \tag{7.32}$$

以(r_{11}, r_{21})為直角座標時，不同θ角之點$R_1(r_{11}, r_{21})$均會位於一圓周上，該圓之圓心在點$C(t,0)$，半徑為r。另一組點$R_2(r_{22}, r_{12})$亦同樣會位於該圓周上，且R_1CR_2在一直線上。



圖一 莫爾圓

(1)先討論$a_{11} < a_{22}$之情形：以$a_{21} > 0$為例，未轉換前之點$P_1(a_{11}, a_{21})$在圓周之左上$1/4$圓弧，點$P_2(a_{22}, -a_{12})$在圓周之右下$1/4$圓弧；用QR法做轉換時，即由$-a_{11}s+a_{21}c = 0$求轉角θ_R，點$R_1(r_{11}, r_{21})$必位於P_1點之右，點$R_2(r_{22}, r_{12})$必位於P_2點之左。因此QR轉換使左上對角元素r_{11}加大，右下對角元素r_{22}減小，且遲早會使$r_{11} > r_{22}$，可稱其為調序階段。注意此階段並不保証$|r_{21}|$會變小。另有二種特殊情形值得注意：

(1a)原點O若正好在S點處，則R_1點在G點，R_2點在S點，此時$r_{12} = r_{21} = 0$為最理想之情形。

(1b)原點O若正好在P_1'點處，則R_1點在P_2點，R_2點在P_1點，此時正好將對角元素交換位置，通常對角元素均大於零，因此該點為原點之可能最右位置。故R_1在P_1GP_2之間，R_2在P_2SP_1之間，即$2\theta_R \leq \pi$。

(2)現討論$a_{11} > a_{22}$之情形：以$a_{21} > 0$為例，未轉換前之點$P_1(a_{11}, a_{21})$在圓周之右上$1/4$圓弧，點$P_2(a_{22}, -a_{12})$在圓周之左下$1/4$圓弧；用QR法做轉換時，除點$R_1(r_{11}, r_{21})$必位於P_1點之右與點$R_2(r_{22}, r_{12})$必位於P_2點之左外，$|r_{21}|$亦必小於$|a_{21}|$，且遲早會使r_{21}與r_{12}變為零，可稱其為收斂階段。此時前述之二種特殊情形分別為：

(2a)原點O若正好在S點處，則R_1點在G點，R_2點在S點，此時$r_{12} = r_{21} = 0$亦為最理想之情形。

(2b)原點O若正好在P_2'點處，則R_1點會在G點之下，即r_{21}會與a_{21}異號，但$|r_{21}|$亦必小於$|a_{21}|$。

以QL法做轉換時，係由$a_{12}c+a_{22}s = 0$求轉角θ_L，新點位置$L_1(l_{11}, l_{21})$與$L_2(l_{22}, -l_{12})$亦標示於圖一中，此時於$a_{11} > a_{22}$時為調序階段，於$a_{11} < a_{22}$時為收斂階段，因此QL法會使對角元素之最小值移至左上角之a_{11}。

7.7 特徵值推移法

將$[A]\{X\}=\lambda\{X\}$二邊減$\sigma[I]\{X\}$，得$([A]-\sigma[I])\{X\}=(\lambda-\sigma)\{X\}$，故$([A]-\sigma[I])$之特徵值變成$(\lambda-\sigma)$，而其特徵向量不變。即將$[A]$改為$([A]-\sigma[I])$可使所有特徵值均減少$\sigma$，因此稱為特徵值推移法。例如在$[Q][L]$法中，若改用$[A]-\sigma[I]$分解，則$[L][Q]$為$[A]-\sigma[I]$之相似矩陣，其特徵值為$\lambda-\sigma$。因此$[\bar{A}]=[L][Q]+\sigma[I]$為$[A]$之相似矩陣，其特徵值為$\lambda$。

$$[A]-\sigma[I]=[Q_\sigma][L_\sigma] \quad \text{或} \quad [Q_\sigma]^T([A]-\sigma[I])=[L_\sigma] \tag{7.33}$$

$$[\bar{A}]=[L_\sigma][Q_\sigma]+\sigma[I] \quad \text{或} \quad [\bar{A}]=[Q_\sigma]^T[A][Q_\sigma] \tag{7.34}$$

上述求$[Q_\sigma]$之方法，可照式(7.33)將$[A]$之對角元素減σ後利用式(7.10)與式(7.11)之方式計算$[Q_\sigma]^T=[P_{12}^2][P_{23}^3][P_{34}^4][P_{45}^5]$。此種方式稱顯式推移法。另一種求$[Q_\sigma]$之方法稱為隱式推移法，介紹如下：以對稱矩陣為例（不對稱矩陣常用下節之特徵值雙推移法），先求出與顯式推移法相同之第一個正交轉換矩陣$[P_{45}^5]$，並即由下式先算出對應之相似矩陣：

$$[P_{45}^5]\begin{bmatrix} x & x & & & \\ x & x & x & & \\ & x & x & x & \\ & & x & x & x \\ & & & x & x \end{bmatrix}[P_{45}^5]^T=\begin{bmatrix} x & x & & & \\ x & x & x & & \\ & x & x & x' & x' \\ & & 'x & 'x' & 'x' \\ & & 'x & 'x' & 'x' \end{bmatrix} \tag{7.35}$$

此矩陣較三對角矩陣多了a_{35}與a_{53}之非零元素，接著可照第一段之方法，即式(7.9)，將其三對角化。注意當以$[P_{35}^4]$消去a_{35}後，會產生非零元素a_{24}，以$[P_{24}^3]$消去a_{24}後，會產生非零元素a_{13}，最後再以$[P_{13}^2]$消去a_{13}，即得三對角矩陣。亦注意此時每一行僅有一非零元素須消去，而所得之$[Q_\sigma]^T=[P_{13}^2][P_{24}^3][P_{35}^4][P_{45}^5]$，與顯式推移法所得者應為相同，但其中計算各$[P_{ik}^j]$時所用之元素不同(除$[P_{45}^5]$外)：如隱式之$[P_{35}^4]$用$a_{35}$與$a_{45}$配對消去$a_{35}$；顯式之$[P_{34}^4]$用$a_{34}$與$(a_{44}-\sigma)$配對消去$a_{34}$。其中隱式之計算過程中不再用$\sigma$；而顯式之計算過程中須將對角元素減$\sigma$，於轉換後再加回$\sigma$，如式(7.34)。最後再注意：因$[Q_\sigma]^T=[P_{13}^2][P_{24}^3][P_{35}^4][P_{45}^5]$，故$[Q_\sigma]^T$之最後一列與$[P_{45}]$之最後一列相同，而隱式與顯式之$[P_{45}^5]$相同，故二者之$[Q_\sigma]$之最後一行相同。由上節式(7.20)至式(7.30)之分析証明，可知二者之$[Q_\sigma]$與所得三對角矩陣之元素，除可能有正負號之差異外，絕對值應相等。

7.8 特徵值雙推移法

不對稱矩陣之特徵值有可能為複數，利用上節之特徵值推移法，須改用複數值運算，對於實數矩陣，將極為不便。以下介紹一種同時做二次特徵值推移之雙推移法。此雙推移法如其二次推移之複數特徵值互為共軛(注意實數矩陣之複數特徵值必成對出現且互為共軛)，則仍然可以用實數運算。以QR法為例，設$[A]$經二次推移如下，並得式(7.40)之$[\bar{\bar{A}}]$。通常σ_1, σ_2可取$[A]$矩陣右下角之2×2子矩陣之特徵值。

$$[A]-\sigma_1[I]=[Q_1][R_1] \tag{7.36}$$

$$[\bar{A}]=[R_1][Q_1]+\sigma_1[I] \quad \text{或} \quad [\bar{A}]=[Q_1]^T[A][Q_1] \tag{7.37}$$

$$[\bar{A}]-\sigma_2[I]=[Q_2][R_2] \tag{7.38}$$

$$[\bar{\bar{A}}]=[R_2][Q_2]+\sigma_2[I] \quad \text{或} \quad [\bar{\bar{A}}]=[Q_2]^T[\bar{A}][Q_2] \tag{7.39}$$

$$[\bar{\bar{A}}]=[Q_2]^T[Q_1]^T[A][Q_1][Q_2] \tag{7.40}$$

由式(7.37)與式(7.38)可得式(7.41)，該式後乘式(7.36)可得式(7.42)。由該式可知式(7.40)中之轉換矩陣$[Q]=[Q_1][Q_2]$可由矩陣$[\hat{A}]$用QR法求得，而免除了採用式(7.36)與式(7.38)因複數對角元素造成之不便。注意$[R]=[R_2][R_1]$仍然為上三角矩陣。又由式(7.43)知$[\hat{A}]$為實數，因共軛複數對之和$(\sigma_1+\sigma_2)$與積$(\sigma_1\sigma_2)$均為實數。

$$[A]-\sigma_2[I]=[Q_1][Q_2][R_2][Q_1]^T \tag{7.41}$$

$$([A]-\sigma_2[I])([A]-\sigma_1[I])=[Q_1][Q_2][R_2][R_1] \tag{7.42}$$

$$[\hat{A}]=([A]-\sigma_2[I])([A]-\sigma_1[I])=[A]^2-(\sigma_1+\sigma_2)[A]+\sigma_1\sigma_2[I] \tag{7.43}$$

現考慮以Householder法求式(7.42)中$[\hat{A}]$之$[Q]=[Q_1][Q_2]$，可得$[Q]^T=[P_{n-2}]\cdots[P_2][P_1]$。由$[P_k]$矩陣之特性知$[Q]^T$之第一列與$[P_1]$之第一列相同，因$[P_1]$之前乘以矩陣$[P_2]$至$[P_{n-2}]$不改變$[P_1]$第一列之值。另由式(7.20)至式(7.30)之分析証明可知式(7.40)中$[A]$之$[Q]=[Q_1][Q_2]$，只需由$[Q]$之第一行即可確定整個$[Q]$矩陣及赫申伯格型式之$[\bar{\bar{A}}]$矩陣。若$[A]$矩陣亦以Householder法求其正交矩陣$[Q]=[Q_1][Q_2]$，同樣可得$[Q]^T=[\bar{P}_{n-1}]\cdots[\bar{P}_2][\bar{P}_1]$，且$[Q]^T$之第一列與$[\bar{P}_1]$之第一列相同。為使式(7.40)與式(7.42)所得之$[Q]$相同，二者之$[Q]^T$之第一列應相同，因此$[\bar{P}_1]=[P_1]$。求$[P_1]$僅需要$[\hat{A}]$第一行之值列如下式，係由式(7.43)求得。按Householder法令

$\{U\} = \{\hat{a}_{11} + s,\, \hat{a}_{21},\, \hat{a}_{31},\, 0,\, 0\}$ ，$s = \sqrt{\hat{a}_{11}^2 + \hat{a}_{21}^2 + \hat{a}_{31}^2}$ ，即可求得$[P_1]$。並即由式(7.47)先算出對應之相似矩陣$[P_1][A][P_1]$。

$$\hat{a}_{11} = a_{11}^2 + a_{12}a_{21} - (\sigma_1 + \sigma_2)a_{11} + \sigma_1\sigma_2 \tag{7.44}$$

$$\hat{a}_{21} = a_{21}(a_{11} + a_{22} - (\sigma_1 + \sigma_2)) \tag{7.45}$$

$$\hat{a}_{31} = a_{21}a_{32} \tag{7.46}$$

$$[P_1]\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix}[P_1] = \begin{bmatrix} 'x' & 'x' & 'x' & 'x & 'x \\ 'x' & 'x' & 'x' & 'x & 'x \\ 'x' & 'x' & 'x' & 'x & 'x \\ x' & x' & x' & x & x \\ & & & x & x \end{bmatrix} \tag{7.47}$$

此矩陣較赫申伯格矩陣多了非零之a_{31} 、a_{41}與a_{42}，接著照第一段之方法，即式(7.12)至式(7.18)，將其化為赫申伯格矩陣。注意當以$[\bar{P}_2]$消去$a_{31},\, a_{41}$後，會產生非零之$a_{52},\, a_{53}$，最後再以$[\bar{P}_3]$消去$a_{42},\, a_{52}$，及以$[\bar{P}_4]$消去a_{53}，即得赫申伯格矩陣。亦注意此時每行最多僅有二非零元素須消去，而所得之$[Q]^T = [\bar{P}_4][\bar{P}_3][\bar{P}_2][P_1]$，應與$[P_3][P_2][P_1]$相同，但其中計算各$[\bar{P}_j]$時所用之元素不同，且過程中不再用$\sigma_1$與$\sigma_2$，故為隱式之做法。

7.9 帶矩陣之處理

對稱帶矩陣轉換為三對角矩陣如按前述做法，即以行為順序消去次對角以下之元素，則其帶寬會持續增加。如欲維持帶寬不變，可考慮採用如下標示之x下標之消去順序：即先消去左邊矩陣之$x_1, x_2, \ldots, x_5$，求得如右邊之矩陣後再消去其中之$x_1, x_2, \ldots, x_6$。

$$\begin{bmatrix} x & x & x & x_1 & & & & \\ x & x & x & x & x_2 & & & \\ x & x & x & x & x & x_3 & y_1 & \\ x_1 & x & x & x & x & x & x_4 & \\ & x_2 & x & x & x & x & x & x_5 \\ & & x_3 & x & x & x & x & x \\ & & y_1 & x_4 & x & x & x & x \\ & & & & x_5 & x & x & x \end{bmatrix} \qquad \begin{bmatrix} x & x & x_1 & & & & & \\ x & x & x & x_2 & y_1 & & & \\ x_1 & x & x & x & x_3 & & & \\ & x_2 & x & x & x & x_4 & z_1 & \\ & y_1 & x_3 & x & x & x & x_5 & \\ & & & x_4 & x & x & x & x_6 \\ & & & z_1 & x_5 & x & x & x \\ & & & & & x_6 & x & x \end{bmatrix}$$

注意在消去x_1時(前乘$[P_{41}^3]$)會產生y_1，該值應即時消去以避免儲存之，在消去y_1時若又產生另一非零元素如z_1，亦同樣應即時消去，餘類推。不對稱矩陣亦可採用類似方式以維持下(或上)三角部分之帶寬。

7.10 準三角矩陣之特徵值與特徵向量

對於不對稱矩陣，經過正交矩陣之相似轉換後均可轉為準三角矩陣。準三角矩陣較三角矩陣多了一排非零之次對角線，但不會有二個(含二個以上)非零元素連續出現在次對角線上，此點為其所以與赫申伯格矩陣不同之處。準三角矩陣之特徵值極易求得：若(1)某對角元素之相鄰二個次對角元素均為零，則該對角元素即為特徵值；否則(2)每一非零次對角元素即對應於一組共軛複數特徵值，為包含此次對角元素a_{ij}之2×2之子矩陣之特徵值，即$(a_{ii}-\lambda)(a_{jj}-\lambda)-a_{ij}a_{ji}=0$之二根：$\lambda=\frac{1}{2}(a_{ii}+a_{jj})\pm i\sqrt{\Delta-\frac{1}{4}(a_{ii}+a_{jj})^2}$，其中$\Delta=a_{ii}a_{jj}-a_{ij}a_{ji}$。注意$\Delta>\frac{1}{4}(a_{ii}+a_{jj})^2>0$，否則即非複數特徵值。準三角矩陣之特徵向量亦極易求得：

(1)對實數特徵值：以$\lambda_2{=}\lambda_3{=}a_{22}{=}a_{33}{=}\lambda$為例，其對應之$([A]-\lambda[I])\{X\}=\{0\}$如式(7.48)，可改如式(7.49)，即可由前進代入求得二個特徵向量，如式(7.50)。注意1上方之零值經前進代入後仍維持零值，且係數矩陣第一行之值不影響特徵向量之計算。亦注意於$a_{22}=a_{33}$之重根時，a_{32}須等於零，式(7.50)才有二個獨立特徵向量。因為由第3式：$x_3=-a_{32}x_2+x_3$，若$a_{32}\neq 0$，則$x_2=0$。故只有x_3可取任意非零值，亦即僅第二行之向量為特徵向量。

$$\begin{bmatrix} a_{11}-\lambda & & & & \\ a_{21} & 0 & & & \\ a_{31} & a_{32} & 0 & & \\ a_{41} & a_{42} & a_{43} & a_{44}-\lambda & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55}-\lambda \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \{0\} \tag{7.48}$$

$$\left[\begin{array}{c|cc|cc} 1 & & & & \\ \hline & 1 & & & \\ & & 1 & & \\ \hline & a_{42} & a_{43} & a_{44}-\lambda & \\ & a_{52} & a_{53} & a_{54} & a_{55}-\lambda \end{array}\right] \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -a_{32} & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} x_2 \\ x_3 \end{Bmatrix} \tag{7.49}$$

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -a_{32} & 1 \\ x_{42} & x_{43} \\ x_{52} & x_{53} \end{bmatrix} \begin{Bmatrix} x_2 \\ x_3 \end{Bmatrix} \tag{7.50}$$

(2)對複數特徵值：以λ_2=$\bar{\lambda}_3$=λ為例，其對應之$([A]-\lambda[I])\{X\}=\{0\}$如式(7.51)，經過以第1列消去第2列以後$x_1$之係數，再以第2列消去第3列$x_2$之係數後，該式變成式(7.52)。注意其中$c_{23}=a_{23}/(a_{22}-\lambda)=(a_{33}-\lambda)/a_{32}$，$c_{33}=(a_{33}-\lambda)-a_{32}c_{23}=(a_{33}-\lambda)-a_{32}(a_{33}-\lambda)/a_{32}=0$。將式(7.52)改寫成式(7.53)，即可很容易地由前進代入求得特徵向量，如式(7.54)。

$$\begin{bmatrix} a_{11}-\lambda & & & & \\ a_{21} & a_{22}-\lambda & a_{23} & & \\ a_{31} & a_{32} & a_{33}-\lambda & & \\ a_{41} & a_{42} & a_{43} & a_{44}-\lambda & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55}-\lambda \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \{0\} \tag{7.51}$$

$$\left[\begin{array}{c|cc|cc} 1 & & & & \\ \hline 0 & 1 & c_{23} & & \\ 0 & 0 & 0 & & \\ \hline 0 & a_{42} & a_{43} & a_{44}-\lambda & \\ 0 & a_{52} & a_{53} & a_{54} & a_{55}-\lambda \end{array}\right] \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \{0\} \tag{7.52}$$

$$\left[\begin{array}{c|cc|cc} 1 & & & & \\ \hline & 1 & & & \\ & 0 & 1 & & \\ \hline & a_{42} & a_{43} & a_{44}-\lambda & \\ & a_{52} & a_{53} & a_{54} & a_{55}-\lambda \end{array}\right] \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -c_{23} \\ 1 \\ 0 \\ 0 \end{Bmatrix} x_3 \tag{7.53}$$

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -c_{23} \\ 1 \\ x_{43} \\ x_{53} \end{Bmatrix} x_3 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ x_{32} & y_{33} \\ x_{42} & y_{43} \\ x_{52} & y_{53} \end{bmatrix} \begin{Bmatrix} 1 \\ i \end{Bmatrix} \tag{7.54}$$

上式中係選$x_3 = -\bar{c}_{23}/(\bar{c}_{23}c_{23}) = x_{32}+y_{33}i$，以使第一個非零元素為實數1。現將複數特徵向量之實數部分x放在第2行，虛數部分y放在第3行，再配合其他實數之特徵向量，可將所有特徵向量之非零係數置於下三角矩陣中。則特徵向量矩陣$[U]$及相似矩陣$[U]^{-1}[A][U]$可寫成(其中$\alpha_2+i\beta_2 = \lambda_2=\bar{\lambda}_3$)：

$$[U] = \begin{bmatrix} 1 & & & & \\ u_{21} & 1 & & & \\ u_{31} & x_{32} & y_{33} & & \\ u_{41} & x_{42} & y_{43} & 1 & \\ u_{51} & x_{52} & y_{53} & u_{54} & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & 1 & & \\ & i & -i & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$$

$$[U]^{-1}[A][U] = \begin{bmatrix} 1 & & & & \\ & \frac{1}{2} & -\frac{i}{2} & & \\ & \frac{1}{2} & \frac{i}{2} & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 & & & & \\ & \alpha_2 & \beta_2 & & \\ & -\beta_2 & \alpha_2 & & \\ & & & \lambda_4 & \\ & & & & \lambda_5 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & 1 & & \\ & i & -i & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$$

計算特徵向量時如果從最後一行開始，並於每次求出一個特徵向量後，馬上將相似矩陣算出，即可使相似矩陣從右下角起逐步變成對角線為1×1或2×2之子矩陣之準對角矩陣，於計算新的特徵向量時(特別在計算複數特徵向量時)較為單純。

7.11 一般特徵值問題之解法

前面提過一般特徵值問題$[A]\{X\} = \lambda[B]\{X\}$，如$[A]^{-1}$或$[B]^{-1}$存在，則可改為標準型：$[B]^{-1}[A]\{X\} = \lambda\{X\}$ 或 $[A]^{-1}[B]\{X\} = \lambda^{-1}\{X\}$。以下說明一種類似對稱矩陣為保住對稱特性所用之方法，該法亦適用於$[A]$與$[B]$均為奇異矩陣之情形。首先解式(7.57)之標準特徵值問題，並得其正交轉換矩陣$[V]$，與相似矩陣$[T]$，如式(7.58)所示。其中$[T]$為準下三角矩陣，即$[T]$矩陣之上次對角元素無連續不為零者。每一非零之上次對角元素對應一對共軛複數特徵值。再由上節方法計算特徵向量，因這些特徵向量可組成一下三角矩陣$[U]$，以該矩陣為轉換矩陣，可將$[T]$相似轉換為如下之$[S] = [U]^{-1}[T][U]$，注意矩陣$[S]$為準對角矩陣，對角元素為實數特徵值或複數特徵值之實數部分，次對角元素為複數特徵值之虛數部分。亦

注意對應特徵值為零之對角元素為零。

$$[S]=[U]^{-1}[T][U]=\begin{bmatrix}\lambda_1 & & & & \\ & \alpha_2 & \beta_2 & & \\ & -\beta_2 & \alpha_2 & & \\ & & & \lambda_4 & \\ & & & & \lambda_5\end{bmatrix}$$

為說明方便，進一步令$[S]=[D][K][J][D]$。其中$[K]$為準對角矩陣，$[D]$與$[J]$均為對角矩陣：

(1)若$[S]$之次對角元素$s_{ij}=0$，令$j=i+1$：

(1a)若$s_{ii}>0$，則令$d_{ii}=\sqrt{+s_{ii}}$，$j_{ii}=1$，$k_{ii}=1$；如$d_{11}=x$。

(1b)若$s_{ii}<0$，則令$d_{ii}=\sqrt{-s_{ii}}$，$j_{ii}=1$，$k_{ii}=-1$；如$d_{55}=y$。

(1c)若$s_{ii}=0$，則令$d_{ii}=1$，$j_{ii}=0$，$k_{ii}=1$；如$d_{44}=0$。

(2)若$[S]$之次對角元素$s_{ij}\neq 0$，則令$d_{ii}=d_{jj}=\sqrt[4]{s_{ii}s_{jj}-s_{ij}s_{ji}}$，$j_{ii}=j_{jj}=1$，$k_{ij}=-k_{ji}=s_{ij}/d_{ii}d_{jj}$；如$d_{22}=d_{33}=z$，$k_{22}=k_{33}=c$，$k_{23}=-k_{32}=s$。

典型之$[S]=[D][K][J][D]$列示如下：

$$[S]=\begin{bmatrix}x^2 & & & & \\ & cz^2 & sz^2 & & \\ & -sz^2 & cz^2 & & \\ & & & 0 & \\ & & & & -y^2\end{bmatrix}=[D][K][J][D]= \tag{7.55}$$

$$\begin{bmatrix}x & & & & \\ & z & & & \\ & & z & & \\ & & & 1 & \\ & & & & y\end{bmatrix}\begin{bmatrix}1 & & & & \\ & c & s & & \\ & -s & c & & \\ & & & 1 & \\ & & & & -1\end{bmatrix}\begin{bmatrix}1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & 1\end{bmatrix}\begin{bmatrix}x & & & & \\ & z & & & \\ & & z & & \\ & & & 1 & \\ & & & & y\end{bmatrix}$$

由上列關係及式(7.58)與式(7.59)可得$[B]$如式(7.60)。將式(7.60)代入式(7.56)可得式(7.61)，令$[\bar{A}]$與$\{\bar{X}\}$如式(7.62)與式(7.63)，即可將式(7.56)之一般特徵值問題改為式(7.64)之“準”標準特徵值問題。若$[J]$等於$[I]$，該式即為標準特徵值問題。注意對於任意正交矩陣$[Q]$，$[Q]^T[I][Q]$均會等於$[I]$，但$[Q]^T[J][Q]$不一定會等於$[J]$。

$$[A]\{X\} = \lambda[B]\{X\} \tag{7.56}$$

$$[B]\{Y\} = \mu\{Y\} \tag{7.57}$$

$$[V]^T[B][V] = [T] \tag{7.58}$$

$$[U]^{-1}[T][U] = [S] \tag{7.59}$$

$$[B] = [V][U][D][K][J][D][U]^{-1}[V]^T \tag{7.60}$$

$$\begin{aligned}[K]^{-1}[D]^{-1}[U]^{-1}[V]^T[A][V][U][D]^{-1}[D][U]^{-1}[V]^T\{X\}& \\ = \lambda[J][D][U]^{-1}[V]^T\{X\}&\end{aligned} \tag{7.61}$$

$$[\bar{A}] = [K]^{-1}[D]^{-1}[U]^{-1}[V]^T[A][V][U][D]^{-1} \tag{7.62}$$

$$\{\bar{X}\} = [D][U]^{-1}[V]^T\{X\} \tag{7.63}$$

$$[\bar{A}]\{\bar{X}\} = \lambda[J]\{\bar{X}\} \tag{7.64}$$

$$[\hat{A}]\{\hat{X}\} = \hat{\lambda}\{\hat{X}\} \tag{7.65}$$

如$[A]$與$[B]$均為對稱(此時$[U]=[I]$)，且$[K]=[I]$(無小於零之特徵值)，則$[\bar{A}]$亦為對稱；否則$[\bar{A}]$為不對稱。但$[\bar{A}]$不是$[A]$之相似矩陣，因$[D]$不是正交矩陣，且$[K]^{-1}$僅做前乘運算。若$[J]=[I]$，則式(7.64)即可照標準特徵值問題求解。若$[J]\neq[I]$，即其對角元素有零值，可按式(7.66)求$[\hat{A}]$後，再解式(7.65)之標準問題。以$n=4$，$j_{33}=0$為例：

(1)若$\bar{a}_{33}\neq 0$，可按下式計算$[\hat{A}]$及$\{\hat{X}\}$與$\{\bar{X}\}$之關係。式(7.66)之運算類似高斯消去法，亦相當於結構動力分析之靜態濃縮法。此時如將$[\hat{A}]$之第3行與第3列移除，$\hat{x}_3$亦自$\{\hat{X}\}$中移去，而$[J]$之零對角元素之行與列移除後即為$[I]$矩陣，因此即變成標準特徵值問題。

$$\begin{aligned}[\hat{A}] &= \begin{bmatrix} 1 & & -\frac{\bar{a}_{13}}{\bar{a}_{33}} & \\ & 1 & -\frac{\bar{a}_{23}}{\bar{a}_{33}} & \\ & & 1 & \\ & & -\frac{\bar{a}_{43}}{\bar{a}_{33}} & 1 \end{bmatrix} \begin{bmatrix} \bar{a}_{11} & \bar{a}_{12} & \bar{a}_{13} & \bar{a}_{14} \\ \bar{a}_{21} & \bar{a}_{22} & \bar{a}_{23} & \bar{a}_{24} \\ \bar{a}_{31} & \bar{a}_{32} & \bar{a}_{33} & \bar{a}_{34} \\ \bar{a}_{41} & \bar{a}_{42} & \bar{a}_{43} & \bar{a}_{44} \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ -\frac{\bar{a}_{31}}{\bar{a}_{33}} & -\frac{\bar{a}_{32}}{\bar{a}_{33}} & 1 & -\frac{\bar{a}_{34}}{\bar{a}_{33}} \\ & & & 1 \end{bmatrix} \\ &= \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} & 0 & \hat{a}_{14} \\ \hat{a}_{21} & \hat{a}_{22} & 0 & \hat{a}_{24} \\ 0 & 0 & \bar{a}_{33} & 0 \\ \hat{a}_{41} & \hat{a}_{42} & 0 & \hat{a}_{44} \end{bmatrix} \end{aligned} \tag{7.66}$$

$$\{\hat{X}\} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ \frac{\bar{a}_{31}}{\bar{a}_{33}} & \frac{\bar{a}_{32}}{\bar{a}_{33}} & 1 & \frac{\bar{a}_{34}}{\bar{a}_{33}} \\ & & & 1 \end{bmatrix} \{\bar{X}\} \tag{7.67}$$

如$[J]$之零對角元素不只一個，可依序做類似處理。注意若$[\bar{A}]$對稱則$[\hat{A}]$亦對稱，但$[\hat{A}]$不是$[\bar{A}]$之相似轉換。亦特別注意式(7.65)對應於$j_{33}=0$之特徵值為$\hat{\lambda}_3=\bar{a}_{33}$；但式(7.64)對應於$j_{33}=0$之特徵值為$\lambda_3=\bar{a}_{33}/j_{33}$，即等於無窮大。

(2)若$\bar{a}_{33}=0$，且$[\bar{A}]$之第3行與第3列元素均為零，則可直接令$[\hat{A}]=[\bar{A}]$，$\{\hat{X}\}=\{\bar{X}\}$，再解式(7.65)之標準特徵值問題。但注意對應於$j_{33}=0$之特徵值$\hat{\lambda}_3=0$，但特徵值$\lambda_3=0/0$，即可等於任意值。

(3)若$\bar{a}_{33}=0$，但$[\bar{A}]$之第3行與第3列元素不全為零，則表示原特徵值問題不合理或方程式矛盾而無解。如將$[\bar{A}]$之第3行與第3列元素直接設為零，亦可勉強按(2)之方式繼續求解。

最後注意：$[U]$為下三角矩陣，故$[T]$前乘$[U]^{-1}$可由前進代入計算。有關計算詳[表四]副程式$AZEXBZ$。

7.12 矩陣係數之平衡

對於不對稱矩陣如其係數大小太懸殊，常會造成較大之捨入誤差，此誤差通常與歐氏模長(Euclidean norm)成正比。歐氏模長為矩陣係數平方和之方根值，該值與矩陣係數絕對值之和相差不多，故為節省計算時間，以後計算即以後者代替。以下介紹之平衡法(balancing)係採用對角矩陣型式之轉換矩陣做相似轉換，以使相似矩陣之歐氏模長較原矩陣者為小。考慮式(7.68)之轉換，$[D]$為對角矩陣，因該項轉換不改變對角元素之值，故以式(7.69)與式(7.70)計算C_i, $\tilde{C}_i$與R_i, $\tilde{R}_i$時，並不包含矩陣$[A]$或$[\tilde{A}]$之對角元素。現考慮$[D]$中僅某一i值之$d_i\neq 1$，其餘之$d_k=1$，$k\neq i$，則$[\tilde{A}]$與$[A]$僅第i行與第i列不同，歐氏模長之改變量為$\tilde{C}_i+\tilde{R}_i$與C_i+R_i之差。欲使歐氏模長變小，應使$\tilde{C}_i+\tilde{R}_i$為最小，即應選d_i使$C_i*d_i+R_i/d_i$為最小，因此得$C_i*d_i=R_i/d_i$。另外為了避免因$[D]$之轉換而造成捨入誤差，因此宜選d_i為計算機實數之基底(大部分為2)之整數次方之值，設基底為2，應使$0.5\leq\tilde{C}_i/\tilde{R}_i<2$。

$$[\tilde{A}] = [D]^{-1}[A][D] \tag{7.68}$$

$$C_i = \sum_{j=1,j\neq i}^{n} |a_{ij}| \ , \qquad \tilde{C}_i = \sum_{j=1,j\neq i}^{n} |\tilde{a}_{ij}| = C_i * d_i \tag{7.69}$$

$$R_i = \sum_{j=1,j\neq i}^{n} |a_{ji}| \ , \qquad \tilde{R}_i = \sum_{j=1,j\neq i}^{n} |\tilde{a}_{ji}| = R_i/d_i \tag{7.70}$$

上述之i可依序由1至n對各行各列運算求d_i，並算出轉換得之相似矩陣，稱一循環。但因某行某列調整後會影響已經平衡過之其他各行各列，故上述之運算應反覆多次，直至整個循環之各行各列均不必做進一步調整為止。

7.13 由特性方程式計算特徵值

特徵值之計算亦可由$P_n(\lambda) = |[A] - \lambda[I]| = 0$之$n$次多項式，稱特性方程式，求得$n$個特徵值。將各特徵值代入特徵方程式即可求得對應之特徵向量。但如直接由$|[A] - \lambda[I]|$展開求多項式之係數將較為繁瑣且效率較差。較理想之做法為先將$[A]$矩陣做相似轉換轉成赫申伯格矩陣或三對角矩陣，再求其特性方程式較為方便。以下為下赫申伯格矩陣之特性方程式之求法，其中$P_i(\lambda)$為赫申伯格左上角之部分$i \times i$之子矩陣之i次特性多項式。由式(7.75)計算$i = 1, 2, \ldots, n$時之$P_i(\lambda)$，最後即可求得$P_n(\lambda)$。對於對稱之三對角矩陣，$P_i(\lambda)$可簡化為式(7.76)。

$$P_o(\lambda) = 1 \tag{7.71}$$

$$P_1(\lambda) = (a_{11} - \lambda) \tag{7.72}$$

$$P_2(\lambda) = (a_{22} - \lambda)P_1(\lambda) - a_{12}a_{21}P_o(\lambda) \tag{7.73}$$

$$P_3(\lambda) = (a_{33} - \lambda)P_2(\lambda) - a_{23}a_{32}P_1(\lambda) + a_{23}a_{12}a_{31}P_o(\lambda) \tag{7.74}$$

$$\cdots$$

$$P_i(\lambda) = (a_{ii} - \lambda)P_{i-1}(\lambda) + \sum_{k=1}^{i-1}(-1)^{i-k}(\prod_{j=k}^{i-1} a_{j,j+1})a_{ik}P_{k-1}(\lambda) \tag{7.75}$$

$$P_i(\lambda) = (a_{ii} - \lambda)P_{i-1}(\lambda) - a_{i,i-1}^2 P_{i-2}(\lambda) \tag{7.76}$$

7.14 一般特徵值問題之直接解法

一般特徵值問題亦可用QZ法直接對$[A]$與$[B]$同時做轉換。$[Q]$指對$[A]$與$[B]$之前乘正交矩陣，$[Z]$指對$[A]$與$[B]$之後乘正交矩陣。QZ法係根據如下定理：存在正交矩陣$[Q]$與$[Z]$可同時使$[Q][A][Z]$與$[Q][B][Z]$成為上三角矩陣。因$[A]\{X\} = \lambda[B]\{X\}$與$[Q][A][Z]\{Y\} = \lambda[Q][B][Z]\{Y\}$有相同之特徵值，且其特徵向量具如下之關係：$\{X\} = [Z]\{Y\}$。注意在標準特徵值問題之相似轉換為$[Q]^T[A][Q]$，其前乘正交矩陣與後乘正交矩陣必須互為轉置。而在一般特徵值問題之相似轉換為$[Q][A][Z]$與$[Q][B][Z]$，其前乘正交矩陣與後乘正交矩陣不必互為轉置，亦可謂二者沒有關聯，而二特徵向量之轉換僅與$[Z]$有關而與$[Q]$無關。QZ法基本上分為三個階段求特徵值：(1)第一段分為二步：(1a)先將矩陣$[B]$轉成上三角矩陣；(1b)再將矩陣$[A]$轉成上赫申伯格矩陣同時維持$[B]$為上三角矩陣。(2)第二段將上赫申伯格矩陣$[A]$轉成準上三角矩陣同時維持$[B]$為上三角矩陣。(3)第三段將準上三角矩陣$[A]$轉成上三角矩陣同時維持$[B]$為上三角矩陣，並求出特徵值。第一段第一步先用Householder型或Givens型正交矩陣$[Q]$自第1行起(消去順序如下標所示)，將$[B]$轉成上三角矩陣$[Q][B]$，同時計算$[Q][A]$。第二步再用Givens型正交矩陣$[Q]$與$[Z]$自第1行起由下而上(消去順序如下標所示)，將$[A]$轉成上赫申伯格矩陣$[Q][A][Z]$，同時保持$[Q][B][Z]$為上三角矩陣。事實上所有之$[Z]$均是為了將$([Q][B])[Z]$恢復為上三角矩陣所需之正交轉換。

$$\begin{bmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a_3 & a & a & a & a \\ a_2 & a_5 & a & a & a \\ a_1 & a_4 & a_6 & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b \\ b_1 & b & b & b & b \\ b_1 & b_2 & b & b & b \\ b_1 & b_2 & b_3 & b & b \\ b_1 & b_2 & b_3 & b_4 & b \end{bmatrix}$$

注意在用$[Q] = [P_{ik}^{i-1}]$以$a_{i-1,k}$做配對消去$[A]$之a_{ik}時，在$[B]$之下次對角產生$b_{i,i-1}$，此元素須用$[Z] = [P_{i,i-1}^{\prime i}]$以$b_{ii}$做配對將其消去，注意此$[Z]$後乘$[Q][A]$不影響已變為零之元素。以下為消去$a_{41}$之情形，產生元素為$b_{43}$。前乘矩陣為$[P_{41}^{3}]$，後乘矩陣為$[P_{43}^{\prime 4}]$。以上為第一段之運算，二步之計算量均為有限。

$$\begin{bmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a_3 & a & a & a & a \\ 0 & a_5 & a & a & a \\ & a_4 & a_6 & a & a \end{bmatrix} \qquad \begin{bmatrix} b & b & b & b & b \\ & b & b & b & b \\ & & b & b & b \\ & & b_2 & b & b \\ & & & & b \end{bmatrix}$$

第二段基本上以$[C]=[A][B]^{-1}$為準則，反覆以QR法配合特徵值之推移，以加速將$[A]$與$[B]$化為準上三角矩陣或上三角矩陣。式(7.77)與式(7.78)為對$[C]=[A][B]^{-1}$所做之QR法，若按式(7.79)對$[A]$與$[B]$做QZ轉換，由式(7.80)可知亦相當於式(7.78)對$[C]$之QR轉換。

$$[R] = [Q]([C]-\sigma[I]) = [Q]([A][B]^{-1}-\sigma[I]) \tag{7.77}$$

$$[\bar{C}] = [R][Q]^T+\sigma[I] = [Q][C][Q]^T = [Q][A][B]^{-1}[Q]^T \tag{7.78}$$

$$[\bar{A}] = [Q][A][Z]\,, \qquad [\bar{B}] = [Q][B][Z] \tag{7.79}$$

$$[\bar{A}][\bar{B}]^{-1} = [Q][A][Z][Z]^T[B]^{-1}[Q]^T = [Q][A][B]^{-1}[Q]^T = [\bar{C}] \tag{7.80}$$

$$[Q]([A]-\sigma[B]) = [R"] = [R][B] \tag{7.81}$$

但$[B]^{-1}$不一定存在，因此式(7.77)之$[Q]$改由式(7.81)計算，即對$[A]-\sigma[B]$前乘$[Q]$使成為上三角矩陣$[R"]$，此上三角矩陣即等於$[R][B]$，因$[R]$與$[B]$均為上三角矩陣，其乘積亦為上三角矩陣。$[Z]$則係配合所得之$[Q]$，使$[Q][B][Z]$恢復為上三角矩陣，同時$[Q][A][Z]$亦恢復為上赫申伯格矩陣。以下為詳細做法：因$[A]-\sigma[B]$為上赫申伯格矩陣，故選用$[Q]=[P^{n-1}_{n,n-1}]\cdots[P^2_{32}][P^1_{21}]$可自第1行起將下次對角元素消去。因$[P^1_{21}][B]$會使上三角矩陣$[B]$多出一非零元素$b_{21}$，此元素可用Givens型正交矩陣$[P'^2_{21}]$以$b_{22}$做配對消去之。同理前乘$[P^2_{32}]$於$[P^1_{21}][B][P'^2_{21}]$產生$b_{32}$可用$[P'^3_{32}]$後乘消去，餘類推，可得$[Z]=[P'^2_{21}][P'^3_{32}]\cdots[P'^n_{n,n-1}]$。此$[Z]$除使$[Q][B][Z]$為上三角矩陣外，亦會使$[Q][A][Z]$為上赫申伯格矩陣，因為：由式(7.81)可知$[Q][A][Z]=[R"][Z]+\sigma[Q]B][Z]$，但$[R"][Z]=[R"][P'^2_{21}][P'^3_{32}]\cdots[P'^n_{n,n-1}]$為上赫申伯格矩陣，與上三角矩陣$\sigma[Q]B][Z]$相加仍然為上赫申伯格矩陣。事實上$[Q][A][Z]$可由$[R"][Z]+\sigma[Q]B][Z]$計算較有效率。在此提醒注意一般型之$QZ$法與標準型之$QR$法之異同：(1)求$[Q]$使第一段之$[Q][A]$為上赫申伯格矩陣，或第二段之$[Q]([A]-\sigma[I])$或$[Q]([A]-\sigma[B])$為上三角矩陣。做法上二者相同。(2)一般型求$[Z]$使$([Q][B])[Z]$恢復為上三角矩陣；標準型之$[Z]=[Q]^T$即可使$([Q][I])[Q]^T$恢復為$[I]$。注意此$[Z]$與對應之$[Q]^T$型

式相同，僅數值不同，$[Z]$須根據$[Q][B]$求得。因此對應於一般型之QZ法之稱呼，可稱標準型之QR法為QQ^T法。有了以上之對應後，可知特徵值雙推移法亦可比照辦理，在此不擬贅述。最後注意QZ法無法保住$[A]$與$[B]$之對稱性，故效率上至少打對折。對於不對稱矩陣，QZ法與二次標準型之解法之運算量大致相等， 但運算複雜度增加，故不建議採用。

7.15 程式

［表一］ 實數對稱矩陣之標準特徵值問題

```
************************************************************************
      SUBROUTINE TRIDIG(NA,N,A,D,E,Z)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NA,N),D(N),E(N),Z(NA,N)
C**   ============================================================    **
C**   Simularity transform symmetric matrix to tridiagonal matrix     **
C**   By Householder method :    [P]'[A][P] ==> [T]                    **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = Symmetric matrix given upper triangular part           **
C*O          = [T] in upper triangular part                           **
C*O   D(K)   = T(K,K)   = Diagonal elements of tridiagonal matrix     **
C*O   E(K)   = T(K-1,K) = T(K,K-1) = Sub-diagonal elements of [T]     **
C*I   Z(N,N)  = [I] or [X] = Matrix to be post-multiplied by [P]      **
C*O           = [P] or [X][P]                                        **
C**   ============================================================    **
      DO 80 K=N,1,-1
      AMX=0.0
      DO 20 I=1,K-1
   20 AMX=DMAX1(AMX,DABS(A(I,K)))
C**   +----------------------------------------------------------+    **
C**   | In case of A(1:K-1,K)=0, set D(K) & E(K) (also for K=1)  |    **
C**   +----------------------------------------------------------+    **
      E(K)=0.0
      D(K)=A(K,K)
      IF(AMX.EQ.0.0) GO TO 80
C**   +----------------------------------------------------------+    **
C**   | Scale u-vector to avoid overflow, {u} = {w}sqrt(2) = {D} |    **
C**   +----------------------------------------------------------+    **
      S=0.0
      DO 25 I=1,K-1
      D(I)=A(I,K)/AMX
   25 S=S+D(I)*D(I)
      S=DSIGN(DSQRT(S),D(K-1))
      E(K)=-S*AMX
      D(K-1)=D(K-1)+S
      T=DSQRT(S*D(K-1))
      DO 30 I=1,K-1
   30 D(I)=D(I)/T
C**   +----------------------------------------------------------+    **
C**   | Compute {p} = [A]{u} = [A]{w}sqrt(2) = {E}                |    **
C**   +----------------------------------------------------------+    **
      DO 40 J=1,K-1
```

```
      EJ=A(J,J)*D(J)
      DO 35 I=1,J-1
      E(I)=E(I)+A(I,J)*D(J)
   35 EJ  =EJ  +A(I,J)*D(I)
   40 E(J)=EJ
C**   +--------------------------------------------------------+    **
C**   | Compute {q} = {p}-k{u} ; k = <u>{p}/2 = <w>[A]{w} = WAW  |    **
C**   +--------------------------------------------------------+    **
      UAU=0.0
      DO 50 I=1,K-1
   50 UAU=UAU+E(I)*D(I)
      WAW=UAU*0.5
      DO 60 I=1,K-1
   60 E(I)=E(I)-WAW*D(I)
C**   +--------------------------------------------------------+    **
C**   | Compute [A] = [A] - {u}<q> - {q}<u>                     |    **
C**   +--------------------------------------------------------+    **
      DO 70 J=1,K-1
      DO 65 I=1,J
   65 A(I,J)=A(I,J)-D(I)*E(J)-E(I)*D(J)
   70 A(J,K)=0.0
      A(K-1,K)=E(K)
C**   +--------------------------------------------------------+    **
C**   | Compute  S(1:N) = [Z]{u}  from  D(1:K-1) = <u>           |    **
C**   +--------------------------------------------------------+    **
      DO 75 I=1,N
      S=0.0
      DO 72 J=1,K-1
   72 S=S+Z(I,J)*D(J)
C**   +--------------------------------------------------------+    **
C**   | Compute  [Z] = [Z][[I] - 2{w}<w>] = [Z] - [Z]{u}<u>      |    **
C**   +--------------------------------------------------------+    **
      DO 75 J=1,K-1
   75 Z(I,J)=Z(I,J)-S*D(J)
   80 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE TRIQLI(NA,N,D,E,Z,IERR)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION D(N),E(N),Z(NA,N)
      EQUIVALENCE (C0,C1),(S0,S1),(G0,G1),(P0,P1)
C**   ========================================================    **
C**   Simularity transform tridiagonal matrix to diagonal matrix     **
C**   By QL Implicit shift algorithm : [Q]'[T][Q] ==> [D]            **
C**   --------------------------------------------------------    **
C*I   D(K)   = T(K,K) = Diagonal element of tridiagonal matrix [T]    **
C*O          = D(K,K) = Eigenvalues of [T]                          **
CWI   E(K)   = T(K-1,K) = T(K,K-1) =  Sub-diagonal elements of [T]    **
C*I   Z(N,N)  = [P] or [X] = Matrix to be post-multiplied by [Q]     **
C*O           = [P][Q] or [X][Q]                                    **
C*O   IERR   = J: J-th eigenvalue no converge; = 0: normal return     **
C**   ========================================================    **
      IERR=0
      IF(N.EQ.1) RETURN
      DO 10 I=1,N-1
   10 E(I)=E(I+1)
      E(N)=0.0
C**   --------------------------------------------------------    **
```

```
      DO 80 K=1,N
      DO 70 ITER=1,30
      DO 20 L=K,N-1
      DD=DABS(D(L))+DABS(D(L+1))
      IF(DD+DABS(E(L)).EQ.DD) GO TO 30
   20 CONTINUE
      L=N
   30 IF(L.EQ.K) GO TO 80
C**   +-------------------------------------------------------------+    **
C**   | | D-X  E  | = | -Y    E   |   Y=X-D ; Z=Y/E ; P=(F-D)/2E |    **
C**   | |  E  F-X |   |  E  F-D-Y |   Y*Y - (F-D)*Y - E*E = 0     |    **
C**   | ----------------------------------------------------------- |    **
C**   | Z*Z - 2PZ - 1 = 0 ; (Y/E)**2 - 2((F-D)/2E)*(Y/E) - 1 = 0 |    **
C**   | Large.Z = P+sqrt(P*P+1) ; Small.Z = -1/Large.Z           |    **
C**   +-------------------------------------------------------------+    **
      P=DABS(D(K+1)-D(K))/(E(K)+E(K))
      SHIFT=DMIN1(D(K),D(K+1))-E(K)/(P+DSIGN(DSQRT(P*P+1),P))
C**    ------------------------   -------------------------------     **
C**   |D(I-1) E(I-1)           | | D(I-1)  E(I-1)*C0 *S0          |    **
C**   |E(I-1)  D(I)  E0    F1  | |C0*E(I-1) D(I)-P0   G0          |    **
C**   |        E0    D1    G1  | |S0*E(I-1)    G0   F(I+1) G(I+1)|    **
C**   |        F1    G1  F(I+2)| |                   G(I+1) F(I+2)|    **
C**    -------< before >-------   ----------< after >------------     **
C**     C1,S1,P1,G1,D(I),E(I)  ==>   C0,S0,P0,G0,F(I+1),G(I+1)        **
C**     ---------------------------------------------------------     **
C**   | F1=E(I)*S1;    R1=SQRT(F1**2+G1**2);    G(I+1)=R1        |    **
C**   | E0=E(I)*C1;    C0=G1/R1;   S0=F1/R1;                     |    **
C**   | P0=(D0-D1)*S0**2+E0*2*C0*S0      =S0*T0                   |    **
C**   | G0=(D0-D1)*C0*S0+E0*(C0**2-S0**2)=C0*T0-E0                |    **
C**   | D0=D(I);   D1=D(I+1)-P1;   F(I+1)=D(I+1)-P1+P0=D1+P0      |    **
C**     ---------------------------------------------------------     **
C**   |     | C0 -S0 || D0  E0 || C0  S0 | ==> | D0-P0   G0  |     |    **
C**   |     | S0  C0 || E0  D1 ||-S0  C0 |     |  G0   D1+P0 |     |    **
C**   +-------------------------------------------------------------+    **
      G1=D(L)-SHIFT ! D(L-2) E(L-2)               !
      C1=1.0        ! E(L-2) D(L-1) E(L-1) E(L-1) ! => Force to get
      S1=1.0        !        E(L-1)  D(L)    G1   ! S0:C0=E(L-1):G1
      P1=0.0        !        E(L-1)   G1     XX   !
C**   -------------------------------------------------------------    **
      DO 60 I=L-1,K,-1
      E0=E(I)*C1
      F1=E(I)*S1
      R1=DSQRT(F1*F1+G1*G1)
      E(I+1)=R1
      IF(R1.EQ.0.0) THEN
        D(I+1)=D(I+1)-P1
        E(L)=0.0
        GO TO 70
      ENDIF
      C0=G1/R1
      S0=F1/R1
      D1=D(I+1)-P1
      T0=(D(I)-D1)*S0+2*C0*E0
      P0=S0*T0
      G0=C0*T0-E0
      D(I+1)=D1+P0
C**   +-------------------------------------------------------------+    **
C**   | Post multiply the transformation (rotation) matrix          |    **
C**   +-------------------------------------------------------------+    **
      DO 50 J=1,N
```

```
      Z0=Z(J,I)
      Z(J,I  )=Z0*C0-Z(J,I+1)*S0
   50 Z(J,I+1)=Z0*S0+Z(J,I+1)*C0
   60 CONTINUE
      D(K)=D(K)-P1
      E(K)=G0
      E(L)=0.0
      IF(T0.EQ.0.0) GO TO 80
   70 CONTINUE
      WRITE(*,'('' NO CONVERGE IN 30 ITERATIONS !'')')
      IERR=K
      RETURN
   80 CONTINUE
      RETURN
      END
************************************************************************
```

［表二］ 實數對稱矩陣之一般特徵值問題

```
************************************************************************
      SUBROUTINE AUEXBU(A,B,V,N,X,Y,IERR,EPSI,NV,NA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),V(1),X(N),Y(N)
C**   ============================================================    **
C**   DIMENSION A,B(NA,N),V(NV,N),X(N),Y(N)                           **
C**   ------------------------------------------------------------    **
C**   [A] {X} = d [B] {X}   --->   [A] [V] = [B] [V] [D]              **
C**   Normalize as : [V]'[A] [V] = [D],      [V]'[B] [V] = [I]        **
C**   ------------------------------------------------------------    **
C**   Input : A,B,V,N,EPSI,NV,NA         Output : A,V,X,IERR          **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = [A] matrix required upper triangular part only         **
C*O          = [D] diagonal matrix                                    **
CIW   B(N,N) = [B] matrix required upper triangular part only         **
C*O   V(N,N) = Eigen vector matrix                                    **
C*I   N      = Order of matrix [A]                                    **
C*O   X(N)   = Eigenvalues (also in the diagonal of [A])              **
C*W   Y(N)   = Working array                                          **
C*O   IERR   = J: J-th eigenvalue no converge; = 0: normal return     **
C*I   EPSI   = Used by GAG                                            **
C*I   NV     = Dummy (for JACOBI only)                                **
C*I   NA     = Row dimensions of array V(NA,N) and A(NA,N)            **
C**   ============================================================    **
      DO 20 J=1,N
      JV=(J-1)*NA
      DO 10 I=1,N
   10 V(I+JV)=0.0
   20 V(J+JV)=1.0
C**   +----------------------------------------------------------+    **
C**   | Solve [B] {Y} = e {Y}  for  [V] & [E]                     |    **
C**   +----------------------------------------------------------+    **
      CALL TRIDIG(NA,N,B,X,Y,V)
      CALL TRIQLI(NA,N,X,Y,V,IERR)
C**   +----------------------------------------------------------+    **
C**   | Compute : [V] [E]^{-1/2} ---> [V]                         |    **
C**   +----------------------------------------------------------+    **
      DO 40 J=1,N
      JV=(J-1)*NA
      IF(X(J).GT.0.0) THEN
```

```
        BB=1.0/DSQRT(X(J))
        DO 30 I=1,N
        V(I+JV)=V(I+JV)*BB
   30   CONTINUE
      ENDIF
   40 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Compute : [V]'[A] [V] ---> [A]                         |    **
C**   +--------------------------------------------------------+    **
      CALL MAV(A,B,A,V,X,N,N,NA,NA)
      CALL MAV(A,B,B,V,X,N,0,NA,NA)
C**   +--------------------------------------------------------+    **
C**   | Compute : [G]'[A] [G] ---> [A];  [V] [G] ---> [V]      |    **
C**   +--------------------------------------------------------+    **
      CALL GAG(A,B,V,N,X,EPSI,NA,NA)
C**   +--------------------------------------------------------+    **
C**   | Solve [A] {X} = d {X}  for  [V] & [D]                  |    **
C**   +--------------------------------------------------------+    **
      CALL TRIDIG(NA,N,A,X,Y,V)
      CALL TRIQLI(NA,N,X,Y,V,IERR)
      RETURN
      END
*************************************************************************
```

［表三］ 實數一般矩陣之標準特徵值問題

```
*************************************************************************
      SUBROUTINE ORTHEG(NA,N,ILW,IGH,A,D,Z)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NA,N),D(N),Z(NA,N)
C**   ============================================================    **
C**   Simularity transform real matrix to upper-Hessenberg matrix     **
C**   By Householder QR method                                        **
C**   ------------------------------------------------------------    **
C*I   ILW,IGH = Part of matrix considered is A(ILW:IGH,ILW:IGH)       **
C*I   A(N,N)  = [B] = Real matrix                                     **
C*O           = [H] = Upper Hessenberg matrix = [P]'[B][P]           **
C*O           + Information about the orthogonal transformations     **
C*O   D(N)    = Further informations about the transformations        **
C*I   Z(N,N)  = [I] or [X] = Matrix to be post-multiplied by [P]      **
C*O           = [P] or [X][P]                                        **
C**   ============================================================    **
      DO 80 K=ILW,IGH-2
      TK=0.0
      DO 20 I=K+1,IGH
   20 TK=TK+DABS(A(I,K))
      IF(TK.EQ.0.0) GO TO 80
C**   +--------------------------------------------------------+    **
C**   | Scale u-vector to avoid overflow, {u} = {w}sqrt(2) = {D} |    **
C**   +--------------------------------------------------------+    **
      S=0.0
      DO 25 I=K+1,IGH
      D(I)=A(I,K)/TK
   25 S=S+D(I)*D(I)
      S=DSIGN(DSQRT(S),D(K+1))
      D(K+1)=D(K+1)+S
      T=DSQRT(S*D(K+1))
      DO 30 I=K+1,IGH
   30 D(I)=D(I)/T
```

```
C**   +--------------------------------------------------------+    **
C**   | Compute  [A] = [[I] - 2{w}<w>][A] = [A] - {u}<u>[A]     |    **
C**   +--------------------------------------------------------+    **
      DO 40 J=K+1,N
      P=0.0
      DO 35 I=K+1,IGH
   35 P=P+D(I)*A(I,J)
      DO 40 I=K+1,IGH
   40 A(I,J)=A(I,J)-D(I)*P
C**   +--------------------------------------------------------+    **
C**   | Compute  [A] = [A][[I] - 2{w}<w>] = [A] - [A]{u}<u>     |    **
C**   +--------------------------------------------------------+    **
      DO 50 I=1,IGH
      P=0.0
      DO 45 J=K+1,IGH
   45 P=P+A(I,J)*D(J)
      DO 50 J=K+1,IGH
   50 A(I,J)=A(I,J)-P*D(J)
      A(K+1,K)=-S*TK
C**   +--------------------------------------------------------+    **
C**   | Compute  [Z] = [Z][[I] - 2{w}<w>] = [Z] - [Z]{u}<u>     |    **
C**   +--------------------------------------------------------+    **
      DO 60 I=1,IGH
      Q=0.0
      DO 55 J=K+1,IGH
   55 Q=Q+Z(I,J)*D(J)
      DO 60 J=K+1,IGH
   60 Z(I,J)=Z(I,J)-Q*D(J)
      D(K+1)=D(K+1)*TK*T
   80 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE HQRIDS(NA,N,ILW,IGH,H,WR,WI,Z,IERR)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(NA,N),WR(N),WI(N),Z(NA,N)
C**   ============================================================    **
C**   Upper-Hessenberg ==> Quasi-Upper-Triangular Transformed by      **
C**   QR algorithm with Implicit Double Shifts                        **
C**   Also update the transformation matrix Z(N,N)                    **
C**   ------------------------------------------------------------    **
C*I   H(N,N) = Upper Hessenberg matrix       : [H] = [P]'[B][P]       **
C*O          = Quasi upper triangular matrix : [T] = [V]'[B][V]       **
C*O   WR(N)  = Real part of eigenvalues                               **
C*O   WI(N)  = Imag part of eigenvalues                               **
C*I   Z(N,N) = Transformation matrix : [P] for [P]'[B][P]=[H]         **
C*O          = Transformation matrix : [V] for [V]'[B][V]=[T]         **
C*O   IERR   = J: J-th eigenvalue no converge; = 0: normal return     **
C**   ============================================================    **
      IERR=0
C**   +--------------------------------------------------------+    **
C**   | Get eigenvalues outside (ILW:IGH)                      |    **
C**   +--------------------------------------------------------+    **
      DO 20 J=1,N
      IF(J.LT.ILW.OR.J.GT.IGH) THEN
        WR(J)=H(J,J)
        WI(J)=0.0
      ENDIF
   20 CONTINUE
```

```
C**   +--------------------------------------------------------+    **
C**   | Find H(K-1,K)=0 for isolating H(K:L,K:L)                |    **
C**   +--------------------------------------------------------+    **
C**   | D  H  H  H  H  H  H  H | 1                             |    **
C**   | H  D  H  H  H  H  H  H |    This program transforms    |    **
C**   |    0  D  H  H  H  H  H | K                             |    **
C**   |       H  D  H  H  H  H |    Upper-Hessenber-matrix     |    **
C**   |          H  D  H  H  H |                               |    **
C**   |             H  D  H  H | L  ----< to >-----------------|    **
C**   |                0  D  H |                               |    **
C**   |                   X  D | N  Quasi-upper-triangular-matrix|    **
C**   +--------------------------------------------------------+    **
C**   DO 280 L=IGH,ILW,-1/-2
      L=IGH
  100 DO 270 ITER=1,30
      DO 120 K=L,ILW+1,-1
      DD=DABS(H(K-1,K-1))+DABS(H(K,K))
      IF(DD+DABS(H(K,K-1)).EQ.DD) GO TO 130
  120 CONTINUE
      K=ILW
  130 IF(K.EQ.L) THEN
        WR(K)=H(K,K)                  ! One real root
        WI(K)=0.0
        L=L-1
        GO TO 280
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | | H(L-1,L-1)-X H(L-1,L) | = X*X-2*SM*X+SP = (X-SM)**2-V  |    **
C**   | |  H(L,L-1)    H(L,L)-X |   2*SM = X1+X2  ;  SP = X1*X2  |    **
C**   |                         V =-(X1-SM)*(X2-SM) = SM*SM-SP |    **
C**   +--------------------------------------------------------+    **
      W =(H(L-1,L-1)-H(L,L))*0.5              ! W =(D-G)/2
      SM=W+H(L,L)                             ! SM=(D+G)/2
      SP=H(L-1,L-1)*H(L,L)-H(L,L-1)*H(L-1,L)  ! SP=D*G-E*F
      V=W*W+H(L,L-1)*H(L-1,L)                 ! V=W*W+E*F=SM*SM-SP
      IF(K.EQ.L-1) THEN
        IF(V.LT.0.0) THEN
          WR(K)=SM                    ! Two complex roots
          WR(L)=SM
          WI(K)=DSQRT(-V)
          WI(L)=-WI(K)
          L=L-2
          GO TO 280
        ELSE
          WR(K)=SM+DSIGN(DSQRT(V),W)  ! Two real roots
          WR(L)=SP/WR(K)
          WI(K)=0.0
          WI(L)=0.0
        ENDIF
C**     +--------------------------------------------------------+    **
C**     | Perform Jacobi transformation for H(L,K) = E ==> 0     |    **
C**     |    | C  S || D  F || C -S | ==> | DD  FF |             |    **
C**     |    |-S  C || E  G || S  C |     |  0  GG |             |    **
C**     |       E*C*C - (D-G)*C*S - F*S*S = 0                    |    **
C**     |    ==>  C/S = ((D-G)+SQRT((D-G)**2+4*E*F)/(2*E)        |    **
C**     +--------------------------------------------------------+    **
        C=W+DSIGN(DSQRT(V),W)         ! Transform for two real roots
        S=H(L,K)
        T=DABS(C)+DABS(S)
        C=C/T
```

```
        S=S/T
        T=DSQRT(C*C+S*S)
        C=C/T
        S=S/T
        DO 160 J=K,N
        T=H(K,J)
        H(K,J)= C*T+S*H(L,J)
        H(L,J)=-S*T+C*H(L,J)
  160   CONTINUE
        DO 170 J=1,L
        T=H(J,K)
        H(J,K)= T*C+H(J,L)*S
        H(J,L)=-T*S+H(J,L)*C
  170   CONTINUE
        DO 175 J=ILW,IGH
        T=Z(J,K)
        Z(J,K)= T*C+Z(J,L)*S
        Z(J,L)=-T*S+Z(J,L)*C
  175   CONTINUE
        L=L-2
        GO TO 280
      ENDIF
C**   +-----------------------------------------------------------+    **
C**   | Double shift QR will add 2 more sub-diagonals             |    **
C**   +-----------------------------------------------------------+    **
      DO 190 J=K,L-2
      DO 190 I=J+2,MIN0(J+3,L)
  190 H(I,J)=0.0
C**   +-----------------------------------------------------------+    **
C**   | {P,Q,R}*H(K+1,K) = 1st col of [H]**2-2*SM*[H]+SP*[I]      |    **
C**   +-----------------------------------------------------------+    **
      P=((H(K,K)-SM)**2-V)/H(K+1,K)+H(K,K+1)  ! V=SM*SM-SP
      Q=(H(K,K)-SM)+(H(K+1,K+1)-SM)           ! SM=(X1+X2)/2
      R=H(K+2,K+1)                            ! SP=X1*X2
C**   -------------------------------------------------------------    **
      DO 260 I=K,L-1
      T=DABS(P)+DABS(Q)+DABS(R)
      IF(T.NE.0.0) THEN
        P=P/T
        Q=Q/T
        R=R/T
C**     +-------------------------------------------------------+    **
C**     | Store new sub-diagonal element in H(I,I-1)            |    **
C**     +-------------------------------------------------------+    **
        S=DSIGN(DSQRT(P*P+Q*Q+R*R),P)
        IF(I.GT.K) H(I,I-1)=-S*T
C**     +-------------------------------------------------------+    **
C**     | Comput  {u}={P+S,Q,R}/S  ;  <v>=<P+S,Q,R>/(P+S)       |    **
C**     +-------------------------------------------------------+    **
        P=P+S
        PS=P/S
        QS=Q/S
        RS=R/S
        QP=Q/P
        RP=R/P
        IF(I.LT.L-1) THEN
C**       +---------------------------------------------------+    **
C**       | Compute  [H] = [[I] - 2{w}<w>][H] = [H] - {u}<v>[A]  |    **
C**       +---------------------------------------------------+    **
          DO 210 J=I,N
```

```
          T=H(I,J)+QP*H(I+1,J)+RP*H(I+2,J)
          H(I  ,J)=H(I  ,J)-PS*T
          H(I+1,J)=H(I+1,J)-QS*T
          H(I+2,J)=H(I+2,J)-RS*T
  210     CONTINUE
C**       +--------------------------------------------------------+    **
C**       | Compute  [H] = [H][[I] - 2{w}<w>] = [H] - [H]{u}<v>  |    **
C**       +--------------------------------------------------------+    **
          DO 220 J=1,MIN0(I+3,L)
          T=H(J,I)*PS+H(J,I+1)*QS+H(J,I+2)*RS
          H(J,I  )=H(J,I  )-T
          H(J,I+1)=H(J,I+1)-T*QP
          H(J,I+2)=H(J,I+2)-T*RP
  220     CONTINUE
C**       +--------------------------------------------------------+    **
C**       | Compute  [Z] = [Z][[I] - 2{w}<w>] = [Z] - [Z]{u}<v>  |    **
C**       +--------------------------------------------------------+    **
          DO 225 J=ILW,IGH
          T=Z(J,I)*PS+Z(J,I+1)*QS+Z(J,I+2)*RS
          Z(J,I  )=Z(J,I  )-T
          Z(J,I+1)=Z(J,I+1)-T*QP
          Z(J,I+2)=Z(J,I+2)-T*RP
  225     CONTINUE
        ELSE
          DO 230 J=I,N
          T=H(I,J)+QP*H(I+1,J)
          H(I  ,J)=H(I  ,J)-PS*T
          H(I+1,J)=H(I+1,J)-QS*T
  230     CONTINUE
          DO 240 J=1,MIN0(I+3,L)
          T=H(J,I)*PS+H(J,I+1)*QS
          H(J,I  )=H(J,I  )-T
          H(J,I+1)=H(J,I+1)-T*QP
  240     CONTINUE
          DO 245 J=ILW,IGH
          T=Z(J,I)*PS+Z(J,I+1)*QS
          Z(J,I  )=Z(J,I  )-T
          Z(J,I+1)=Z(J,I+1)-T*QP
  245     CONTINUE
        ENDIF
      ENDIF
C**   +--------------------------------------------------------------+    **
C**   | Get {P,Q,R} for next transformation                          |    **
C**   +--------------------------------------------------------------+    **
C**   IF(I.EQ.L-1) GO TO 260
      P=H(I+1,I)
      Q=H(I+2,I)
      R=0.0
      IF(I+3.LE.L) R=H(I+3,I)
  260 CONTINUE
  270 CONTINUE
      WRITE(*,'('' NOT CONVERGE IN 30 ITERATIONS !'')')
      IERR=L
      RETURN
  280 IF(L.GE.ILW) GO TO 100
      RETURN
      END
*****************************************************************************
      SUBROUTINE QUTEVE(NA,N,ILW,IGH,H,WR,WI,Z)
C**   ==========================================================    **
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(NA,N),WR(N),WI(N),Z(NA,N)
      DATA EPSMAH/1.0D-15/
C**   ================================================================    **
C**   Find Eigenvectors of Quasi-Upper-Triangular matrix by               **
C**   Solving the singular matrix system equations                        **
C**   ----------------------------------------------------------------    **
C*I   H(N,N) = Quasi upper triangular matrix : [T] = [V]'[B][V]           **
C*O          = Transformation matrix : [U] for [U]^{-1}[T][U]=[S]        **
C*I   WR(N)  = Real part of eigenvalues                                   **
C*I   WI(N)  = Imag part of eigenvalues                                   **
C*I   Z(N,N) = [V] : Transformation matrix for     [V]'[B][V]=[T]         **
C*O          = [Z] = [V][U] : Tran. matrix for [Z]^{-1}[B][Z]=[S]        **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Compute norm HNORM                                           |    **
C**   +--------------------------------------------------------------+    **
      HNORM=0.0
      DO 20 J=1,N
      DO 20 I=1,MIN0(J+1,N)
   20 HNORM=HNORM+DABS(H(I,J))
C**   +--------------------------------------------------------------+    **
C**   | Get  WR(N),WI(N) : Eigen values                              |    **
C**   +--------------------------------------------------------------+    **
      DO 500 L=N,1,-1
      XR=WR(L)
      XI=WI(L)
      IF(XI.EQ.0.0) THEN
C**     +--------------------------------------------------------+    **
C**     | Eigenvector of real eigenvalue                         |    **
C**     +--------------------------------------------------------+    **
        H(L,L)=1.0
        DO 300 I=L-1,1,-1
        W=H(I,I)-XR
        R=0.0
        I1=I+1
        IF(WI(I).GT.0.0) I1=I+2
        DO 250 J=I1,L
        R=R-H(I,J)*H(J,L)
  250   CONTINUE
        IF(WI(I).EQ.0.0) THEN
          IF(W.EQ.0.0) W=HNORM/EPSMAH
          H(I,L)=R/W
        ELSE IF(WI(I).LT.0.0) THEN
          W1=W
          R1=R
        ELSE
C**       +--------------------------------------------------------+    **
C**       | |    W     H(I,I+1) || H(I   ,L) | = | R  |           |    **
C**       | | H(I+1,I)    W1    || H(I+1,L) |   | R1 |           |    **
C**       +--------------------------------------------------------+    **
          Q=(WR(I)-XR)**2+WI(I)**2      ! Q=W*W1-H(I,I+1)*H(I+1,I)
          H(I   ,L)=(W1*R-H(I,I+1)*R1)/Q
          H(I+1,L)=(W*R1-H(I+1,I)*R)/Q
        ENDIF
  300   CONTINUE
      ELSE IF(XI.LT.0.0) THEN
C**     +--------------------------------------------------------+    **
C**     | Eigenvectors of complex eigenvalues                    |    **
C**     +--------------------------------------------------------+    **
```

```
C**     | | HKK-XR+XI*i     HKL      || UKK+UKL*i | = | 0 |       |    **
C**     | |    HLK       HLL-XR+XI*i || ULK+ULL*i |   | 0 |       |    **
C**     |-------------------------------------------------------|    **
C**     | | 1  0 || UKK+UKL*i | = | -((HLL-XR)+XI*i)*i/HLK |      |    **
C**     | | 0  1 || ULK+ULL*i |   |              0+1*i     |      |    **
C**     |-------------------------------------------------------|    **
C**     | | 1  0 || UKK+UKL*i | = | -HKL*i/((HKK-XR)+XI*i) |      |    **
C**     | | 0  1 || ULK+ULL*i |   |  0+1*i                 |      |    **
C**     +-------------------------------------------------------+    **
        K=L-1
        IF(DABS(H(L,K)).GT.DABS(H(K,L))) THEN
          H(K,K)=XI/H(L,K)
          H(K,L)=-(H(L,L)-XR)/H(L,K)
        ELSE
          CALL CDIV(0.0D0,-H(K,L),H(K,K)-XR,XI,H(K,K),H(K,L))
        ENDIF
        H(L,K)=0.0
        H(L,L)=1.0
C**     ---------------------------------------------------------    **
        DO 400 I=L-2,1,-1
        W=H(I,I)-XR
        S=0.0
        R=0.0
        I1=I+1
        IF(WI(I).GT.0.0) I1=I+2
        DO 350 J=I1,L
        S=S-H(I,J)*H(J,K)
        R=R-H(I,J)*H(J,L)
  350   CONTINUE
        IF(WI(I).EQ.0.0) THEN
          CALL CDIV(S,R,W,XI,H(I,K),H(I,L))
        ELSE IF(WI(I).LT.0.0) THEN
          W1=W
          S1=S
          R1=R
        ELSE
C**       +-------------------------------------------------+    **
C**       | W+XI*i  H(I,I+1) || H(I  ,K)+H(I  ,L)i | = | S +R *i |    **
C**       | H(I+1,I) W1+XI*i || H(I+1,K)+H(I+1,L)i |   | S1+R1*i |    **
C**       +-------------------------------------------------+    **
          QR=(WR(I)-XR)**2+WI(I)**2-XI**2 ! W*W1-H(I,I+1)*H(I+1,I)-XI*XI
          QI=(WR(I)-XR)*XI*2.0            ! (W+W1)*XI
          CALL CDIV(W1*S-H(I,I+1)*S1-XI*R,W1*R-H(I,I+1)*R1+XI*S
     *                                   ,QR,QI,H(I,K),H(I,L))
          CALL CDIV(W*S1-H(I+1,I)*S-XI*R1,W*R1-H(I+1,I)*R+XI*S1
     *                                   ,QR,QI,H(I+1,K),H(I+1,L))
        ENDIF
  400   CONTINUE
      ENDIF
  500 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Compute [Z] = [V][U] = V(ILW:IGH,ILW:IGH)*U(ILW:IGH,I:N) |    **
C**   +-------------------------------------------------------+    **
      DO 520 I=N,1,-1
      IF(I.GE.ILW.AND.I.LE.IGH) GO TO 520
      DO 510 J=I,N
  510 Z(I,J)=H(I,J)
  520 CONTINUE
      DO 550 J=N,1,-1
      DO 550 I=ILW,IGH
```

```
      S=0.0
      DO 530 K=ILW,MIN0(J,IGH)
  530 S=S+Z(I,K)*H(K,J)
      Z(I,J)=S
  550 CONTINUE
      RETURN
      END
*******************************************************************************
```

［表四］ 實數一般矩陣之一般特徵值問題

```
*******************************************************************************
      SUBROUTINE AZEXBZ(NA,N,A,B,D,E,F,Z,IERR)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NA,N),B(NA,N),D(N),E(N),F(N),Z(NA,N)
C**   ============================================================    **
C**   Find the eigenvalues and eigenvectors of [A]{Z} = e [B]{Z}      **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = Given real matrix [A]                                  **
C*O          = [Y] in upper triangular part (no need for user)        **
C*I   B(N,N) = Given real matrix [B]                                  **
C*O          = [U] in upper triangular part (no need for user)        **
C*O   D(J)   = Real part of J-th eigenvalue (if F(J).NE.0)            **
C*O   E(J)   = Imag part of J-th eigenvalue (if F(J).NE.0)            **
C*O   F(J)   = Absolute value of J-th eigenvalue of [B]{X}=e{X}       **
C**           = 0 & D(J)+E(J)i.NE.0 then J-th e-value is infinity     **
C**           = 0 & D(J)+E(J)i.EQ.0 then J-th e-value is arbitary     **
C*O   Z(N,J) = Z(1:N,J) = eigenvector of real eigenvalue D(J)         **
C**            Z(1:N,J)+Z(1:N,J+1)i = eigenvector of D(J)+E(J)i      **
C**            Z(1:N,J)-Z(1:N,J+1)i = eigenvector of D(J)-E(J)i      **
C*O   IERR   = J: J-th eigenvalue no converge; = 0: normal return     **
C**   ------------------------+-----------------------------------    **
C**           B               |              A  Z                     **
C**   ------------------------+-----------------------------------    **
C**      [V]'[B][V] = [T]     |          [V]'[A][V] = [C]             **
C**   [u][V]'[B][V][U] = [S]  |       [u][V]'[A][V][U]                **
C**    [k][d][S][d] = [J]   /   [k][d][u][V]'[A][V][U][d] = [G]       **
C**       [L][J][R] = [J] /  [L][k][d][u][V]'[A][V][U][d][R]          **
C**              [Qut] = [Z]'[L][k][d][u][V]'[A][V][U][d][R][Z]       **
C**           [Qdg] = [y][Z]'[L][k][d][u][V]'[A][V][U][d][R][Z][Y]    **
C**   ------------------------------------------------------------    **
C**     Note : [y],[k],[d],[u] are inverses of [Y],[K],[D],[U]        **
C**          : [V],[Z] are orthogonal matrices                        **
C**          : [U],[Y] are upper triangular matrices                  **
C**          : [D],[d] are diagonal matrices                          **
C**          : [K],[k] are quasi-diagonal orthogonal matrices         **
C**          : [L],[R] are static condensation matrices               **
C**          : [T],[Qut] are quasi-upper triangular matrices          **
C**          : [S],[Qdg] are quasi-diagonal matrices                  **
C**          : [J] is quasi-identity matrix : [V]'[J][V] .NE. [J]     **
C**          : [J] = [y][Z]'[J][Z][Y] only for restrict [Z] & [Y]     **
C**   ============================================================    **
      DO 20 J=1,N
      DO 15 I=1,N
   15 Z(I,J)=0.0
   20 Z(J,J)=1.0
      CALL ORTHEG(NA,N,1,N,B,D,Z)         !      [ ]'[B][ ]=[H]
      CALL HQRIDS(NA,N,1,N,B,D,E,Z,IERR) !      [V]'[B][V]=[T]
      IF(IERR.NE.0) RETURN                !
```

```
      CALL ZTAZ  (A,F,Z,N,NA)            !      [V]'[A][V]=[C]
      CALL QUTEVE(NA,N,1,N,B,D,E,Z)      !       [u][T][U]=[S]
      CALL QUTGEN(A,B,D,E,F,Z,N,NA)      ! [k][d][u][C][U][d]=[G]
      CALL STCOND(A,D,E,F,Z,N,NA)        !       [L][G][R]
      CALL ORTHEG(NA,N,1,N,A,D,Z)        !   [ ]'[L][G][R][ ]=[Hes]
      CALL HQRIDS(NA,N,1,N,A,D,E,Z,IERR) !   [Z]'[L][G][R][Z]=[Tri]
      CALL QUTEVE(NA,N,1,N,A,D,E,Z)      ! [V][U][d][R][Z][Y]=[Eve]
      RETURN   ! [y][Z]'[L][k][d][u][V]'[A][V][U][d][R][Z][Y]=[Eva]
      END
*****************************************************************************
      SUBROUTINE QUTGEN(A,U,WR,WI,F,Z,N,NA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NA,N),U(NA,N),WR(N),WI(N),F(N),Z(NA,N)
C**   ============================================================    **
C**   Compute  [G] = [K]^{-1}[D]^{-1}[U]^{-1}[C][U][D]^{-1}           **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = [C] = [V]'[A][V]                                       **
C*O          = [G] = [k][d][u][V]'[A][V][U][d]                        **
C*I   U(N,N) = [U] : Trans. matrix for [U]^{-1}[V]'[B][V][U] = [S]    **
C*I   WR(N)  = Real part of eigenvalues                               **
C*I   WI(N)  = Imag part of eigenvalues                               **
C*O   F(N)   = Sqrt(WR(N)**2+WI(N)**2)                                **
C*I   Z(N,N) = [V][U] : Trans. matrix for [u][V]'[B][V][U] = [S]      **
C*O          = [V][U][d] : [d] = [D]^{-1} for [S] = [D][K][J][D]      **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Compute [A] = [A][U] : [U] is an upper triangular matrix |    **
C**   +----------------------------------------------------------+    **
      DO 550 J=N,1,-1
      DO 550 I=1,N
      AIJ=0.0
      DO 530 K=1,J
  530 AIJ=AIJ+A(I,K)*U(K,J)
  550 A(I,J)=AIJ
C**   +----------------------------------------------------------+    **
C**   | Compute [A] = [U]^{-1}[A] : By backward substitutions    |    **
C**   +----------------------------------------------------------+    **
      DO 580 J=1,N
      DO 580 K=N,1,-1
      AKJ=A(K,J)/U(K,K)
      DO 560 I=1,K-1
  560 A(I,J)=A(I,J)-U(I,K)*AKJ
  580 A(K,J)=AKJ
C**   +----------------------------------------------------------+    **
C**   | Compute [A] = [D]^{-1}[A][D]^{-1} ; [Z] = [Z][D]^{-1}    |    **
C**   +----------------------------------------------------------+    **
      DO 650 J=1,N
      F(J)=CDABS(DCMPLX(WR(J),WI(J)))
      IF(F(J).EQ.0.0) GO TO 650
      DJ=1.0/DSQRT(F(J))
      DO 620 I=1,N
      Z(I,J)=Z(I,J)*DJ
      A(I,J)=A(I,J)*DJ
  620 A(J,I)=A(J,I)*DJ
  650 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Compute [A] = [K]^{-1}[A]                                |    **
C**   +----------------------------------------------------------+    **
      DO 680 K=1,N
```

```
      IF(WI(K).NE.0.0) THEN
        IF(WI(K).LT.0.0) GO TO 680
        C=WR(K)/F(K)      ! Complex eigenvalue of [B]
        S=WI(K)/F(K)
        DO 660 J=1,N
        AKJ=A(K,J)
        A(K  ,J)=C*AKJ-S*A(K+1,J)
  660   A(K+1,J)=S*AKJ+C*A(K+1,J)
      ELSE IF(WR(K).LT.0.0) THEN
        DO 670 J=1,N      ! Negative real eigenvalue of [B]
  670   A(K,J)=-A(K,J)
      ENDIF
  680 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE STCOND(A,D,E,F,Z,N,NA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NA,N),D(N),E(N),F(N),Z(NA,N)
C**   ============================================================    **
C**   Perform static condensations  [L][G][R] ==> [A]                 **
C**   ------------------------------------------------------------    **
C*I   A(N,N) = [G] = [k][d][u][V]'[A][V][U][d] : matrix to be cond.   **
C*O          = [L][G][R] = [L][k][d][u][V]'[A][V][U][d][R]           **
C*W   D,E(N) = Working array                                         **
C*I   F(N)   = Absolute value of eigenvalue                          **
C*I   Z(N,N) = [V][U][d] : for [k][d][u][V]'[B][V][U][d] = [J]        **
C*O          = [V][U][d][R]                                         **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Compute  [A] = [L][G][R]                                 |    **
C**   |   ------[ L ]------     ------------[ R ]-------------   |    **
C**   |  | 1   -g13/g33    |    |    1                         |  |    **
C**   |  |   1 -g23/g33    |    |            1                |  |    **
C**   |  |         1       |    |-g31/g33 -g32/g33  1 -g34/g33 |  |    **
C**   |  |     -g43/g33  1 |    |                        1     |  |    **
C**   +----------------------------------------------------------+    **
      DO 800 K=1,N
      IF(F(K).NE.0.0) GO TO 800
      IF(A(K,K).EQ.0.0) THEN
        S=0.0
        DO 705 I=1,N
        S=S+DABS(A(I,K)*A(K,I))
        A(I,K)=0.0
  705   A(K,I)=0.0
        A(K,K)=1.0
        IF(S.NE.0.0) WRITE(*,'('' MATRIX IS NOT WELLPOSED!'')')
        GO TO 800
      ENDIF
      DO 710 I=1,N
      E(I)=Z(I,K)
      D(I)=A(I,K)
  710 A(I,K)=0.0
      A(K,K)=D(K)
C**   +----------------------------------------------------------+    **
C**   | Compute  [A] = [L][G][R]  and  [Z] = [Z][R]               |    **
C**   +----------------------------------------------------------+    **
      DO 760 J=1,N
      IF(J.EQ.K) GO TO 760
```

```
      AKJ=A(K,J)/A(K,K)
      DO 720 I=1,N
      Z(I,J)=Z(I,J)-E(I)*AKJ
  720 A(I,J)=A(I,J)-D(I)*AKJ
      A(K,J)=0.0
  760 CONTINUE
  800 CONTINUE
      RETURN
      END

*************************************************************************
      SUBROUTINE ZTAZ(A,W,Z,N,NA)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NA,N),W(N),Z(NA,N)
C**   ================================================================    **
C**   Compute  [A] = [Z]'[A][Z]                                           **
C**   ----------------------------------------------------------------    **
C*I   A(N,N) = [A]                                                        **
C*O          = [Z]'[A][Z]                                                 **
C*W   W(N)   = Working array                                              **
C*I   Z(N,N) = Orthogonal matrix                                          **
C**   ================================================================    **
      DO 30 I=1,N
      DO 10 K=1,N
   10 W(K)=A(I,K)
      DO 20 J=1,N
      S=0.0
      DO 15 K=1,N
   15 S=S+W(K)*Z(K,J)
   20 A(I,J)=S
   30 CONTINUE
      DO 80 J=1,N
      DO 60 K=1,N
   60 W(K)=A(K,J)
      DO 70 I=1,N
      S=0.0
      DO 65 K=1,N
   65 S=S+Z(K,I)*W(K)
   70 A(I,J)=S
   80 CONTINUE
      RETURN
      END
*************************************************************************
```

習題

1. 試寫一程式呼叫實數特徵值之求解程式以求解複數特徵值問題。可針對標準型或一般型為之。

2. 試寫一程式呼叫標準特徵值之求解程式以求解一般特徵值問題。可針對不對稱矩陣或對稱矩陣為之。

3. 試就QR法之收斂階段（即$a_{11} > a_{22}$）証明：當$a_{21} << a_{11}$時，$r_{21}/a_{21} \approx a_{22}/a_{11}$。注：$QR$法之$\theta_R$由$-a_{11}\sin\theta_R + a_{21}\cos\theta_R = 0$之關係求得，而$r_{21} = a_{21}\cos^2\theta_R - a_{12}\sin^2\theta_R + (a_{22} - a_{11})\cos\theta_R\sin\theta_R$。

4. 令$[H] = [Q]^T[A][Q]$，$[H]$為下赫申伯格矩陣，$[Q]$為正交矩陣。若已知$[Q]$之第一行$\{q_1\}$與矩陣$[A]$，試求出$[Q]$矩陣及$[H]$矩陣。

5. 該寫一副程式計算下赫申伯格矩陣之特性方程式之係數。

參考文獻

1. Wilkinson, J. H. and Reinsch, C., *Linear Algebra,* Vol.II of *Handbook for Automatic Computation,* New York, Springer-Veriag, 1971.

2. Moler, C. B. and Stewart, G. W., "An Algorithm for Generalized Matrix Eigenvalue Problems," SIAM Journal on Numerical Analysis, Vol.10, No.2, pp.241-256, 1973.

3. Smith, B. T., et al., *Matrix Eigensystem Routines – EISPACK Guide,* 2nd ed., Vol.6 of Lecture Notes in Computer Science, New York, Springer-Veriag, 1976.

4. Garbow, B. S., Boyle, J. M., Dongarra, J. J. and Moler, C. B., *Matrix Eigensystem Routines – EISPACK Guide Extension,* New York, Springer-Veriag, 1977.

第八章

函數插值法

8.1 前言

在許多問題上常需要由如[表一]之一組數據中求得任一已知x所對應之y值，最簡單之做法為常用於查各種函數表之直線內插法：

$$y = y_i + (x - x_i)(y_{i+1} - y_i)/(x_{i+1} - x_i) \tag{8.1}$$

[表一]　已知函數表

x_i	x_o	x_1	x_2	$\cdots$	x_n
y_i	y_o	y_1	y_2	$\cdots$	y_n

其中$x_i \le x \le x_{i+1}$，若x不在x_i與x_{i+1}之間，則稱為直線外插法。

直線內插法視二已知點間以直線連接之，中間各x值則按此直線函數計算y值。一般說來，通過$(n+1)$點有一唯一之n次多項式函數曲線：

$$y = c_n x^n + \cdots + c_1 x + c_o \tag{8.2}$$

該函數中之$c_o,\ c_1, \cdots,\ c_n$等$(n+1)$個係數可由已知之$(n+1)$點，即$(n+1)$組(x_i, y_i)值，代入上式即得$(n+1)$個含$c_o,\ c_1, \cdots,\ c_n$等$(n+1)$個未知數之聯立線性方程式：

$$\begin{bmatrix} x_o^n & \cdots & x_o & 1 \\ x_1^n & \cdots & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^n & \cdots & x_n & 1 \end{bmatrix} \begin{Bmatrix} c_n \\ \vdots \\ c_1 \\ c_o \end{Bmatrix} = \begin{Bmatrix} y_o \\ y_1 \\ \vdots \\ y_n \end{Bmatrix} \tag{8.3}$$

解該聯立方程式即可得諸c_i值，並進而可計算任一x值對應之y。本章將介紹之牛頓(Newton)插值公式可不必經由上述解聯立方程式之過程而求得各係數。但一般應用常不求插值公式之係數，而直接針對某一已知x值計算對應之y值。除牛頓插值法外，本章同時亦介紹：拉格蘭治(Lagrange)多項式及對應之插值法，與常用之葉特肯(Aitken)插值法。有時利用有理函數 (Rational function)做插值函數可得較理想之結果，本章將介紹一種類似牛頓插值法之紀雷(Thiele)插值法。在作圖上常需用低階函數求得較平順之近似曲線，則可用樑(spline)曲線近似法。

8.2 拉格蘭治多項式

考慮下列x之3次函數

$$L_1(x) = \frac{(x-x_o)(x-x_2)(x-x_3)}{(x_1-x_o)(x_1-x_2)(x_1-x_3)} \tag{8.4}$$

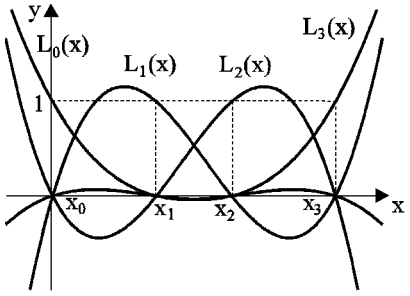
該函數在$x = x_o,\ x_1,\ x_2,\ x_3$時分別為$0,\ 1,\ 0,\ 0$各值，即當$i \neq 1$時，$L_1(x_i) = 0$，而$L_1(x_1) = 1$，其函數曲線如[圖一]所示。同理可寫出下列$n$次函數

$$\begin{aligned} L_k(x) &= \frac{(x-x_o)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_o)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \\ &= \prod_{\substack{i=0 \\ i\neq k}}^{n} \left(\frac{x-x_i}{x_k-x_i}\right) \end{aligned} \tag{8.5}$$

而有下列特性

$$\begin{aligned} L_k(x_i) &= 0\ , \qquad i = 0, 1, \cdots, k-1, k+1, \cdots, n \\ L_k(x_k) &= 1 \end{aligned} \tag{8.6}$$

式(8.5)稱為拉格蘭治多項式。



圖一 拉格蘭治函數

8.3 拉格蘭治插值法

由$L_k(x)$之簡單特性不難看出通過[表一]之$(n+1)$點之n次多項式函數曲線可直接寫成

$$\begin{aligned} y(x) &= y_o L_o(x) + y_1 L_1(x) + \cdots + y_n L_n(x) = \sum_{k=0}^{n} y_k L_k(x) \\ &= \sum_{k=0}^{n} y_k \left[\prod_{\substack{i=0 \\ i \neq k}}^{n} \left(\frac{x - x_i}{x_k - x_i} \right) \right] \end{aligned} \tag{8.7}$$

上式代以$x = x_i$時，$L_k(x_i) = 0,\ k \neq i$，而$L_i(x_i) = 1$，因此得

$$y(x_i) = \sum_{k=0}^{n} y_k L_k(x_i) = y_i L_i(x_i) = y_i \tag{8.8}$$

式(8.7)即為拉格蘭治插值公式。將其各項展開再合併可得式(8.2)之係數，但此展開手續仍相當麻煩，一般均以x值代入式(8.7)直接計算函數值。

[表二]之副程式$LAGRAG(X,Y,N,XX,YY,M)$為利用式(8.7)寫成。其中$X(N),Y(N)$為已知之N組數據，$X(1)$至$X(N)$須由小至大或由大至小排列，M為拉格蘭治插值曲線欲通過之點數。亦即用於插值公式之數據點數，故插值函數曲線為$(M-1)$次多項式函數。

由於插值函數對於較接近x_o至x_n之中間部分之x值計算所得之y值較為準確。故該副程式自動選擇M點數據，儘可能使小於x與大於x之$X(i)$值各佔一半。XX為已知之x值，YY為副程式以XX內插算得之y值。

［表二］ 拉格蘭治內插副程式

```
************************************************************************
      SUBROUTINE LAGRAG(X,Y,N,XX,YY,M)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),Y(N)
C**   ================================================================    **
C**   Lagrange's interpolation formula
C**
C**   F(x) = \sum_{k=IA}^IB \prod_{i=IA}^IB (X-X(i))/(X(k)-X(i))
C**                                 i!=k
C**   ================================================================    **
C**   +------------------------------------------------------------+    **
C**   | X(I) must be in ascending or descending order              |    **
C**   | If M >= N or M <= 1 then use all data : IA=1, IB=N         |    **
C**   +------------------------------------------------------------+    **
      MM=M/2
```

```
      IA=1
      IB=N
      IF(M.GE.N.OR.M.LE.1) GO TO 30
C**   +-------------------------------------------------------------+    **
C**   | Find IA and IB such that: 1 <= IA and IB=IA+M-1 <= N and     |    **
C**   | #{X(IA:IB) < XX} - #{X(IA:IB) > XX} as small as possible    |    **
C**   +-------------------------------------------------------------+    **
      PP=DSIGN(1.0D0,X(N)-X(1))
      DO 10 I=1,N
      IF(XX*PP.LE.X(I)*PP) GO TO 20
   10 CONTINUE
      I=N
   20 IA=MAX0(I-MM,1)
      IB=MIN0(IA+M-1,N)
      IA=IB-M+1
C**   ---------------------------------------------------------------    **
   30 YY=0.0
      DO 60 K=IA,IB
      PP=Y(K)
      DO 50 I=IA,IB
      IF(I.NE.K) PP=PP*(XX-X(I))/(X(K)-X(I))
   50 CONTINUE
   60 YY=YY+PP
      RETURN
      END
*******************************************************************************
```

[表三]主程式用以讀入下列7組數據，並讀入欲求內插值之諸XX及欲採用之點數M，再呼叫副程式$LAGRAG$算出內插值YY。程式後為輸入資料及對應之計算結果。

x_i	0.3	0.4	0.5	0.6	0.7	0.8	0.9
y_i	−1.203973	−0.916291	−0.693147	−0.510826	−0.356675	−0.223144	−0.105361

[表三] 拉格蘭治內插主程式及輸入數據與結果

```
*******************************************************************************
C**   TEST PROGRAM FOR LAGRANGE'S INTERPOLATION                             **
C**   ===================================================================   **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(20),Y(20)
C**   ===================================================================   **
   10 READ(*,'(I5)') N
      READ(*,'(2F10.0)') (X(I),Y(I),I=1,N)
      WRITE(*,'(9X,''XX'',11X,''YY'',9X,''M'')')
   30 READ(*,'(F10.0,I5)') XX,M
      IF(M.EQ.0.0) GO TO 10
      CALL LAGRAG(X,Y,N,XX,YY,M)
      WRITE(*,'(3X,1P2E13.5,I5)') XX,YY,M
      GO TO 30
      END
*******************************************************************************
```

```
   7
0.3       -1.203973
0.4       -0.916291
0.5       -0.693147
0.6       -0.510826
0.7       -0.356675
0.8       -0.223144
0.9       -0.105361            |         XX           YY          M
0.43          2                |    4.30000E-01 -8.49348E-01     2
0.43          3                |    4.30000E-01 -8.45061E-01     3
0.43          5                |    4.30000E-01 -8.43864E-01     5
0.43          7                |    4.30000E-01 -8.43949E-01     7
```

8.4 除式差分表

對一係列函數$f(x_o)$, $f(x_1)$, $f(x_2)$,$\cdots$定義各階除式差分 (Divided Difference) 如下：

零階除式差分：

$$f[x_o] = f(x_o) \tag{8.9}$$

一階除式差分：

$$\begin{aligned} f[x_o, x_1] &= \frac{f(x_o) - f(x_1)}{x_o - x_1} \\ f[x_1, x_2] &= \frac{f(x_1) - f(x_2)}{x_1 - x_2} \\ f[x_2, x_3] &= \frac{f(x_2) - f(x_3)}{x_2 - x_3} \end{aligned} \tag{8.10}$$

二階除式差分：

$$\begin{aligned} f[x_o, x_1, x_2] &= \frac{f[x_o, x_1] - f[x_1, x_2]}{x_o - x_2} \\ f[x_1, x_2, x_3] &= \frac{f[x_1, x_2] - f[x_2, x_3]}{x_1 - x_3} \end{aligned} \tag{8.11}$$

三階除式差分：

$$f[x_o, x_1, x_2, x_3] = \frac{f[x_o, x_1, x_2] - f[x_1, x_2, x_3]}{x_o - x_3} \tag{8.12}$$

其一般式可寫成：

$$f[x_n, x_m, \overbrace{x_i, x_j, \cdots}^{\text{相同}}] = \frac{f[x_n, \overbrace{x_i, x_j, \cdots}^{\text{相同}}] - f[x_m, \overbrace{x_i, x_j, \cdots}^{\text{相同}}]}{x_n - x_m} \tag{8.13}$$

注意$f[x_i, x_j, x_k]$內諸x值不必按一定順序，即$f[x_i, x_j, x_k] = f[x_j, x_i, x_k] = f[x_j, x_k, x_i] = \cdots$。且可以有許多算法，即

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_k] - f[x_j, x_k]}{x_i - x_j} = \frac{f[x_i, x_j] - f[x_k, x_j]}{x_i - x_k} = \frac{f[x_j, x_i] - f[x_k, x_i]}{x_j - x_k}$$

根據以上定義可做下列除式差分表：

［表四］ 除式差分表

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot,\cdot]$
x_o	$f[x_o]$				
		$f[x_o, x_1]$			
x_1	$f[x_1]$		$f[x_o, x_1, x_2]$		
		$f[x_1, x_2]$		$f[x_o, x_1, x_2, x_3]$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$		$f[x_o, x_1, x_2, x_3, x_4]$
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	$f[x_4]$				

8.5 牛頓插值公式

根據除式差分定義可得下列各式：

$$\frac{f(x) - f(x_o)}{x - x_o} = f[x, x_o] \tag{8.14}$$

或

$$f(x) = f(x_o) + (x - x_o) f[x, x_o] \tag{8.15}$$

$$f[x, x_o] = f[x_o, x_1] + (x - x_1) f[x, x_o, x_1] \tag{8.16}$$

$$f[x, x_o, x_1] = f[x_o, x_1, x_2] + (x - x_2) f[x, x_o, x_1, x_2] \tag{8.17}$$

$\cdots$

$$f[x, x_o, \cdots, x_{n-1}] = f[x_o, x_1, \cdots, x_n] + (x - x_n) f[x, x_o, \cdots, x_n] \tag{8.18}$$

將式(8.16)以後各式代入式(8.15)即得下列之牛頓插值公式。

$$f(x) = F_n(x) + E_n(x) \tag{8.19}$$

$$F_n(x) = f(x_o) + (x - x_o)f[x_o, x_1] + (x - x_o)(x - x_1)f[x_o, x_1, x_2] + \cdots + (x - x_o)(x - x_1)\cdots(x - x_{n-1})f[x_o, x_1, \cdots, x_n] \tag{8.20}$$

$$E_n(x) = (x - x_o)(x - x_1)\cdots(x - x_{n-1})(x - x_n)f[x, x_o, x_1, \cdots, x_n] \tag{8.21}$$

由上式知 $E_n(x_o) = E_n(x_1) = \cdots = E_n(x_n) = 0$ 。故 $F_n(x_i) = f(x_i)$ ，$i = 0, 1, \cdots, n$ ，且 $F_n(x)$ 為 x 之 n 次多項式，因此為滿足 $(n+1)$ 個已知函數 $f(x_o), f(x_1), \cdots, f(x_n)$ 之唯一 n 次多項式，應與由拉格蘭治多項式求得之函數相同。

利用除式差分表做插值的好處為可以根據所需精度，決定做幾階除式差分。一旦插入值達到所需精確 (以最後加入項之值判斷) 即可停止，否則，可再增加一已知函數值，繼續算出高一階除式差分後乘以 $(x - x_o)(x - x_1)\cdots$ 之值，再加入原先算得之插入值。

以下表為例，由 $\ln 0.3, \ln 0.4, \ln 0.5, \cdots$ 內插計算 $\ln 0.43$ 之值，利用四階除式差分可得：

$$\begin{aligned} f(x) &= -1.203973 + (0.13)(2.87682) + (0.13)(0.03)(-3.22690) \\ &\quad +(0.13)(0.03)(-0.07)(3.95250) \\ &\quad +(0.13)(0.03)(-0.07)(-0.17)(-4.60918) \\ &= -1.203973 + 0.373987 - 0.0125849 - 0.00107903 - 0.000213912 \\ &= -0.843864 \end{aligned}$$

［表五］ 除式差分表

x_i	$x - x_i$	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdots]$	$f[\cdots]$
0.3	0.13	−1.203973				
			2.87682			
0.4	0.03	−0.916291		−3.22690		
			2.23144		3.95250	
0.5	−0.07	−0.693147		−2.04115		−4.60918
			1.82321		2.10883	
0.6	−0.17	−0.510826		−1.40850		
			1.54151			
0.7	−0.27	−0.356675				
0.8	−0.37	−0.223144				

由上表算至四階除式差分時，算得之 $f(0.43) = -0.843864$ ，且其最後加入項等於 -0.0002139 ，誤差仍大(實際誤差為 $\ln 0.43 - (-0.843864) = -0.000106$)，欲求較精確之值，可由前一除式差分表加入函數 $\ln 0.8$ 開始，計算除式差分至第五階，可算得

$$\begin{aligned} f(x) &= -0.843864 + (0.13)(0.03)(-0.07)(-0.17)(-0.27)(4.96596) \\ &= -0.843864 - 0.000062 \\ &= -0.843926 \end{aligned}$$

上值與正確值 $\ln 0.43 = -0.843970$ 之誤差為 -0.000044 ，與最後加入項 -0.000062 非常接近。

[表六] 除式差分表之擴大

x_i	$x - x_i$	$\ln x_i$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdots]$	$f[\cdots]$	$f[\cdots]$
.3	0.13	-1.203973					
	$DX(1)$						
			·				
.4	0.03	·		·			
	$DX(2)$						
			·		·		
.5	$-.07$	·		·		-4.60918	
	$DX(3)$					$DY(1)$	
			·		2.10883		4.96586
					$DY(2)$		$DY(1)$
.6	$-.17$	·		-1.40850		-2.12625	
	$DX(4)$			$DY(3)$		$DY(2)$	
			1.54151		1.25833		
			$DY(4)$		$DY(3)$		
.7	$-.27$	-0.356675		-1.03100			
	$DX(5)$	$DY(5)$		$DY(4)$			
			1.33531				
			$DY(5)$				
.8	$-.37$	-0.223144					
	$DX(6)$	$DY(6)$					

由上表計算可知欲求得高一階除式差分時，只要計算表中最後一列除式差分值，而且，計算最後一列數值時只需用到前一列除式差分值，故計算時可僅保留最後一列除式差分值，其保留在 $DIMENSION\ DY(I)$ 中之安排示於[表六]中各除式差分值之下方。如將原除式差分值 1.33531(存於

$DY(5))$及1.54151(存於$DY(4))$算得之除式差分值-1.03100仍存於$DY(4)$中(取代原存之1.54151)；餘類推。

做除式差分表時，$x_o, x_1, x_2, \cdots$之排列可不必照大小之次序，因此，可按$(x-x_i)$之絕對值由小至大之次序排列，使每次加入之變數x_i為最接近x者，以使內插函數值誤差較小。改按$|x-x_i|$之大小排列後，其除式差分表變成：

[表七]　按$|x-x_i|$排列之除式差分表

x_i	$x-x_i$	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdots]$	$f[\cdots]$	$f[\cdots]$
0.4	0.03	-0.916291					
			2.23144				
0.5	-0.07	-0.693147		-3.22690			
			2.55413		3.95250		
0.3	0.13	-1.203973		-2.43640		-4.60917	
			2.31049		2.56975		4.96585
0.6	-0.17	-0.510826		-1.92245		-2.62283	
			1.54151		1.78290		
0.7	-0.27	-0.356675		-1.03100			
			1.33531				
0.8	-0.37	-0.223144					

且

$$\begin{aligned} f(x) &= -0.916291 + (0.03)(2.23144) + (0.03)(-0.07)(-3.22690) \\ &\quad +(0.03)(-0.07)(0.13)(3.95250) \\ &\quad +(0.03)(-0.07)(0.13)(-0.17)(-4.60917) \\ &= -0.916291 + 0.0669432 + 0.00677649 - 0.00107903 - 0.000213913 \\ &= -0.843864 \end{aligned}$$

由[表六]與[表七]插值計算之結果，取至4項時結果相同。但若只取2項時，[表六]結果為$-1.203973 + 0.373987 = -0.829986$；[表七]結果為$-0.916291 + 0.0669432 = -0.849348$。與正確值$-0.843970$比較，可知[表七]照$|x-x_i|$之大小排列之結果較為準確。此乃因[表七]前2項相當於$\ln 0.4$與$\ln 0.5$內插計算；而[表六]前2項相當於用$\ln 0.3$與$\ln 0.4$外插計算；0.5較0.3更接近0.43，故結果較準確。至於取至3項以上時，兩表所用以插值之諸函數及諸x_i值完全相同，故結果亦無差異。

8.6 副程式 $DIFINT$

副程式係利用前節之牛頓插值公式計算函數內插值，至最後加入項小於要求誤差為止，若未滿足誤差要求，則採用所有函數值。其中

$X(N)$	=	N 個已知函數之 x_i 。
$Y(N)$	=	對應之函數 $f(x_i)$ 。
N	=	已知函數點數。
$DX(N)$	=	暫時運算位置。
$DY(N)$	=	暫時運算位置。
$L(N)$	=	暫時運算位置。
XX	=	欲插值之變數 x 。
YY	=	由程式算得之內插值 $f(x)$ 。
NN	=	由程式算得之內插值使用點數。內插計算所用之 x_i 為最接近 x 之 NN 個 x_i 。
YE	=	由程式算出之最後加入項之值。
EPS	=	精度要求。程式以 $\|YE/YY\| < EPS$ 時，即認為答案合乎精度要求而回呼叫程式。

試程式為找出 $x - x_i$ 之大小順序，須呼叫副程式 $SORTAL(A, L, N)$ 。副程式 $SORTAL$ 係根據 $A(1)$ 至 $A(N)$ 數值之大小，將最小之 $A(I)$ 之指標存於 $L(1)$ ，第二小之 $A(I)$ 之指標存於 $L(2)$ ，因此

$$A(L(1)) < A(L(2)) < A(L(3)) < \cdots < A(L(N))$$

［表八］ 牛頓插值法副程式

```
************************************************************************
      SUBROUTINE DIFINT(X,Y,N,DX,DY,L,XX,YY,NN,YE,EPS)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),Y(N),DX(N),DY(N),L(N)
C**   ================================================================    **
C**   Newton's interpolation formula                                      **
C**                                                                       **
C**   F(x) = f(x1) + (x-x1)f[x1,2] + (x-x1)(x-x2)f[x1,2,3] + ...          **
C**   or (for example, N=4) :                                             **
C**   F(x) = f(x4) + (x-x4)f[x4,3] + (x-x4)(x-x3)f[x4,3,2] + ...          **
C**   ================================================================    **
C**   +-------------------------------------------------------------+     **
C**   |           f(4)-f(3)              f[x4,3]-f[x3,2]            |     **
C**   | f[x4,3] = ---------; f[x4,3,2] = ---------------           |     **
C**   |            x4 - x3                    x4 - x2              |     **
C**   |                                                            |     **
```

```
C**   |                    f[x4,3,2]-f[x3,2,1]                    |    **
C**   |      p[x4,3,2,1] = -------------------                    |    **
C**   |                          x4 - x1                          |    **
C**   |                                                           |    **
C**   | x1 f(x1)                                                  |    **
C**   | x2 f(x2)    f[x2,1]                                       |    **
C**   | x3 f(x3)(3) f[x3,2](2) f[x3,2,1](1)                       |    **
C**   | xM f(x4)(4) f[x4,3](3) f[x4,3,2](2) f[x4,3,2,1](1)        |    **
C**   | x3    DY(3)   \  DY(I)   \    DY(1)   \                   |    **
C**   | xM    DY(M)   -> DY(I+1) ->   DY(I)   ->     DY(1)       |    **
C**   +-----------------------------------------------------------+    **
      DO 10 I=1,N
   10 DX(I)=DABS(XX-X(I))
      CALL SORTAL(DX,L,N)
C**   -------------------------------------------------------------    **
      YY=0.0
      PP=1.0
      DO 50 M=1,N
      NN=M
      I=L(M)
      DY(M)=Y(I)
      DX(M)=XX-X(I)
      DO 20 I=M-1,1,-1
      DY(I)=(DY(I+1)-DY(I))/(DX(I)-DX(M))
   20 CONTINUE
C     WRITE(*,'(3X,1P6E13.5)') DX(M),(DY(I),I=M,1,-1)
C**   +-----------------------------------------------------------+    **
C**   | YY = f(x1) + (x-x1)f[x1,2] + (x-x1)(x-x2)f[x1,2,3] + ... |    **
C**   | PP = (x-x1)(x-x2)(x-x3)...                                |    **
C**   +-----------------------------------------------------------+    **
      YE=PP*DY(1)
      YY=YY+YE
      PP=PP*DX(M)
      IF(DABS(YE).LT.EPS*DABS(YY)) RETURN
   50 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE SORTAL(A,L,N)
C**   =============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N),L(N)
C**   =============================================================    **
      DO 10 I=1,N
   10 L(I)=I
      M=N
   20 NE=M-1
      IF(NE) 60,60,30
   30 DO 50 I=1,NE
      J=L(I)
      K=L(I+1)
      IF(A(J)-A(K)) 50,50,40
   40 L(I)=K
      L(I+1)=J
      M=I
   50 CONTINUE
      IF(M-NE) 20,20,60
   60 RETURN
      END
*************************************************************************
```

[表九]為一主程式用以讀入N組數據$X(1)$至$X(N)$與$Y(1)$至$Y(N)$，再讀入XX與EPS後呼叫$DIFINT$計算YY並將結果印出。主程式後為輸入資料及計算結果。

[表九] 牛頓插值法主程式及輸入輸出資料

```
************************************************************************
C**   TEST PROGRAM FOR NEWTON'S INTERPOLATION                          **
C**   ================================================================ **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(20),Y(20),DX(20),DY(20),L(20)
C**   ================================================================ **
   10 READ(*,'(I5)') N
      READ(*,'(2F10.0)') (X(I),Y(I),I=1,N)
      WRITE(*,'(9X,''XX'',11X,''YY'',11X,''YE'',8X,''NN'')')
   30 READ(*,'(2F10.0)') XX,EPS
      IF(EPS.EQ.0.0) GO TO 10
      CALL DIFINT(X,Y,N,DX,DY,L,XX,YY,NN,YE,EPS)
      WRITE(*,'(3X,1P3E13.5,I5)') XX,YY,YE,NN
      GO TO 30
      END
************************************************************************
    7
 0.3      -1.203973
 0.4      -0.916291
 0.5      -0.693147
 0.6      -0.510826
 0.7      -0.356675
 0.8      -0.223144
 0.9      -0.105361    |        XX           YY           YE        NN
 0.43      0.1         |    4.30000E-01 -8.49348E-01  6.69432E-02    2
 0.43      0.01        |    4.30000E-01 -8.42571E-01  6.77649E-03    3
 0.43      0.001       |    4.30000E-01 -8.43864E-01 -2.13911E-04    5
 0.43      0.0001      |    4.30000E-01 -8.43926E-01 -6.22254E-05    6
 0.43      0.00001     |    4.30000E-01 -8.43949E-01 -2.27568E-05    7
```

8.7 副程式$NEWTON$

副程式係利用前節之牛頓插值公式計算函數之係數。其中

$X(N)$ = N個已知函數之x_i。
$Y(N)$ = 對應之函數$f(x_i)$。
N = 已知函數點數。
$C(N)$ = 牛頓插值函數之係數。
$P(N)$ = 暫時運算位置。
$F(N)$ = 暫時運算位置。

［表十］ 牛頓插值函數副程式(含試用程式及結果)

```
************************************************************************
      SUBROUTINE NEWTON(X,Y,N,C,P,F)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),Y(N),C(N),P(N),F(N)
C**   ================================================================    **
C**   Newton's interpolation formula                                      **
C**                                                                       **
C**   F(X(i)) = Y(i), for i=1,2,...,N                                     **
C**                                                                       **
C**   F(X) = C(1)*X**(N-1) + C(2)*X**(N-2) + ... + C(N-1)*X + C(N)       **
C**                                                                       **
C**   F(X) = Y(1)+(X-X(1))F[X1,X2]+(X-X(1))(X-X(2))F[X1,X2,X3]+...       **
C**   or:                                                                 **
C**   F(X) = Y(4)+(X-X(4))F[X4,X3]+(X-X(4))(X-X(3))F[X4,X3,X2]+...       **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   |           f(4)-f(3)              f[x4,3]-f[x3,2]             |    **
C**   | f[x4,3] = ---------; f[x4,3,2] = ----------------           |    **
C**   |            x4 - x3                   x4 - x2                 |    **
C**   |                                                              |    **
C**   |                     f[x4,3,2]-f[x3,2,1]                      |    **
C**   |      p[x4,3,2,1] = -------------------                       |    **
C**   |                           x4 - x1                            |    **
C**   |                                                              |    **
C**   | x1 f(x1)                                                     |    **
C**   | x2 f(x2)    f[x2,1]                                          |    **
C**   | x3 f(x3)(3) f[x3,2](2) f[x3,2,1](1)                          |    **
C**   | xM f(x4)(4) f[x4,3](3) f[x4,3,2](2) f[x4,3,2,1](1)           |    **
C**   | x3     F(3)   \   F(I)   \     F(1)   \                      |    **
C**   | xM     F(M)   ->  F(I+1) ->    F(I)   ->      F(1)          |    **
C**   +--------------------------------------------------------------+    **
      PM=1.0
      DO 50 M=1,N
      F(M)=Y(M)
      DO 20 I=M-1,1,-1
      F(I)=(F(I+1)-F(I))/(X(M)-X(I))
   20 CONTINUE
C     WRITE(*,'(3X,1P6E13.5)') X(M),(F(I),I=M,1,-1)
C**   +--------------------------------------------------------------+    **
C**   | F = C(1)*X**(N-1) + C(2)*X**(N-2) + ... + C(N-1)*X + C(N)    |    **
C**   |   = Y(1)+(X-X(1))F[X1,X2]+(X-X(1))(X-X(2))F[X1,X2,X3]+...    |    **
C**   |                                                              |    **
C**   | P = P(1)*X**(M-1) + P(2)*X**(M-2) + ... + P(M-1)*X + P(M)    |    **
C**   |   = (X-X(1))(X-X(2))...(X-X(M-1))                            |    **
C**   +--------------------------------------------------------------+    **
      P(M)=PM
      DO 40 I=M-1,1,-1
      C(I+1)=C(I)+P(I+1)*F(1)
      P(I+1)=P(I+1)-P(I)*X(M)
   40 CONTINUE
      C(1)=F(1)
      PM=-PM*X(M)
   50 CONTINUE
      RETURN
      END
```

```
************************************************************************
      FUNCTION POLFUN(C,N,X)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION C(N)
C**   ============================================================    **
C**   F(X) = C(1)*X**(N-1) + C(2)*X**(N-2) + ... + C(N-1)*X + C(N)    **
C**   ============================================================    **
      POLFUN=0.0
      DO 20 I=1,N
      POLFUN=POLFUN*X+C(I)
   20 CONTINUE
      RETURN
      END
************************************************************************
C**   TEST PROGRAM FOR NEWTON'S INTERPOLATION FORMULA                  **
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(20),Y(20),C(20),P(20),F(20)
C**   ============================================================    **
   10 READ(*,'(I5)') N
      READ(*,'(2F10.0)') (X(I),Y(I),I=1,N)
      CALL NEWTON(X,Y,N,C,P,F)
      WRITE(*,'(10X,''C'',12X,''P'',12X,''X'',12X,''Y'',12X,''F'')')
      WRITE(*,'(I3,1P5E13.5)') (I,C(I),P(I),X(I),Y(I),F(I),I=1,N)
      DO 50 I=1,N
      YY=POLFUN(C,N,X(I))
      WRITE(*,'(I3,1P2E13.5)') I,X(I),YY
   50 CONTINUE
      GO TO 10
      END
************************************************************************
         C            P            X            Y            F
 1  4.96583E+00  1.00000E+00  3.00000E-01 -1.20397E+00  4.96583E+00
 2 -1.70237E+01 -3.30000E+00  4.00000E-01 -9.16291E-01 -2.12625E+00
 3  2.44153E+01  4.45000E+00  5.00000E-01 -6.93147E-01  1.25833E+00
 4 -1.92897E+01 -3.13500E+00  6.00000E-01 -5.10826E-01 -1.03100E+00
 5  9.93725E+00  1.21540E+00  7.00000E-01 -3.56675E-01  1.33531E+00
 6 -2.98247E+00 -2.45520E-01  8.00000E-01 -2.23144E-01 -2.23144E-01
 1  3.00000E-01 -1.20397E+00
 2  4.00000E-01 -9.16291E-01
 3  5.00000E-01 -6.93147E-01
 4  6.00000E-01 -5.10826E-01
 5  7.00000E-01 -3.56675E-01
 6  8.00000E-01 -2.23144E-01
```

8.8 葉特肯插值公式

另一種插值公式為下列之葉特肯插值公式：

零階插值公式：

$$f(x|x_o) \;=\; f(x_o) \tag{8.22}$$

一階插值公式：

$$\begin{aligned} f(x|x_o,x_1) &= \frac{(x_1-x)f(x|x_o)-(x_o-x)f(x|x_1)}{x_1-x_o} \\ f(x|x_1,x_2) &= \frac{(x_2-x)f(x|x_1)-(x_1-x)f(x|x_2)}{x_2-x_1} \\ f(x|x_2,x_3) &= \frac{(x_3-x)f(x|x_2)-(x_2-x)f(x|x_3)}{x_3-x_2} \end{aligned} \tag{8.23}$$

二階插值公式：

$$\begin{aligned} f(x|x_o,x_1,x_2) &= \frac{(x_2-x)f(x|x_o,x_1)-(x_o-x)f(x|x_1,x_2)}{x_2-x_o} \\ f(x|x_1,x_2,x_3) &= \frac{(x_3-x)f(x|x_1,x_2)-(x_1-x)f(x|x_2,x_3)}{x_3-x_1} \end{aligned} \tag{8.24}$$

三階插值公式：

$$f(x|x_o,x_1,x_2,x_3) = \frac{(x_3-x)f(x|x_o,x_1,x_2)-(x_o-x)f(x|x_1,x_2,x_3)}{x_3-x_o} \tag{8.25}$$

其一般插值公式可寫成：

$$\begin{aligned} f(x|x_n,x_m,\overbrace{x_i,x_j,\cdots}^{\text{相同}}) &= \frac{1}{x_m-x_n}\begin{vmatrix} f(x|x_n,\overbrace{x_i,x_j,\cdots}) & x_n-x \\ f(x|x_m,\overbrace{x_i,x_j,\cdots}) & x_m-x \end{vmatrix} \\ &= \frac{(x_m-x)f(x|x_n,\overbrace{x_i,x_j,\cdots}^{\text{相同}})-(x_n-x)f(x|x_m,\overbrace{x_i,x_j,\cdots}^{\text{相同}})}{x_m-x_n} \end{aligned} \tag{8.26}$$

注意$f(x|x_i,x_j,x_k)$內諸x_i值不必按一定順序。亦請注意該插值公式與除式差分式間之相似與相異之處：

(1)葉特肯式與x有關，各值亦可仿照除式差分表排列成葉特肯插值表。表中各值均為所求函數之近似值，僅精度不同而已。例如當$x=x_i$時，則葉特肯插值表中凡是包含x_i之$f(x|\cdots,x_i,\cdots)$均等於$f(x_i)$。

(2)除式差分與x無關，該除式差分可用以計算不同x之函數值，但須另外計算與x有關之$\pi_i(x)=(x-x_o)(x-x_1)\cdots(x-x_i)$之值。

［表十一］ 葉特肯插值表

x	$f(x\|\cdot)$	$f(x\|\cdot,\cdot)$	$f(x\|\cdot,\cdot,\cdot)$	$f(x\|\cdot,\cdot,\cdot,\cdot)$
x_o	$f(x\|x_o)$			
		$f(x\|x_o,x_1)$		
x_1	$f(x\|x_1)$		$f(x\|x_o,x_1,x_2)$	
		$f(x\|x_1,x_2)$		$f(x\|x_o,x_1,x_2,x_3)$
x_2	$f(x\|x_2)$		$f(x\|x_1,x_2,x_3)$	
		$f(x\|x_2,x_3)$		
x_3	$f(x\|x_3)$			

葉特肯n階插值公式之展開式為x之n次多項式。而此n次多項式會通過與其有關之$(n+1)$點，即$f(x_i|x_i,x_j,\cdots)=f(x_i)$, $f(x_j|x_i,x_j,\cdots)=f(x_j)$等(此點可由定義式直接看出)，故此$n$次多項式必然與牛頓插值公式及拉格蘭治插值公式相同。

［表十二］ 按$|x_i-x|$排列之葉特肯插值表

x_i	x_i-x	$f(x\|\cdot)$	$f(x\|\cdot,\cdot)$	$f(x\|\cdots)$	$f(x\|\cdots)$	$f(x\|\cdots)$	$f(x\|\cdots)$
.4	−0.03	−0.916291					
			−.849348				
.5	0.07	−0.693147		−.842571			
			−.871936		−.843650		
.3	−0.13	−1.203973		−.849765		−.843864	
			−.903609		−.845789		−.843926
.6	0.17	−0.510826		−.861123		−.844694	
			−.772883		−.850485		
.7	0.27	−0.356675		−.820206			
			−.717209				
.8	0.37	−0.223144					

倫伯格積分表實際上為葉特肯插值表：由已知之$x_i=h_i^2$之面積(函數值)，外插計算$x=h^2=0$之面積(函數值)。

8.9 副程式 $AITINT$

副程式係利用前節之葉特肯插值公式計算函數內插值，至前後二次近似值之差小於要求誤差為止，若未滿足誤差要求，則採用所有函數值。該副程式之計算流程與副程式$DIFINT$十分相似且更簡單，其呼叫方式(與使用範例)則與後者相同。

［表十三］ 葉特肯插值法副程式

```
************************************************************************
      SUBROUTINE AITINT(X,Y,N,DX,AY,L,XX,YY,NN,YE,EPS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),Y(N),DX(N),AY(N),L(N)
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Aitken's iteration formula                               |    **
C**   |                                                          |    **
C**   |                        1     | y(x|x3,2) x2-x |          |    **
C**   |    y(x|x4,3,2) = --------- |                |          |    **
C**   |                    x4 - x2  | y(x|x4,3) x4-x |          |    **
C**   |                                                          |    **
C**   |                  (x2-x)y(x|x4,3) - (x4-x)y(x|x3,2)       |    **
C**   |                = ----------------------------------      |    **
C**   |                                 x2 - x4                  |    **
C**   | x1 y(x|x1)                                               |    **
C**   | x2 y(x|x2)    y(x|x2,1)                                  |    **
C**   | x3 y(x|x3)(3) y(x|x3,2)(2) y(x|x3,2,1)(1)                |    **
C**   | xM y(x|x4)(4) y(x|x4,3)(3) y(x|x4,3,2)(2) y(x|x4,3,2,1)(1)    **
C**   | x3      AY(3)   \    AY(I)   \      AY(1)   \             |    **
C**   | xM      AY(M)   ->   AY(I+1) ->     AY(I)   ->       AY(1)    **
C**   |                                                          |    **
C**   | 20 AY(I)=(AY(I+1)*DX(I)-AY(I)*DX(M))/(DX(I)-DX(M))       |    **
C**   | For extrapolation from |DX(1)| > |DX(2)| > ... to 0     |    **
C**   | 20 AY(I)=AY(I+1)+(AY(I+1)-AY(I))*DX(M)/(DX(I)-DX(M))     |    **
C**   | For interpolation from |DX(1)| < |DX(2)| < |DX(3)| < ... |    **
C**   | 20 AY(I)=AY(I)+(AY(I+1)-AY(I))*DX(I)/(DX(I)-DX(M))       |    **
C**   +----------------------------------------------------------+    **
      DO 10 I=1,N
   10 DX(I)=DABS(X(I)-XX)
      CALL SORTAL(DX,L,N)
C**   ------------------------------------------------------------    **
      YY=0.0
      DO 50 M=1,N
      NN=M
      I=L(M)
      AY(M)=Y(I)
      DX(M)=X(I)-XX
      DO 20 I=M-1,1,-1
   20 AY(I)=AY(I)+(AY(I+1)-AY(I))*DX(I)/(DX(I)-DX(M))
C     WRITE(*,'(3X,1P6E13.5)') DX(M),(AY(I),I=M,1,-1)
      YE=AY(1)-YY
      YY=AY(1)
      IF(DABS(YE).LT.EPS*DABS(YY)) RETURN
   50 CONTINUE
      RETURN
      END
************************************************************************
```

8.10 有理函數之紀雷插值公式

當函數值在某些點會趨近於無窮大時，以多項式函數近似常無法得到理想之結果。此種函數可考慮以下列之有理函數近似：

$$F(x)=\frac{c_p x^p+\cdots+c_1 x+c_o}{d_q x^q+\cdots+d_1 x+d_o} \tag{8.27}$$

通過[表一]中$(n+1)$點之有理函數之係數，如$p+q=n$，可用下列$(n+1)$元之聯立線性方程式求得(須設其中一個係數為1，如$c_p=1$或$d_q=1$)：

$$\begin{bmatrix} x_o^p & \cdots & x_o & 1 \\ x_1^p & \cdots & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^p & \cdots & x_n & 1 \end{bmatrix} \begin{Bmatrix} c_p \\ \vdots \\ c_1 \\ c_o \end{Bmatrix} = \begin{bmatrix} y_o & & & \\ & y_1 & & \\ & & \ddots & \\ & & & y_n \end{bmatrix} \begin{bmatrix} x_o^q & \cdots & x_o & 1 \\ x_1^q & \cdots & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^q & \cdots & x_n & 1 \end{bmatrix} \begin{Bmatrix} d_q \\ \vdots \\ d_1 \\ d_o \end{Bmatrix}$$

以下介紹之紀雷插值公式所求得之有理函數之$p=q$或$p=q+1$。

若已知點數$(n+1)=2m+1$時，$p=q=m$：

$$F(x)=\frac{c_1 x^m+c_3 x^{m-1}+\cdots+c_{2m+1}}{x^m+c_2 x^{m-1}+\cdots+c_{2m}} \tag{8.28}$$

若已知點數$(n+1)=2m$時，$p=m,\ q=m-1$：

$$F(x)=\frac{x^m+c_2 x^{m-1}+\cdots+c_{2m}}{c_1 x^{m-1}+\cdots+c_{2m-1}} \tag{8.29}$$

首先定義下列各階互換差分 (Reciprocal difference)：

零階互換差分：

$$\rho(x_o) \;=\; f(x_o) \tag{8.30}$$

一階互換差分：

$$\begin{aligned} \rho(x_o,x_1) &= \frac{x_o-x_1}{f(x_o)-f(x_1)} \\ \rho(x_1,x_2) &= \frac{x_1-x_2}{f(x_1)-f(x_2)} \\ \rho(x_2,x_3) &= \frac{x_2-x_3}{f(x_2)-f(x_3)} \end{aligned} \tag{8.31}$$

二階互換差分：

$$\begin{aligned} \rho(x_o,x_1,x_2) &= \frac{x_o-x_2}{\rho(x_o,x_1)-\rho(x_1,x_2)}+f(x_1) \\ \rho(x_1,x_2,x_3) &= \frac{x_1-x_3}{\rho(x_1,x_2)-\rho(x_2,x_3)}+f(x_2) \end{aligned} \tag{8.32}$$

三階互換差分：

$$\rho(x_o, x_1, x_2, x_3) = \frac{x_o - x_3}{\rho(x_o, x_1, x_2) - \rho(x_1, x_2, x_3)} + \rho(x_1, x_2) \qquad (8.33)$$

其一般式可寫成：

$$\rho(x_n, x_m, \underbrace{x_i, x_j, \cdots}_{\text{相同}}) = \frac{x_n - x_m}{\rho(x_n, \underbrace{x_i, x_j, \cdots}_{\text{相同}}) - \rho(x_m, \underbrace{x_i, x_j, \cdots}_{\text{相同}})} + \rho(\underbrace{x_i, x_j, \cdots}_{\text{相同}}) \qquad (8.34)$$

注意$\rho(x_i, x_j, x_k)$內諸x值不必按一定順序，即$\rho(x_i, x_j, x_k) = \rho(x_j, x_i, x_k) = \rho(x_j, x_k, x_i) = \cdots$。根據以上定義可做下列互換差分表：

［表十四］ 互換差分表

x	$\rho(\cdot)$	$\rho(\cdot,\cdot)$	$\rho(\cdot,\cdot,\cdot)$	$\rho(\cdot,\cdot,\cdot,\cdot)$	$\rho(\cdot,\cdot,\cdot,\cdot,\cdot)$
x_o	$\rho(x_o)$				
		$\rho(x_o, x_1)$			
x_1	$\rho(x_1)$		$\rho(x_o, x_1, x_2)$		
		$\rho(x_1, x_2)$		$\rho(x_o, x_1, x_2, x_3)$	
x_2	$\rho(x_2)$		$\rho(x_1, x_2, x_3)$		$\rho(x_o, x_1, x_2, x_3, x_4)$
		$\rho(x_2, x_3)$		$\rho(x_1, x_2, x_3, x_4)$	
x_3	$\rho(x_3)$		$\rho(x_2, x_3, x_4)$		
		$\rho(x_3, x_4)$			
x_4	$\rho(x_4)$				

根據互換差分定義可得下列各式：

$$f(x) = f(x_o) + \frac{x - x_o}{\rho(x, x_o)} \qquad (8.35)$$

$$\rho(x, x_o) = \rho(x_o, x_1) + \frac{x - x_1}{\rho(x, x_o, x_1) - f(x_o)} \qquad (8.36)$$

$$\rho(x, x_o, x_1) = \rho(x_o, x_1, x_2) + \frac{x - x_2}{\rho(x, x_o, x_1, x_2) - \rho(x_o, x_1)} \qquad (8.37)$$

$$\cdots$$

$$\rho(x, x_o, \cdots, x_{n-1}) = \rho(x_o, x_1, \cdots, x_n) + \frac{x - x_n}{\rho(x, x_o, \cdots, x_n) - \rho(x_o, \cdots, x_{n-1})} \qquad (8.38)$$

將式(8.36)以後各式代入式(8.35)即得下列紀雷插值公式。

$$F(x) = f(x_o) + \frac{x - x_o}{\rho(x_o, x_1) + \dfrac{x - x_1}{\rho(x_o, x_1, x_2) - f(x_o) + \dfrac{x - x_2}{\rho(x_o, x_1, x_2, x_3) - \rho(x_o, x_1) + \cdots}}} \quad (8.39)$$

注意上式展開即為x之有理函數，其分子多項式之次數，等於分母多項式之次數，或多一次。並可直接看出$F(x_i) = f(x_i)$，$i = 0, 1, \cdots, n$：因為當$F(x_i) = f(x_i)$時，上式即與$\rho(x_o, \cdots, x_i)$之定義式相當。可知該有理函數通過其有關之$(n+1)$點。

8.11 副程式$THIINT$與$THIELE$

副程式$THIINT$係利用前節之紀雷插值公式計算函數內插值，至前後二次近似值之差小於要求誤差為止，若未滿足誤差要求，則採用所有函數值。該副程式之計算流程與副程式$DIFINT$亦頗相似，其呼叫方式則與後者相同。副程式$THIELE$則以紀雷插值公式計算有理函數之係數$C(N)$。所含試用程式之用法與副程式$NEWTON$者類似。

［表十五］ 紀雷插值法副程式

```
************************************************************************
      SUBROUTINE THIINT(X,Y,N,DX,PY,L,XX,YY,NN,YE,EPS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),Y(N),DX(N),PY(N),L(N)
C**   ============================================================    **
C**   Thiele's interpolation formula                                  **
C**                                                                   **
C**   F(x) = f(x4)+(x-x4)   (x-x3)           (x-x2)                   **
C**                 -------- ---------------- --------------------    **
C**                 p(x4,3)+ p(x4,3,2)-f(x4)+ p(x4,3,2,1)-p(x4,3)+    **
C**   ============================================================    **
C**   +-----------------------------------------------------------+    **
C**   |             x4 - x3                    x4 - x2            |    **
C**   | p(x4,3) = ---------; p(x4,3,2) = ---------------- + f(x3) |    **
C**   |          f(4)-f(3)               p(x4,3)-p(x3,2)          |    **
C**   |                                                           |    **
C**   |                          x4 - x1                          |    **
C**   |      p(x4,3,2,1) = ------------------- + p(x3,2)          |    **
C**   |                    p(x4,3,2)-p(x3,2,1)                    |    **
C**   |                                                           |    **
C**   | x1 f(x1)                                                  |    **
C**   | x2 f(x2)    p(x2,1)                                       |    **
C**   | x3 f(x3)(3) p(x3,2)(2) p(x3,2,1)(1)                       |    **
C**   | xM f(x4)(4) p(x4,3)(3) p(x4,3,2)(2) p(x4,3,2,1)(1)        |    **
C**   | x3    PY(3)   \  PY(I)   \    PY(1)   \                   |    **
C**   | xM    PY(M)   -> PY(I+1) ->   PY(I)   ->     PY(1)       |    **
```

```
C**   +--------------------------------------------------------+    **
      DO 10 I=1,N
   10 DX(I)=DABS(XX-X(I))
      CALL SORTAL(DX,L,N)
C**   ---------------------------------------------------------    **
      YY=0.0
      DO 50 M=1,N
      YO=YY
      NN=M
      I=L(M)
      PY(M)=Y(I)
      DX(M)=XX-X(I)
      PYIP1=0.0
      DO 20 I=M-1,1,-1
      PYI=PY(I)
      DIFPY=PY(I+1)-PY(I)
      IF(DIFPY.EQ.0.0) DIFPY=EPS*DMAX1(DABS(PY(I)),EPS)
      PY(I)=(DX(I)-DX(M))/DIFPY+PYIP1
   20 PYIP1=PYI
C     WRITE(*,'(3X,1P6E13.5)') DX(M),(PY(I),I=M,1,-1)
C**   +--------------------------------------------------------+    **
C**   | YY = f(x4)+(x-x4)   (x-x3)           (x-x2)            |    **
C**   |             -------- ---------------- --------------------|    **
C**   |            p(x4,3)+ p(x4,3,2)-f(x4)+ p(x4,3,2,1)-p(x4,3)+|    **
C**   +--------------------------------------------------------+    **
      DD=0.0
      DO 40 I=1,M-2
   40 DD=DX(I+1)/(PY(I)-PY(I+2)+DD)
      IF(M.GT.1) DD=DX(M)/(PY(M-1)+DD)
      YY=PY(M)+DD
      YE=YY-YO
      IF(DABS(YE).LT.EPS*DABS(YY)) RETURN
   50 CONTINUE
      RETURN
      END
*************************************************************************
    6
 0.8       25.0
 0.4       20.0
 0.2       16.0
 0.1       13.0
 0.05      11.0
 0.025     10.25       |        XX           YY           YE        NN
 0.0       0.00001     |    0.00000E+00  8.55121E+00 -3.82919E-01    6
 0.075     0.00001     |    7.50000E-02  1.20580E+01 -1.72505E-02    6
 0.1       0.00001     |    1.00000E-01  1.30000E+01  0.00000E+00    2
 0.2       0.00001     |    2.00000E-01  1.60000E+01  0.00000E+00    2
```

［表十六］ 紀雷插值函數副程式(含試用程式及結果)

```
*************************************************************************
      SUBROUTINE THIELE(X,Y,N,C,P)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),Y(N),C(N),P(N)
      DATA EPS/1.0D-15/
C**   ========================================================    **
C**   Thiele's interpolation formula                              **
C**                                                               **
C**   F(X(i)) = Y(i), for i=1,2,...,N;  N=1,3,...,2*M+1;          **
```

```
C**                                                                    **
C**          C(1)*X**M + C(3)*X**(M-1) + ... + C(N-2)*X + C(2*M+1)     **
C**   F(X) = ---------------------------------------------------       **
C**               X**M + C(2)*X**(M-1) + ... + C(N-3)*X + C(2*M  )     **
C**                                                                    **
C**   F(X(i)) = Y(i), for i=1,2,...,N;  N=2,4,...,2*M;                 **
C**                                                                    **
C**               X**M + C(2)*X**(M-1) + ... + C(N-2)*X + C(2*M  )     **
C**   F(X) = ---------------------------------------------------       **
C**                     C(1)*X**(M-1) + ... + C(N-3)*X + C(2*M-1)      **
C**                                                                    **
C**   F(X) = Y(X4)+(X-X4)   (X-X3)           (X-X2)                   **
C**                --------- ---------------- ---------------------    **
C**                P(X4,3)+ P(X4,3,2)-Y(X4)+ P(X4,3,2,1)-P(X4,3)+     **
C**   ================================================================ **
C**   +--------------------------------------------------------------+ **
C**   |            x4 - x3                    x4 - x2                | **
C**   | p(x4,3) = ---------; p(x4,3,2) = --------------- + f(x3)     | **
C**   |           f(4)-f(3)              p(x4,3)-p(x3,2)             | **
C**   |                                                              | **
C**   |                           x4 - x1                            | **
C**   |      p(x4,3,2,1) = ------------------- + p(x3,2)             | **
C**   |                     p(x4,3,2)-p(x3,2,1)                      | **
C**   |                                                              | **
C**   | x1 f(x1)                                                     | **
C**   | x2 f(x2)    p(x2,1)                                          | **
C**   | x3 f(x3)(3) p(x3,2)(2) p(x3,2,1)(1)                          | **
C**   | xM f(x4)(4) p(x4,3)(3) p(x4,3,2)(2) p(x4,3,2,1)(1)           | **
C**   | x3     P(3)   \   P(I)   \     P(1)   \                      | **
C**   | xM     P(M)   ->  P(I+1) ->    P(I)   ->      P(1)           | **
C**   +--------------------------------------------------------------+ **
      DO 30 M=1,N
      P(M)=Y(M)
      PI1=0.0
      DO 20 I=M-1,1,-1
      PI=P(I)
      DP=P(I+1)-P(I)
      IF(DP.EQ.0.0) DP=EPS*DMAX1(DABS(P(I)),EPS)
      P(I)=(X(M)-X(I))/DP+PI1
   20 PI1=PI
C     WRITE(*,'(3X,1P6E13.5)') X(M),(P(I),I=M,1,-1)
   30 CONTINUE
C**   +--------------------------------------------------------------+ **
C**   | P(4),P(3); P(2)=p(x4,3,2)-f(x4);P(1)=p(x4,3,2,1)-p(x4,3)     | **
C**   +--------------------------------------------------------------+ **
      DO 40 I=1,N-2
   40 P(I)=P(I)-P(I+2)
C**   +--------------------------------------------------------------+ **
C**   | F(x)=f(x4)+(x-x4)   (x-x3)           (x-x2)                  | **
C**   |             --------- ---------------- ---------------------| **
C**   |             p(x4,3)+ p(x4,3,2)-f(x4)+ p(x4,3,2,1)-p(x4,3)+   | **
C**   +--------------------------------------------------------------+ **
      DO 60 M=1,N
      C(M)=0.0
      DO 50 I=M,3,-2
   50 C(I)=C(I)-C(I-2)*X(M)+C(I-1)*P(M)
      IF(MOD(M,2).EQ.0) C(2)=C(2)-X(M)+C(1)*P(M)
   60 IF(MOD(M,2).NE.0) C(1)=C(1)+P(M)
      RETURN
      END
```

```
************************************************************************
      FUNCTION RATFUN(C,N,X)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION C(N),D(2)
      DATA EPS/1.0D-15/
C**   ============================================================    **
C**   N = 1,3,...,2*M+1                                               **
C**                                                                   **
C**           C(1)*X**M + C(3)*X**(M-1) + ... + C(2*M+1)    D(2)      **
C**   F(X) = ------------------------------------------ = ------      **
C**                X**M + C(2)*X**(M-1) + ... + C(2*M  )    D(1)      **
C**                                                                   **
C**   N = 2,4,...,2*M                                                 **
C**                                                                   **
C**                X**M + C(2)*X**(M-1) + ... + C(2*M  )    D(1)      **
C**   F(X) = ------------------------------------------ = ------      **
C**                       C(1)*X**(M-1) + ... + C(2*M-1)    D(2)      **
C**   ============================================================    **
      D(1)=1.0
      D(2)=C(1)
      K=1
      DO 20 I=2,N
      D(K)=D(K)*X+C(I)
   20 K=3-K
      DK=D(K)
      IF(DK.EQ.0.0) DK=EPS
      RATFUN=D(3-K)/DK
      RETURN
      END
************************************************************************
C**   TEST PROGRAM FOR THIELE'S INTERPOLATION FORMULA                 **
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(20),Y(20),C(20),P(20)
C**   ============================================================    **
   10 READ(*,'(I5)') N
      READ(*,'(2F10.0)') (X(I),Y(I),I=1,N)
      CALL THIELE(X,Y,N,C,P)
      WRITE(*,'(10X,''C'',12X,''P'',12X,''X'',12X,''Y'')')
      WRITE(*,'(I3,1P4E13.5)') (I,C(I),P(I),X(I),Y(I),I=1,N)
      DO 50 I=1,N
      YY=RATFUN(C,N,X(I))
   50 WRITE(*,'(I3,1P2E13.5)') I,X(I),YY
      GO TO 10
      END
************************************************************************
         C            P            X            Y
 1  1.67281E-01  1.36246E-01  8.00000E-01  2.50000E+01
 2  4.46517E+00  3.21522E+01  4.00000E-01  2.00000E+01
 3  4.94207E-02 -2.29885E-03  2.00000E-01  1.60000E+01
 4  3.39413E-01 -8.25000E+00  1.00000E-01  1.30000E+01
 5 -1.43868E-03  3.33333E-02  5.00000E-02  1.10000E+01
 6 -1.23025E-02  1.02500E+01  2.50000E-02  1.02500E+01
 1  8.00000E-01  2.50000E+01
 2  4.00000E-01  2.00000E+01
 3  2.00000E-01  1.60000E+01
 4  1.00000E-01  1.30000E+01
 5  5.00000E-02  1.10000E+01
 6  2.50000E-02  1.02500E+01
```

8.12 樑曲線近似法

對於非屬高次多項式之函數，如果用高次多項式近似反而造成函數劇烈之變化，但是如果採用低次多項式，以三次式為例，如x_{i-1}, x_i點間之函數以通過$x_{i-2}, x_{i-1}, x_i, x_{i+1}$四點之三次函數近似；而$x_i, x_{i+1}$點間之函數以通過$x_{i-1}, x_i, x_{i+1}, x_{i+2}$四點之三次函數近似；則這二段函數在$x_i$點處之斜率將不連續而形成一轉折點。如果用牛頓插值公式所得之函數近似值係做為繪製曲線之用，則因這些轉折點而使曲線看起來不夠平滑。 因此圖形曲線之近似常不以牛頓或同類型插值公式為之。以下將以常用之三次函數近似曲線為例，說明一種能求得較平滑曲線之樑曲線近似法。

考慮下列x之三次函數式(8.40)，設其二端點為(x_i, y_i)與(x_j, y_j)，令$t = \frac{x-x_i}{h_i}$，$h_i = x_j - x_i$，則點i之$t = 0$，點j之$t = 1$。以後之推導將改用t之三次函數式(8.41)表示。由$y' = dy/dx = dy/(h_i dt)$，$y'' = d^2y/dx^2 = d^2y/(h_i^2 dt^2)$，可得式(8.42)。

$$y = C_o + C_1 x + C_2 x^2 + C_3 x^3 \tag{8.40}$$

$$= c_o + c_1 t + c_2 t^2 + c_3 t^3 \tag{8.41}$$

$$\begin{Bmatrix} y \\ y' h_i \\ y'' h_i^2 \end{Bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 \\ 0 & 1 & 2t & 3t^2 \\ 0 & 0 & 2 & 6t \end{bmatrix} \begin{Bmatrix} c_o \\ c_1 \\ c_2 \\ c_3 \end{Bmatrix} \tag{8.42}$$

其中四個係數可由二端點之(y_i, y_j, y_i'', y_j'')值，由下式求得。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 6 \end{bmatrix} \begin{Bmatrix} c_o \\ c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} y_i \\ y_j \\ y_i'' h_i^2 \\ y_j'' h_i^2 \end{Bmatrix} \tag{8.43}$$

由上式二邊乘係數矩陣之逆矩陣後代入式(8.42)可得式(8.44)。由該式代$t = 0$與$t = 1$可得二端點之y_i', y_j'與y_i'', y_j'' (及y_i, y_j)間之關係式(8.45)。但該式除y_i, y_j已知外，y_i'', y_j''並未指定，須改為y', y''之連續條件，即ij段與jk段在j點處之y', y''相等。因二段採用相同之y_j''，故僅需令ij段之$y_j' = \frac{y_j - y_i}{h_i} + \frac{h_i}{6} y_i'' + \frac{h_i}{3} y_j''$等於$jk$段之$y_j' = \frac{y_k - y_j}{h_j} - \frac{h_j}{3} y_j'' - \frac{h_j}{6} y_k''$即可，因此得式(8.46)。

$$\begin{Bmatrix} y \\ y'h_i \\ y''h_i^2 \end{Bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 \\ 0 & 1 & 2t & 3t^2 \\ 0 & 0 & 2 & 6t \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & -\frac{1}{3} & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{6} & \frac{1}{6} \end{bmatrix} \begin{Bmatrix} y_i \\ y_j \\ y_i''h_i^2 \\ y_j''h_i^2 \end{Bmatrix}$$

$$= \begin{bmatrix} 1-t & t & \frac{1}{6}(-2t+3t^2-t^3) & \frac{1}{6}(-t+t^3) \\ -1 & 1 & \frac{1}{6}(-2+6t-3t^2) & \frac{1}{6}(-1+3t^2) \\ 0 & 0 & 1-t & t \end{bmatrix} \begin{Bmatrix} y_i \\ y_j \\ y_i''h_i^2 \\ y_j''h_i^2 \end{Bmatrix} \quad (8.44)$$

$$\begin{Bmatrix} -y_i' \\ y_j' \end{Bmatrix} = \begin{Bmatrix} -\frac{y_j-y_i}{h_i} \\ \frac{y_j-y_i}{h_i} \end{Bmatrix} + \begin{bmatrix} \frac{h_i}{3} & \frac{h_i}{6} \\ \frac{h_i}{6} & \frac{h_i}{3} \end{bmatrix} \begin{Bmatrix} y_i'' \\ y_j'' \end{Bmatrix} \quad (8.45)$$

$$\frac{h_i}{6}\,y_i'' + \frac{h_i+h_j}{3}\,y_j'' + \frac{h_j}{6}\,y_k'' = \frac{y_k-y_j}{h_j} - \frac{y_j-y_i}{h_i} \quad (8.46)$$

對於如[表一]之$(n+1)$點僅可列出$(n-1)$個連續條件，另二個條件可採用多種方式：

(1)令$y_o''=0$及或$y_n''=0$。所得曲線稱為自然楔曲線。為最簡單而常用者。

(2)指定y_o''及或y_n''為定值。

(3)指定y_o'及或y_n'為定值，可由式(8.45)分別得對應之條件式。

(4)楔曲線常用於二維平面曲線或三維空間曲線之近似(詳後述)，如o與n為同一點之閉合曲線，則$y_o''=y_n''$，再用y_o'之連續條件(式(8.46))即可。

注意前三種方式所得聯立方程式之係數矩陣均為對稱矩陣且帶寬僅等於三，即所謂三對角矩陣，因此很容易分解與求解。但第四種方式者雖亦為對稱，但非三對角矩陣，不過如用變寬帶矩陣做分解與求解一樣很容易，運算時間約為三對角矩陣之二倍。

對於二維平面曲線或三維空間曲線之近似，以三維曲線為例，可以令$h_i^2=(x_j-x_i)^2+(y_j-y_i)^2+(z_j-z_i)^2$，並將$t$視為(與$x$無前述線性關係之)獨立自變數，對$x_i$做楔曲線近似，可得楔曲線$x(t)$；同樣再分別對$y_i,\ z_i$做楔曲線近似，可分別得楔曲線$y(t),\ z(t)$。則$(x(t),y(t),z(t))$即為以$t$為參數之三維空間楔曲線。注意此時三組未知數$x_i'',\ y_i'',\ z_i''$之係數矩陣均相同，只需做一次矩陣分解，再分別對由$x_i,\ y_i,\ z_i$等求得之三個常數向量

做前進代入及反向代入，即可求得三組x_i'', y_i'', z_i''之值。事實上可令所有之$h_i = 1$，或者任意給定h_i，而視其為第i段之權值，通常h_i愈大該段愈容易彎曲。

樑曲線之推導亦可以y_i'取代y_i''為未知數。其有關計算式為：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{Bmatrix} c_o \\ c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} y_i \\ y_j \\ y_i' h_i \\ y_j' h_i \end{Bmatrix} \tag{8.47}$$

$$\begin{Bmatrix} y \\ y' h_i \\ y'' h_i^2 \end{Bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 \\ 0 & 1 & 2t & 3t^2 \\ 0 & 0 & 2 & 6t \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{Bmatrix} y_i \\ y_j \\ y_i' h_i \\ y_j' h_i \end{Bmatrix}$$

$$= \begin{bmatrix} 1 - 3t^2 + 2t^3 & 3t^2 - 2t^3 & t - 2t^2 + t^3 & -t^2 + t^3 \\ -6t + 6t^2 & 6t - 6t^2 & 1 - 4t + 3t^2 & -2t + 3t^2 \\ -6 + 12t & 6 - 12t & -4 + 6t & -2 + 6t \end{bmatrix} \begin{Bmatrix} y_i \\ y_j \\ y_i' h_i \\ y_j' h_i \end{Bmatrix} \tag{8.48}$$

$$\begin{Bmatrix} -y_i'' \\ y_j'' \end{Bmatrix} = \begin{Bmatrix} -\frac{6(y_j - y_i)}{h_i^2} \\ -\frac{6(y_j - y_i)}{h_i^2} \end{Bmatrix} + \begin{bmatrix} \frac{4}{h_i} & \frac{2}{h_i} \\ \frac{2}{h_i} & \frac{4}{h_i} \end{bmatrix} \begin{Bmatrix} y_i' \\ y_j' \end{Bmatrix} \tag{8.49}$$

$$\frac{2}{h_i} y_i' + (\frac{4}{h_i} + \frac{4}{h_j}) y_j' + \frac{2}{h_j} y_k' = \frac{6(y_k - y_j)}{h_j^2} + \frac{6(y_j - y_i)}{h_i^2} \tag{8.50}$$

附帶一提：式(8.46)相當於結構學中連續梁之三彎矩方程式 (Three moment equation)，梁斷面之彎矩與y''成正比，式(8.45)之方陣相當於柔度 (Flexibility)矩陣。而式(8.50)相當於傾角撓度方程式 (Slope–deflection equation)，y'即為梁之傾角，式(8.49)之方陣相當於勁度 (Stiffness)矩陣。而式(8.49)可由式(8.45)推導出。較高次(須為奇次)之樑曲線亦可按類似方法導出，如五次者可以y'', y''''為未知數，而以y', y'''之連續條件列出方程式。不過高次曲線之彎曲變化常較劇烈，反而不理想，故幾乎不被採用。

上述以$y(x)$或以參數t做$(x(t), y(t))$之近似曲線在一些大的轉折點會有較大的曲率變化。若將式(8.45)及式(8.46)改為下列二式，亦即將y'改

為θ，則所得之近似曲線之曲率變化較緩和。

$$\begin{Bmatrix} -\theta_i \\ \theta_j \end{Bmatrix} = \begin{Bmatrix} -\tan^{-1}\frac{y_j-y_i}{x_j-x_i} \\ \tan^{-1}\frac{y_j-y_i}{x_j-x_i} \end{Bmatrix} + \begin{bmatrix} \frac{h_i}{3} & \frac{h_i}{6} \\ \frac{h_i}{6} & \frac{h_i}{3} \end{bmatrix} \begin{Bmatrix} y_i'' \\ y_j'' \end{Bmatrix} \tag{8.51}$$

$$\frac{h_i}{6}y_i'' + \frac{h_i+h_j}{3}y_j'' + \frac{h_j}{6}y_k'' = \tan^{-1}\frac{y_k-y_j}{x_k-x_j} - \tan^{-1}\frac{y_j-y_i}{x_j-x_i} \tag{8.52}$$

令由二端之y''所得之部分θ為$\Delta\theta_i = -(2y_i''+y_j'')h_i/6$與$\Delta\theta_j = (y_i''+2y_j'')h_i/6$，令$\alpha_{ij} = \Delta\theta_i,\ \beta_{ji} = -\Delta\theta_j$，理論上如$\alpha_{ij} = \beta_{ji} = \phi$則所得曲線為一圓弧，欲得此圓弧曲線可採用如下方式：以n等分點為例，將兩端ϕ均分為n等分，則兩組等分線以相同之時鐘方向依序相交之點即落在圓弧上之等分點。如$\alpha_{ij} \neq \beta_{ji}$則建議如下方式：將曲線$ij$分為$ik$與$kj$二段，令$k$點落在兩端之$\phi = \frac{1}{2}(\alpha_{ij}+\beta_{ji})$之圓弧中點。因該圓弧與曲線在$i$點或$j$點之夾角均為$\omega = \frac{1}{2}(\alpha_{ij}-\beta_{ji})$，而$ik$與$ij$或$jk$與$ji$之夾角為$\frac{1}{2}\phi$，故得$\alpha_{ik} = \frac{1}{2}\phi+\omega$，$\beta_{jk} = \frac{1}{2}\phi-\omega$。並令$\beta_{ki} = \frac{1}{2}\phi+\frac{1}{2}\omega$，$\alpha_{kj} = \frac{1}{2}\phi-\frac{1}{2}\omega$，即完成二段之分割。用同樣之作法可將二段分為四段，四段分為八段等，各分段點即為曲線上之點。注意圓弧曲線亦可用此種方式求得。

式(8.51)中之h_i如改為ij線段彎曲後之曲線長s_i，則若已知點均在一圓周上且至少有三點，即可得閉合曲線為一正圓。此點可由下列事實証知：若y''在各點均相等，則圓弧曲率半徑相等且由式(8.51)知$\Delta\theta_i$及$\Delta\theta_j$與s_i成正比（式中h_i改為s_i）。$\Delta\theta_i$即為圓弧在i點之切線與ij直線之夾角，此夾角與弧長s_i成正比即為圓弧之特性。

對於三維曲線，本章亦建議將曲線ij分為ik與kj二段。設ij曲線兩端點位置向量為$\vec{r}_i$與$\vec{r}_j$，切線單位向量為$\vec{t}_i$與$\vec{t}_j$。以下為k點位置向量$\vec{r}_k$及切線單位向量$\vec{t}_k$之求法：由$h\vec{t} = \vec{r}_j - \vec{r}_i$可得$ij$直線之單位向量$\vec{t}$與長度$h$，令由直線$ij$旋轉至切線$\vec{t}_i$之轉角為$\alpha$，轉軸之單位向量為$\vec{a}$，由切線$\vec{t}_j$旋轉至直線$ij$之延長線之轉角為$\beta$，轉軸之單位向量為$\vec{b}$。令由直線$ij$旋轉至$ik$或由$jk$旋轉至$ji$之轉角為$\frac{1}{2}\phi$，轉軸之單位向量為$\vec{c}$，其中$\phi\vec{c} = \frac{1}{2}(\alpha\vec{a}+\beta\vec{b})$。由此可得$k$點之位置向量$\vec{r}_k = \frac{1}{2}(\vec{r}_i+\vec{r}_j) + \frac{1}{2}h\tan(\frac{1}{2}\phi)\vec{d}$，其中$\vec{d} = \vec{c}\times\vec{t}$。注意$\vec{c}$，$\vec{t}$，$\vec{d}$為一組互相垂直的單位向量。令$\vec{t}_k$係由$k$點之$\vec{t}$先繞$\vec{c}$軸轉一弧角$-\frac{1}{2}\omega$，再繞$\vec{d}$轉一弧角$\frac{1}{2}\rho$而得。其中之$\omega$與$\rho$分別為向量$\vec{p} = \frac{1}{2}(\alpha\vec{a}-\beta\vec{b})$在平行於$\vec{c}$與垂直於$\vec{c}$之分量之大小。注意前述二維曲線之求法為上述三維曲線之求法之特例。

至於三維空間曲線在已知點處之切線可按如下方式求得：將各點分別投影至xy、yz與zx三平面，然後照式(8.51)與式(8.52)求該三平面之二維曲線之切線方向，設其單位向量為$(c_{xy}, c_{yx}, 0)$，$(0, c_{yz}, c_{zy})$，$(c_{xz}, 0, c_{zx})$。令三維曲線切線之單位向量為(t_x, t_y, t_z)，理論上$\frac{t_y}{t_x} = \frac{c_{yx}}{c_{xy}}$，$\frac{t_z}{t_y} = \frac{c_{zy}}{c_{yz}}$，$\frac{t_x}{t_z} = \frac{c_{xz}}{c_{zx}}$，但除非$p^3 = |\frac{c_{yx}}{c_{xy}}\frac{c_{zy}}{c_{yz}}\frac{c_{xz}}{c_{zx}}|$等於$1$，否則該三條件不成立。若令$\frac{t_y}{t_x} = \frac{c_{yx}}{pc_{xy}}$，$\frac{t_z}{t_y} = \frac{c_{zy}}{pc_{yz}}$，$\frac{t_x}{t_z} = \frac{c_{xz}}{pc_{zx}}$。則由$t_x^2 + t_y^2 + t_z^2 = 1$之關係可得$t_x = 1/\sqrt{1 + (\frac{c_{yx}}{pc_{xy}})^2(1 + (\frac{c_{zy}}{pc_{yz}})^2)}$，$t_y = 1/\sqrt{1 + (\frac{c_{zy}}{pc_{yz}})^2(1 + (\frac{c_{xz}}{pc_{zx}})^2)}$，$t_z = 1/\sqrt{1 + (\frac{c_{xz}}{pc_{zx}})^2(1 + (\frac{c_{yx}}{pc_{xy}})^2)}$。$t_x, t_y, t_z$之正負可分別由$c_{xz}, c_{yx}, c_{zy}$決定。

8.13 副程式$SPLINE$，$SPLINF$與$SPLING$

副程式$SPLINE$主要用以計算h_i及係數矩陣後呼叫副程式$VBDECP$做矩陣分解。副程式$SPLINF$則用以計算條件式之常數向量再呼叫副程式$VBSOLX$計算y''值。副程式$SPLING$則可由各段二端之y''計算曲線之座標值。該套程式可做一至三維之曲線近似。二維及三維可做閉合曲線，一維可做週期函數之平滑曲線(類似閉合曲線)。非閉合曲線之二端點可指定y'或y''之值。其詳細用法請直接參考副程式中之說明。

［表十七］ 樑曲線近似法副程式之一(含試用程式)

```
************************************************************************
      SUBROUTINE SPLINE(H,X,Y,Z,N,M,KB,KH,A,L)
C**   ============================================================   **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(N),X(N),Y(N),Z(N),A(1),L(N)
C**   ============================================================   **
CIO   H(I) = Parameter/Length between nodes I & J=I+1 or I=N & J=1   **
C*I   X(I),Y(I),Z(I) = Coordinates of the I-th node                  **
C*I   N    = Number of nodes                                        **
C*I   KB   = Index for boundary conditions at both ends             **
C**        = 0 : left_y'' given, right_y'' given                    **
C**        = 1 : left_y'' given, right_y'  given                    **
C**        = 2 : left_y'  given, right_y'' given                    **
C**        = 3 : left_y'  given, right_y'  given                    **
C**        = 4 : left_y' = right_y' for:                            **
C**              2-D and 3-D closed curve                           **
C**              1-D periodic curve : use KH = 1 or -1              **
C*I   KH   = Index for H(I) values:                                 **
C**        = 0 : set H(I) = Length between nodes I & J              **
C**        = 1 : set H(I) = 1                                       **
C**        = otherwise : H(I) are given                             **
C**          In any case, this subroutine set H(N)=0 for KB<4       **
C*I   M    = Space dimension of the curve                           **
C**        = 1 : For 1-D curve y(x)                                 **
C**        = 2 : For 2-D curve (x(t),y(t))                          **
```

```
C**        = 3 : For 3-D curve (x(t),y(t),z(t))                          **
C*O   A(*) = A(2*N-2) for KB<4,  A(3*N-4) for KB=4.                      **
C*O   L(N) = Location index for variable banded matrix A                 **
C**          The coefficient matrix of the spline equation is            **
C**          (Data in [] are for KB=4 only)                              **
C**       / (Hn+H1)/3  H1/6                                 [Hn/6] \     **
C**       |       (H1+H2)/3  H2/6                           [  0 ] |     **
C**       |             (H2+H3)/3  H3/6                     [  0 ] |     **
C**   A = |                        ....                     [  0 ] |     **
C**       |                            ....    Hn-2/6       [  0 ] |     **
C**       |                                (Hn-2+Hn-1)/3  Hn-1/6 |     **
C**       \                                           (Hn-1+Hn)/3 /     **
C**   ================================================================    **
      IF(KH.EQ.0) THEN
        DO 10 I=1,N
        J=I+1
        IF(I.EQ.N) J=1
        IF(M.EQ.1) H(I)=(X(J)-X(I))**2
        IF(M.EQ.2) H(I)=(X(J)-X(I))**2+(Y(J)-Y(I))**2
        IF(M.EQ.3) H(I)=(X(J)-X(I))**2+(Y(J)-Y(I))**2+(Z(J)-Z(I))**2
        H(I)=DSQRT(H(I))
   10   CONTINUE
      ELSE IF(KH.EQ.1) THEN
        DO 20 I=1,N
        H(I)=1.0
   20   CONTINUE
      ENDIF
      IF(KB.LT.4) H(N)=0.0
C**   ----------------------------------------------------------------    **
      L(1)=0
      A(1)=(H(N)+H(1))/3.0
      DO 40 I=1,N-1
      A(2*I  )=H(I)/6.0
      A(2*I+1)=(H(I)+H(I+1))/3.0
      L(I+1)=L(I)+1
   40 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | For y'' given at  left end (KB=00xB)                   |    **
C**   +--------------------------------------------------------+    **
      IF(KB.EQ.0.OR.KB.EQ.1) A(1)=1.0D30
C**   +--------------------------------------------------------+    **
C**   | For y'' given at right end (KB=0x0B)                   |    **
C**   +--------------------------------------------------------+    **
      IF(KB.EQ.0.OR.KB.EQ.2) A(2*N-1)=1.0D30
C**   +--------------------------------------------------------+    **
C**   | For closed curve           (KB=1xxB)                   |    **
C**   +--------------------------------------------------------+    **
      IF(KB.GE.4) THEN
        L(N)=L(N)+N-2
        A(3*N-3)=A(2*N-1)
        A(3*N-4)=A(2*N-2)
        A(2*N-2)=H(N)/6.0
        DO 55 I=2*N-1,3*N-5
   55   A(I)=0.0
      ENDIF
      CALL VBDECP(A,L,N)
      RETURN
      END
*************************************************************************
      SUBROUTINE SPLINF(H,Y,YPP,N,KB,YPP1,YPPN,A,L)
```

```
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(N),Y(N),YPP(N),A(1),L(N)
C**   ============================================================    **
C*I   H,Y,N,KB,A,L = Should be the same as SPLINE                     **
C*O   YPP(I) = y'' at the I-th node                                   **
C*I   YPP1   = y''  y''  y'   y'   -  at the  left end                **
C*I   YPPN   = y''  y'   y''  y'   -  at the right end                **
C**   for KB = 0    1    2    3    4                                  **
C**                                                                  **
C**                / (y2-y1)/H1 - (y1-yn)/Hn     \                   **
C**                | (y3-y2)/H2 - (y2-y1)/H1     |                   **
C**           -1   | (y4-y3)/H3 - (y3-y2)/H2     |                   **
C**   {YPP} = [A] * |          .....              |                   **
C**                |          .....              |                   **
C**                |          ....               |                   **
C**                \ (y1-yn)/Hn - (yn-yn-1)/Hn-1 /                   **
C**   ============================================================    **
      DO 60 I=2,N-1
      YPP(I)=(Y(I+1)-Y(I))/H(I)-(Y(I)-Y(I-1))/H(I-1)
   60 CONTINUE
      YPP(1)=(Y(2)-Y(1))/H(1)
      YPP(N)=-(Y(N)-Y(N-1))/H(N-1)
C**   +----------------------------------------------------------+    **
C**   | For y''= YPP1 at  left end (KB=00xB)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.0.OR.KB.EQ.1) YPP(1)=YPP1*A(1)
C**   +----------------------------------------------------------+    **
C**   | For y''= YPPN at right end (KB=0x0B)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.0.OR.KB.EQ.2) YPP(N)=YPPN*A(2*N-1)
C**   +----------------------------------------------------------+    **
C**   | For y' = YPP1 at  left end (KB=01xB)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.2.OR.KB.EQ.3) YPP(1)=YPP(1)-YPP1
C**   +----------------------------------------------------------+    **
C**   | For y' = YPPN at right end (KB=0x1B)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.1.OR.KB.EQ.3) YPP(N)=YPP(N)+YPPN
C**   +----------------------------------------------------------+    **
C**   | For closed curve           (KB=1xxB)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.GE.4) YPP(1)=YPP(1)-(Y(1)-Y(N))/H(N)
      IF(KB.GE.4) YPP(N)=YPP(N)+(Y(1)-Y(N))/H(N)
      CALL VBSOLX(A,YPP,L,N)
      RETURN
      END
*********************************************************************
      SUBROUTINE SPLING(H,Y,YPP,N,I,T,YT)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(N),Y(N),YPP(N)
C**   ============================================================    **
C**   Find YT for given H,Y,YPP,N,I,T                                 **
C**   ============================================================    **
      J=I+1
      IF(I.EQ.N) J=1
      S=1-T
      YT=S*Y(I)+T*Y(J)+((S*S*S-S)*YPP(I)+(T*T*T-T)*YPP(J))*H(I)*H(I)/6
      RETURN
```

```
      END
*************************************************************************
C**   TEST PROGRAM FOR SPLINE CURVE FITTING                             **
C**   ================================================================  **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(20),X(20),Y(20),Z(20),XPP(20),YPP(20),ZPP(20)
      DIMENSION A(60),L(20)
C**   ================================================================  **
C**   N  = Number of given nodes                                        **
C**   M  = 1:3 = Dimensions of the curve                                 **
C**   KB = 0:4 see SUBROUTINE SPLINE                                    **
C**   KH = 1 : set H(I) = 1 by SUBROUTINE SPLINE                        **
C**      = 0 : set H(I) = Length between nodes I & J by SUB.SPLINE      **
C**   NS = Number of line segments for a curve between two nodes        **
C**   XPP1,XPPN,...  = See SUBROUTINE SPLINF                            **
C**   X(N),Y(N),Z(N) = Coordinates of nodes                             **
C**   ================================================================  **
   10 READ(*,'(5I5,6F5.0)') N,M,KB,KH,NS,XPP1,XPPN,YPP1,YPPN,ZPP1,ZPPN
      IF(M.LT.1.OR.M.GT.3) M=1
      IF(NS.EQ.0) NS=16
      KH=MAX0(KH,0)
      READ(*,'(3F10.0)') (X(I),Y(I),Z(I),I=1,N)
      WRITE(*,'(5I5,6F5.2)') N,M,KB,KH,NS,XPP1,XPPN,YPP1,YPPN,ZPP1,ZPPN
C**   ----------------------------------------------------------------  **
      CALL SPLINE(H,X,Y,Z,N,M,KB,KH,A,L)
C**   ----------------------------------------------------------------  **
      IF(M.GE.1) CALL SPLINF(H,Y,YPP,N,KB,YPP1,YPPN,A,L)
      IF(M.GE.2) CALL SPLINF(H,X,XPP,N,KB,XPP1,XPPN,A,L)
      IF(M.GE.3) CALL SPLINF(H,Z,ZPP,N,KB,ZPP1,ZPPN,A,L)
      IF(M.GE.1) WRITE(*,'('' YP'',1P6E13.5)') (YPP(I),I=1,N)
      IF(M.GE.2) WRITE(*,'('' XP'',1P6E13.5)') (XPP(I),I=1,N)
      IF(M.GE.3) WRITE(*,'('' ZP'',1P6E13.5)') (ZPP(I),I=1,N)
C**   ----------------------------------------------------------------  **
      WRITE(*,'(4X,''I    K  Sum.H(t)     X(t)      Y(t)      Z(t)'')')
      HO=0.0
      DO 50 I=1,N
      IF(H(I).EQ.0.0) GO TO 50
      DO 40 K=0,NS
      IF(K.EQ.0.AND.I.GT.1) GO TO 40
      T=FLOAT(K)/NS
      HT=HO+H(I)*T
      IF(M.GE.1) CALL SPLING(H,Y,YPP,N,I,T,YT)
      IF(M.GE.2) CALL SPLING(H,X,XPP,N,I,T,XT)
      IF(M.GE.3) CALL SPLING(H,Z,ZPP,N,I,T,ZT)
      IF(M.EQ.1) WRITE(*,'(2I5,4F10.5)') I,K,HT,HT,YT
      IF(M.EQ.2) WRITE(*,'(2I5,4F10.5)') I,K,HT,XT,YT
      IF(M.EQ.3) WRITE(*,'(2I5,4F10.5)') I,K,HT,XT,YT,ZT
   40 CONTINUE
      HO=HO+H(I)
   50 CONTINUE
      GO TO 10
      END
*************************************************************************
    6    2    4    0   32  0.0  0.0  0.0  0.0  0.0  0.0
      0.00      0.00
      1.00      2.00
      2.00      4.00
      3.00      3.00
      4.00      1.00
      5.00      4.00
```

8.14 副程式 $SPLINO$，$SPLIN2$ 與 $SPLIN3$

本節所呼叫之 $SPLINE$ 與上節者相同。副程式 $SPLINO$ 用以取代上節之 $SPLINF$。副程式 $SPLIN2$ 與 $SPLIN3$ 用以取代上節之 $SPLING$ 以計算曲線之座標值。該套程式可做二或三維之非閉合或閉合曲線近似。其詳細用法請直接參考副程式中之說明。圖二列示以 $SPLINE$ 與 $SPLINO$ 二種曲線近似法所得之曲線，所用之點座標為：$(0,0),(1,2),(2,4),(3,3),(4,1),(5,4)$，$SPLINE$ 用 $KH=0$，$SPLINO$ 用 $KL=-2$。

［表十八］ 樑曲線近似法副程式之二(含試用程式)

```
************************************************************************
      SUBROUTINE SPLINO(X,Y,DYX,N,KB,DYX1,DYXN,A,L)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),Y(N),DYX(N),A(1),L(N)
C**   ============================================================    **
C*I   X,Y,N,KB,A,L = Should be the same as SPLINE                     **
C*O   DYX(I) = k at the I-th point (k=curvature, a=rotation angle)    **
C*I   DYX1   = k    k    a    a    -  at the  left end               **
C*I   DYXN   = k    a    k    a    -  at the right end               **
C**   for KB = 0    1    2    3    4                                 **
C**                                                                  **
C**                / atan(y2-y1)/(x2-x1) - atan(y1-yn)/(x1-xn)  \    **
C**                | atan(y3-y2)/(x3-x2) - atan(y2-y1)/(x2-x1)  |    **
C**             -1 | atan(y4-y3)/(x4-x3) - atan(y3-y2)/(x3-x2)  |    **
C**   {DYX} = [A] * |                  .....                     |    **
C**                |                  .....                     |    **
C**                |                  .....                     |    **
C**                \ atan(y1-yn)/(x1-xn)-atan(yn-yn-1)/(xn-xn-1)/    **
C**   ============================================================    **
      DO 60 I=2,N-1
      DYX(I)=DATAN2(Y(I+1)-Y(I),X(I+1)-X(I))
     *      -DATAN2(Y(I)-Y(I-1),X(I)-X(I-1))
      IF(DABS(DYX(I)).GT.3.14159) DYX(I)=DYX(I)-DSIGN(6.28318D0,DYX(I))
   60 CONTINUE
      DYX(1)=DATAN2(Y(2)-Y(1),X(2)-X(1))
      DYX(N)=-DATAN2(Y(N)-Y(N-1),X(N)-X(N-1))
C**   +----------------------------------------------------------+    **
C**   | For k  = DYX1 at  left end (KB=00xB)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.0.OR.KB.EQ.1) DYX(1)=DYX1*A(1)
C**   +----------------------------------------------------------+    **
C**   | For k  = DYXN at right end (KB=0x0B)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.0.OR.KB.EQ.2) DYX(N)=DYXN*A(2*N-1)
C**   +----------------------------------------------------------+    **
C**   | For a  = DYX1 at  left end (KB=01xB)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.2.OR.KB.EQ.3) DYX(1)=DYX(1)-DYX1
C**   +----------------------------------------------------------+    **
C**   | For a  = DYXN at right end (KB=0x1B)                     |    **
C**   +----------------------------------------------------------+    **
      IF(KB.EQ.1.OR.KB.EQ.3) DYX(N)=DYX(N)+DYXN
```

```
C**   +--------------------------------------------------------+    **
C**   | For closed curve          (KB=1xxB)                    |    **
C**   +--------------------------------------------------------+    **
      IF(KB.GE.4) THEN
      DYX(1)=DYX(1)-DATAN2(Y(1)-Y(N),X(1)-X(N))
      DYX(N)=DYX(N)+DATAN2(Y(1)-Y(N),X(1)-X(N))
      IF(DABS(DYX(1)).GT.3.14159) DYX(1)=DYX(1)-DSIGN(6.28318D0,DYX(1))
      IF(DABS(DYX(N)).GT.3.14159) DYX(N)=DYX(N)-DSIGN(6.28318D0,DYX(N))
      ENDIF
      CALL VBSOLX(A,DYX,L,N)
      RETURN
      END
*************************************************************************
      SUBROUTINE SPLIN2(H,X,Y,DYX,N,I,NS,XT,YT)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(N),X(N),Y(N),DYX(N)
      DIMENSION XT(1),YT(1),SA(129),SB(129)
C**   ============================================================    **
C**   Find XT(NS+1),YT(NS+1) for given H,X,Y,DYX,N,I,NS=2**K          **
C**   ============================================================    **
      J=I+1
      IF(I.EQ.N) J=1
      XT(   1)=X(I)
      YT(   1)=Y(I)
      XT(NS+1)=X(J)
      YT(NS+1)=Y(J)
      SA(   1)=-(2*DYX(I)+DYX(J))*H(I)/6
      SB(NS+1)=-(DYX(I)+2*DYX(J))*H(I)/6
      IF(NS.GT.128) WRITE(*,'(''NS > 128'')')
      IX=NS
   20 DO 50 IA=1,NS,IX
      IB=IA+IX
      IC=(IA+IB)/2
      CALL SPLINP(XT(IA),YT(IA),XT(IB),YT(IB),XT(IC),YT(IC)
     *           ,SA(IA),SB(IB),SA(IC),SB(IC))
   50 CONTINUE
      IX=IX/2
      IF(IX.GT.1) GO TO 20
      RETURN
      END
*************************************************************************
      SUBROUTINE SPLIN3(H,X,Y,Z,DZY,DXZ,DYX,N,I,NS,XT,YT,ZT)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(N),X(N),Y(N),Z(N),DZY(N),DXZ(N),DYX(N)
      DIMENSION XT(1),YT(1),ZT(1),CX(129),CY(129),CZ(129)
C**   ============================================================    **
C**   Find XT,YT,ZT(NS+1) for given H,X,Y,Z,DZY,DXZ,DYX,N,I,NS=2**K   **
C**   ============================================================    **
      J=I+1
      IF(I.EQ.N) J=1
      XT(   1)=X(I)
      YT(   1)=Y(I)
      ZT(   1)=Z(I)
      XT(NS+1)=X(J)
      YT(NS+1)=Y(J)
      ZT(NS+1)=Z(J)
      CX(   1)=-(2*DZY(I)+DZY(J))*H(I)/6
      CY(   1)=-(2*DXZ(I)+DXZ(J))*H(I)/6
```

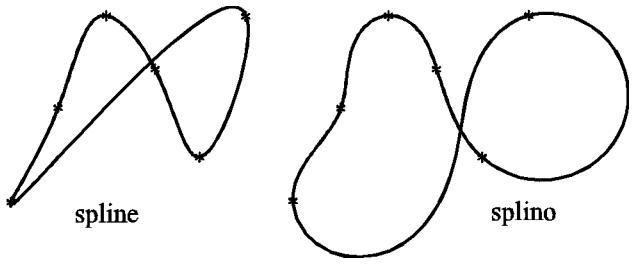
```
      CZ(   1)=-(2*DYX(I)+DYX(J))*H(I)/6
      CX(NS+1)=-(DZY(I)+2*DZY(J))*H(I)/6
      CY(NS+1)=-(DXZ(I)+2*DXZ(J))*H(I)/6
      CZ(NS+1)=-(DYX(I)+2*DYX(J))*H(I)/6
      CALL ABCXYZ(XT(1),YT(1),ZT(1),XT(NS+1),YT(NS+1),ZT(NS+1)
     *           ,CX(1),CY(1),CZ(1),CX(NS+1),CY(NS+1),CZ(NS+1))
      IF(NS.GT.128) WRITE(*,'(''NS > 128'')')
      IX=NS
   20 DO 50 IA=1,NS,IX
      IB=IA+IX
      IC=(IA+IB)/2
      CALL SPLINQ
     *(XT(IA),YT(IA),ZT(IA),XT(IB),YT(IB),ZT(IB),XT(IC),YT(IC),ZT(IC)
     *,CX(IA),CY(IA),CZ(IA),CX(IB),CY(IB),CZ(IB),CX(IC),CY(IC),CZ(IC))
   50 CONTINUE
      IX=IX/2
      IF(IX.GT.1) GO TO 20
      RETURN
      END
************************************************************************
      SUBROUTINE SPLINP(XA,YA,XB,YB,XC,YC,XYA,YXB,XYC,YXC)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C**   Find XC,YC,XYA,YXB,XYC,YXC from XA,YA,XB,YB,XYA,YXB             **
C**   ------------------------------------------------------------    **
C*I   XA,YA = Coordinates of left  end point A                        **
C*I   XB,YB = Coordinates of right end point B                        **
C*O   XC,YC = Coordinates of middle point C                           **
C*I   XYA   = Angle from line AB to tangent of curve AB at A         **
C*I   YXB   = Angle from line BA to tangent of curve BA at B         **
C*O   XYA   = Angle from line AC to tangent of curve AC at A         **
C*O   YXC   = Angle from line CA to tangent of curve CA at C         **
C*O   XYC   = Angle from line CB to tangent of curve CB at C         **
C*O   YXB   = Angle from line BC to tangent of curve BC at B         **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | CXY = Angle from line AB to line AC or from BA to BC     |    **
C**   +----------------------------------------------------------+    **
      CXY=(YXB+XYA)*0.25
      SYX=(YXB-XYA)*0.25
      YXC=CXY-SYX
      XYC=CXY+SYX
      XYA=YXC-SYX
      YXB=XYC+SYX
      DXY=DSIN(CXY)/DCOS(CXY)
      XC=0.5*((XB+XA)-(YB-YA)*DXY)
      YC=0.5*((YB+YA)+(XB-XA)*DXY)
      END
************************************************************************
      SUBROUTINE SPLINQ(XA,YA,ZA,XB,YB,ZB,XC,YC,ZC
     *                 ,TXA,TYA,TZA,TXB,TYB,TZB,TXC,TYC,TZC)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C**   Find (XC,YC,ZC),(TXC,TYC,TZC)                                   **
C**   From (XA,YA,ZA),(XB,YB,ZB),(TXA,TYA,TZA),(TXB,TYB,TZB)          **
C**   ------------------------------------------------------------    **
C*I   XA,YA,ZA    = Coordinates of left  end point A                  **
C*I   XB,YB,ZB    = Coordinates of right end point B                  **
```

```
C*O   XC,YC,ZC    = Coordinates of middle point C                    **
C*I   TXA,TYA,TZA = Unit vector of tangent of curve ACB at A         **
C*I   TXB,TYB,TZB = Unit vector of tangent of curve ACB at C         **
C*O   TXC,TYC,TZC = Unit vector of tangent of curve ACB at B         **
C**   ================================================================    **
C**   +-------------------------------------------------------------+    **
C**   | TX,TY,TZ = Unit vector of line AB                           |    **
C**   +-------------------------------------------------------------+    **
      TX=XB-XA
      TY=YB-YA
      TZ=ZB-ZA
      H=DSQRT(TX*TX+TY*TY+TZ*TZ)
      TX=TX/H
      TY=TY/H
      TZ=TZ/H
C**   ----------------------------------------------------------------    **
      CALL VCROSS(TX,TY,TZ,TXA,TYA,TZA,AX,AY,AZ,AL)
      CALL VCROSS(TXB,TYB,TZB,TX,TY,TZ,BX,BY,BZ,BL)
      CALL VCROSS(BX,BY,BZ,AX,AY,AZ,VX,VY,VZ,VL)
      CA=TX*TXA+TY*TYA+TZ*TZA
      CB=TX*TXB+TY*TYB+TZ*TZB
      SA=DSQRT(1-CA*CA)
      SB=DSQRT(1-CB*CB)
      A=DATAN2(SA,CA)
      B=DATAN2(SB,CB)
C**   +-------------------------------------------------------------+    **
C**   | CX,CY,CZ = Unit vector normal to plane ABC                  |    **
C**   | DX,DY,DZ = Unit vector normal to line AB and vector C       |    **
C**   | CL = Angle from line AB to line AC                          |    **
C**   | DL = Distance from center of line AB to point C             |    **
C**   +-------------------------------------------------------------+    **
      CX=AX*A+BX*B
      CY=AY*A+BY*B
      CZ=AZ*A+BZ*B
      CL=DSQRT(CX*CX+CY*CY+CZ*CZ)
      IF(CL.NE.0) THEN
      CX=CX/CL
      CY=CY/CL
      CZ=CZ/CL
      ENDIF
      CALL VCROSS(CX,CY,CZ,TX,TY,TZ,DX,DY,DZ,DL)
      DL=0.5*H*DTAN(0.25*CL)
      XC=0.5*(XA+XB)+DX*DL
      YC=0.5*(YA+YB)+DY*DL
      ZC=0.5*(ZA+ZB)+DZ*DL
C**   +-------------------------------------------------------------+    **
C**   | PX,PY,PZ = Vector difference of vector A and vector B       |    **
C**   | PC/4 = -Rotation angle about (CX,CY,CZ) at point C          |    **
C**   | PD/4 =  Rotation angle about (DX,DY,DZ) at point C          |    **
C**   +-------------------------------------------------------------+    **
      PX=AX*A-BX*B
      PY=AY*A-BY*B
      PZ=AZ*A-BZ*B
      PC=PX*CX+PY*CY+PZ*CZ
      PD=DSIGN(DSQRT(PX*PX+PY*PY+PZ*PZ-PC*PC),TX*VX+TY*VY+TZ*VZ)
      SC=DSIN(0.25*PC)
      CC=DSQRT(1-SC*SC)
      SD=DSIN(0.25*PD)
      CD=DSQRT(1-SD*SD)
      TXC=(TX*CD-CX*SD)*CC-DX*SC
```

```
      TYC=(TY*CD-CY*SD)*CC-DY*SC
      TZC=(TZ*CD-CZ*SD)*CC-DZ*SC
      RETURN
      END
*****************************************************************************
      SUBROUTINE VCROSS(AX,AY,AZ,BX,BY,BZ,CX,CY,CZ,CL)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C**   Cross product of two vectors:                                   **
C**   / Cx \     / Ax \   / Bx \     / 0  -Az  Ay \ / Bx \            **
C**   | Cy |  =  | Ay | x | By |  =  | Az  0  -Ax | | By |            **
C**   \ Cz /     \ Az /   \ Bz /     \-Ay  Ax  0  / \ Bz /            **
C**   And normalize to unit vector                                    **
C**   ============================================================    **
      CX=      -AZ*BY+AY*BZ
      CY= AZ*BX      -AX*BZ
      CZ=-AY*BX+AX*BY
      CL=DSQRT(CX*CX+CY*CY+CZ*CZ)
      IF(CL.EQ.0) RETURN
      CX=CX/CL
      CY=CY/CL
      CZ=CZ/CL
      RETURN
      END
*****************************************************************************
      SUBROUTINE ABCXYZ(XA,YA,ZA,XB,YB,ZB,AX,AY,AZ,BX,BY,BZ)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C*I   XA,YA,ZA = Coordinates of node A                                 **
C*I   XB,YB,ZB = Coordinates of node B                                 **
C*I   AX,AY,AZ = Angles from vector AB to tangent of curve AB at A     **
C*I   BX,BY,BZ = Angles from vector BA to tangent of curve BA at B     **
C*O   AX,AY,AZ = Unit vector of tangent of curve AB at A               **
C*O   BX,BY,BZ = Unit vector of tangent of curve AB at B               **
C**   ============================================================    **
      RX=DATAN2(ZB-ZA,YB-YA)
      RY=DATAN2(XB-XA,ZB-ZA)
      RZ=DATAN2(YB-YA,XB-XA)
C**   ------------------------------------------------------------    **
      RAX=RX+AX
      RAY=RY+AY
      RAZ=RZ+AZ
      IF(DABS(RAX).GT.3.14159) RAX=RAX-DSIGN(6.28318D0,RAX)
      IF(DABS(RAY).GT.3.14159) RAY=RAY-DSIGN(6.28318D0,RAY)
      IF(DABS(RAZ).GT.3.14159) RAZ=RAZ-DSIGN(6.28318D0,RAZ)
      PX=DTAN(RAX)
      PY=DTAN(RAY)
      PZ=DTAN(RAZ)
      PP=DABS(PX*PY*PZ)**(1.0D0/3)
      PX=(PX/PP)**2
      PY=(PY/PP)**2
      PZ=(PZ/PP)**2
      AX=DSIGN(1.0D0,RAY)/DSQRT(1+PZ+PZ*PX)
      AY=DSIGN(1.0D0,RAZ)/DSQRT(1+PX+PX*PY)
      AZ=DSIGN(1.0D0,RAX)/DSQRT(1+PY+PY*PZ)
C**   ------------------------------------------------------------    **
      RBX=RX-BX
      RBY=RY-BY
```

```
      RBZ=RZ-BZ
      IF(DABS(RBX).GT.3.14159) RBX=RBX-DSIGN(6.28318D0,RBX)
      IF(DABS(RBY).GT.3.14159) RBY=RBY-DSIGN(6.28318D0,RBY)
      IF(DABS(RBZ).GT.3.14159) RBZ=RBZ-DSIGN(6.28318D0,RBZ)
      QX=DTAN(RBX)
      QY=DTAN(RBY)
      QZ=DTAN(RBZ)
      QQ=DABS(QX*QY*QZ)**(1.0D0/3)
      QX=(QX/QQ)**2
      QY=(QY/QQ)**2
      QZ=(QZ/QQ)**2
      BX=DSIGN(1.0D0,RBY)/DSQRT(1+QZ+QZ*QX)
      BY=DSIGN(1.0D0,RBZ)/DSQRT(1+QX+QX*QY)
      BZ=DSIGN(1.0D0,RBX)/DSQRT(1+QY+QY*QZ)
      RETURN
      END
*************************************************************************
      SUBROUTINE SPLINL(H,XT,YT,ZT,NS,M)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XT(1),YT(1),ZT(1)
C**   ============================================================    **
C**   H = Curve length of XT(NS+1),YT(NS+1),ZT(NS+1)                  **
C**   ============================================================    **
      H=0.0
      IF(M.NE.3) THEN
        DO 20 K=1,NS
        H=H+DSQRT((XT(K+1)-XT(K))**2+(YT(K+1)-YT(K))**2)
   20   CONTINUE
      ELSE
        DO 30 K=1,NS
        H=H
     *  +DSQRT((XT(K+1)-XT(K))**2+(YT(K+1)-YT(K))**2+(ZT(K+1)-ZT(K))**2)
   30   CONTINUE
      ENDIF
      RETURN
      END
*************************************************************************
C**   TEST PROGRAM FOR SPLINE CURVE FITTING                           **
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION H(20),X(20),Y(20),Z(20),DZY(20),DXZ(20),DYX(20)
      DIMENSION A(60),L(20),XT(129),YT(129),ZT(129)
C**   ============================================================    **
C**   N  = Number of given nodes                                      **
C**   M  = 2 or 3 = Dimensions of the curve                           **
C**   KB = 0:4 see SUBROUTINE SPLINE                                  **
C**   KL = 1 : set H(I) = 1 by SUBROUTINE SPLINE                      **
C**      = 0 : set H(I) = Length between nodes I & J by SUB.SPLINE    **
C**      < 0 : use H(I) = Length of the curve IJ get by SUB.SPLINL    **
C**            -KL = Number of iterations to get H(I)   (2 is OK)     **
C**   NL = Number of line segments for a curve between two nodes      **
C**   DZY1,DZYN,...  = See SUBROUTINE SPLINO                          **
C**   X(N),Y(N),Z(N) = Coordinates of nodes                           **
C**   ============================================================    **
   10 READ(*,'(5I5,6F5.0)') N,M,KB,KL,NL,DZY1,DZYN,DXZ1,DXZN,DYX1,DYXN
      IF(M.NE.3) M=2
      KH=MAX0(KL,0)
      NS=8
      DO 15 NN=1,4
```

```
   15 IF(NS.LT.NL) NS=2*NS
      READ(*,'(3F10.0)') (X(I),Y(I),Z(I),I=1,N)
      WRITE(*,'(5I5,6F5.2)') N,M,KB,KL,NS,DZY1,DZYN,DXZ1,DXZN,DYX1,DYXN
C**   ------------------------------------------------------------    **
   20 CALL SPLINE(H,X,Y,Z,N,M,KB,KH,A,L)
C**   ------------------------------------------------------------    **
      CALL SPLINO(X,Y,DYX,N,KB,DYX1,DYXN,A,L)
      WRITE(*,'('' XY'',1P6E13.5)') (DYX(I),I=1,N)
      IF(M.EQ.3) THEN
      CALL SPLINO(Y,Z,DZY,N,KB,DZY1,DZYN,A,L)
      CALL SPLINO(Z,X,DXZ,N,KB,DXZ1,DXZN,A,L)
      WRITE(*,'('' YZ'',1P6E13.5)') (DZY(I),I=1,N)
      WRITE(*,'('' ZX'',1P6E13.5)') (DXZ(I),I=1,N)
      ENDIF
C**   ------------------------------------------------------------    **
      WRITE(*,'(4X,''I    K      X(t)      Y(t)      Z(t)'')')
      DO 50 I=1,N
      IF(H(I).EQ.0.0) GO TO 50
      N1=MIN0(I,2)
      IF(M.NE.3) THEN
        CALL SPLIN2(H,X,Y,DYX,N,I,NS,XT,YT)
        IF(KH.GT.KL) THEN
          CALL SPLINL(H(I),XT,YT,ZT,NS,M)
        ELSE
          WRITE(*,'(2I5,2F10.5)') (I,K-1,XT(K),YT(K),K=N1,NS+1)
        ENDIF
      ELSE
        CALL SPLIN3(H,X,Y,Z,DZY,DXZ,DYX,N,I,NS,XT,YT,ZT)
        IF(KH.GT.KL) THEN
          CALL SPLINL(H(I),XT,YT,ZT,NS,M)
        ELSE
          WRITE(*,'(2I5,3F10.5)') (I,K-1,XT(K),YT(K),ZT(K),K=N1,NS+1)
        ENDIF
      ENDIF
   50 CONTINUE
      KH=KH-1
      IF(KH.GE.KL) GO TO 20
      GO TO 10
      END
*************************************************************************
```



圖二 二種方式之近似曲線

習題

1. 假設y為x，z之函數，即$y = f(x, z)$。若已知$z = z_k,\ k = 1, 2, \cdots, M$時之$y_{ik} = f(x_{ik}, z_k),\ i = 1, 2, \cdots, N$。即已知$M$個如下之函數表：

   當$z_k = Z(K)$時：

x_{ik}	$X(1,k)$	$X(2,k)$	$\cdots$	$X(N,k)$
y_{ik}	$Y(1,k)$	$Y(2,k)$	$\cdots$	$Y(N,k)$

   則對任一已知之$x = XX$，$z = ZZ$欲插值計算$YY = f(XX, ZZ)$時，可按下述步驟：

   (a) 當$z = Z(K)$時，由XX利用$X(*, K)$與$Y(*, K)$之關係表插值計算$YK(K),\ K = 1, 2, \cdots, M$。

   (b) 由ZZ利用$Z(*)$與$YK(*)$之關係表插值計算YY。

   試按上述方法寫一副程式$INT3D(X, Y, Z, N, M, XX, YY, ZZ, \cdots)$，其中$DIMENSION$為$X(N, M), Y(N, M), Z(M), YK(M), \cdots$，以由已知之$X, Y, Z, N, M, XX, ZZ$計算$YY$。注意上述兩步驟之插值計算均可呼叫同一副程式$LAGRAG$或$DIFINT$。

2. 上題所述之插值運算須分兩個階段，第一階段先算出所有$YK(K)$值，第二階段再由$YK(K)$與$Z(K)$計算YY。但亦可改為：先考慮由ZZ利用$Z(K)$與$YK(K)$插值計算$YK(K)$。而在需要$YK(K)$值時再由$Z(K)$利用$X(*, K)$與$Y(*, K)$計算$YK(K)$。這樣可以根據所須精度決定是否需要繼續計算另一個$YK(*)$值，而不必將所有M個$YK(*)$值預先算出。
   試再按前述方法將副程式$DIFINT$修改成另一副程式，用以由ZZ利用$Z(K)$與$YK(K)$計算YY，其中之$YK(K)$仍可呼叫未修改之$DIFINT$計算。

3. 試寫一副程式$AITKEN$以葉特肯插值公式計算插值函數之係數。

4. 試寫一副程式$LAGRAN$以拉格蘭治插值公式計算插值函數之係數。

5. 試按式(8.47)至式(8.50)寫有關副程式以計算樑曲線之y_i'。

6. 三維曲線之近似亦可用$SPLINO$求XY平面之曲線及由$SPLINF$求Z方向之座標，試寫一主程式為之。

第九章

曲線近似法

9.1 前言

前章函數插值法中所介紹之方法為如何求一條曲線"通過"已知點，再用以計算其他x值之函數$y(x)$。這些方法適用於已知曲線上之某些點而欲求取其他點之情形。但是，當某一曲線上之點係由實驗所得，而實驗難免有誤差存在，故這些已知點不太可能正好在曲線上，因此通過這些點之曲線即非所要之曲線而不適用。本章所介紹之曲線近似法 (Curve fitting) 所求得之曲線並不通過已知點，但會使已知點與近似曲線上之點之誤差平方和為最小。為了使所得曲線之誤差較小，實驗之已知點愈多愈好，因此用以計算曲線之已知點數通常會很多，並不像函數插值法那樣m次曲線僅用m+1個點即可求得。當欲近似之曲線為多項式函數時，多項式之係數可以由線性方程式求得而較簡單。另外當函數為指數形式時，其指數與係數亦可由線性方程式求出。本章將介紹這二種簡單情形之解法。

有些函數之近似，如該函數為某系統之反應或輸出，影響該函數之參數(相當於多項式函數之係數)為系統之參數。如何求得系統參數使系統之反應與實驗所得者近似，一般稱為系統識別，也是實驗上常會遇到的問題。雖然二者目的(求函數之參數)相同，所用條件(即誤差平方和為最小)也相同，但系統參數之計算，通常不像多項式之係數一樣，可由線性方程式求得。一般常須由(非線性)最佳化方法，令誤差平方和為目標函數，以求取系統參數。有關之詳細做法將於另章說明。

9.2 最小二乘法

曲線近似法會用到下列線性聯立式之近似解法，因此先行介紹：

$$[A]_{n\times m}\{X\}_{m\times 1}=\{B\}_{n\times 1} \tag{9.1}$$

在第三章曾討論過$n=m$時上式之解法，在本章將討論$n>m$時上式之求法。當$n>m$時，除非上列聯立方程式係由不大於m個獨立方程式之線性組合，否則該方程式不可能有解。但有許多工程上的問題，只需求得一組$\{X\}$使其近似滿足式(9.1)即可。使(9.1)近似滿足之方式很多，最常用且最容易之做法是使式(9.1)中每一個方程式的誤差平方和S為最小。此種做法稱為最小誤差平方和法，或簡稱最小二乘法(Least square method)。設式(9.1)第i個方程式之誤差為r_i，且令$\{R\}=\langle r_1,r_2,\ldots,r_n\rangle^T$，則

$$\{R\}=[A]\{X\}-\{B\} \tag{9.2}$$

因此誤差平方和S可寫成

$$S=\sum_{i=1}^{n}r_i^2=\langle\ r_1\quad r_2\quad\cdots\quad r_n\ \rangle\begin{Bmatrix}r_1\\r_2\\\vdots\\r_n\end{Bmatrix}=\{R\}^T\{R\} \tag{9.3}$$

將式(9.2)代入式(9.3)，得

$$\begin{aligned}S&=\{[A]\{X\}-\{B\}\}^T\{[A]\{X\}-\{B\}\}\\&=\left(\{X\}^T[A]^T-\{B\}^T\right)\left([A]\{X\}-\{B\}\right)\\&=\{X\}^T[A]^T[A]\{X\}-\{B\}^T[A]\{X\}-\{X\}^T[A]^T\{B\}+\{B\}^T\{B\}\\&=\{X\}^T[A]^T[A]\{X\}-2\{X\}^T[A]^T\{B\}+\{B\}^T\{B\}\end{aligned} \tag{9.4}$$

欲使S為最小之條件為

$$\begin{Bmatrix}\frac{\partial S}{\partial x_1}\\\frac{\partial S}{\partial x_2}\\\vdots\\\frac{\partial S}{\partial x_m}\end{Bmatrix}=\begin{Bmatrix}0\\0\\\vdots\\0\end{Bmatrix}\qquad\text{或}\qquad\left\{\frac{\partial S}{\partial x_i}\right\}=\{0\} \tag{9.5}$$

由式(9.4)對x_i偏微分可得

$$\left\{\frac{\partial S}{\partial x_i}\right\} = 2\,[A]^T[A]\,\{X\} - 2\,[A]^T\{B\} \tag{9.6}$$

代入式(9.5)即得使S為最小之條件為

$$[A]^T[A]\,\{X\} \;=\; [A]^T\{B\} \tag{9.7}$$

上列方程式左邊係數矩陣$[A]^T[A]$為$m \times m$之方陣，右邊常數矩陣$[A]^T\{B\}$為$m \times 1$之行矩陣，可用第三章之方法解得$\{X\}$，亦可用下節介紹之方法將$[A]$矩陣分解為$[Q]\,[U]$後再求$\{X\}$。

如果每一方程式的誤差平方後先乘以一對應之權w_i再相加得S，則

$$\begin{aligned} S \;&=\; \sum_{i=1}^{n} w_i\, r_i^2 = \langle\; r_1 \quad r_2 \quad \cdots \quad r_n \;\rangle \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_n \end{bmatrix} \begin{Bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{Bmatrix} \\ &=\; \{R\}^T[W]\,\{R\} \end{aligned} \tag{9.8}$$

同樣以式(9.2)代入，並令$\{\frac{\partial S}{\partial x_i}\} = \{0\}$，可得使$S$為最小之條件為

$$[A]^T[W]\,[A]\,\{X\} \;=\; [A]^T[W]\,\{B\} \tag{9.9}$$

9.3 $[Q]\,[U]$ 分解法

式(9.7)雖然可用第三章之方法直接求解，但可能會有較大之誤差，特別當x_i值或y_i值相當接近時，另詳第十一章，因此常須用倍精度數計算。以下介紹另一方法又稱Gram–Schmidt正交化法，係直接用$[A]$矩陣計算，將其分解成正交矩陣$[Q]$與上三角矩陣$[U]$之乘積。若$[A]$為$n \times m$，$n > m$，則$[Q]$亦為$n \times m$，且$[Q]^T[Q] = [I]$，即$[Q]$矩陣之每一行為n元之單位向量$\{q_i\}_{n\times 1}$，$i = 1, 2, \ldots, m$，而這m個向量為互相正交，即$\{q_i\}^T\{q_j\} = \delta_{ij}$。同樣令向量$\{a_i\}$為矩陣$[A]$之第$i$行，以$m$=4為例，則$[A] = [Q]\,[U]$可寫成

$$\Big[\{a_1\}\;\{a_2\}\;\{a_3\}\;\{a_4\}\Big] = \Big[\{q_1\}\;\{q_2\}\;\{q_3\}\;\{q_4\}\Big] \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ & u_{22} & u_{23} & u_{24} \\ & & u_{33} & u_{34} \\ & & & u_{44} \end{bmatrix} \tag{9.10}$$

由上式可得

$$\{a_1\} = \{q_1\}\,u_{11} \tag{9.11}$$
$$\{a_2\} = \{q_1\}\,u_{12} + \{q_2\}\,u_{22} \tag{9.12}$$
$$\{a_3\} = \{q_1\}\,u_{13} + \{q_2\}\,u_{23} + \{q_3\}\,u_{33} \tag{9.13}$$
$$\{a_4\} = \{q_1\}\,u_{14} + \{q_2\}\,u_{24} + \{q_3\}\,u_{34} + \{q_4\}\,u_{44} \tag{9.14}$$

式(9.11)分別前乘$\{a_1\}^T$與$u_{11}\,\{q_1\}^T$可得

$$\{a_1\}^T\{a_1\} = u_{11}\,\{q_1\}^T\{q_1\}\,u_{11} = u_{11}^2$$
$$u_{11} = \sqrt{\{a_1\}^T\{a_1\}} \tag{9.15}$$

式(9.12)前乘$\{q_1\}^T$可得

$$u_{12} = \{q_1\}^T\{a_2\} \tag{9.16}$$

令$\{\bar{a}_2\}$如下式，再分別前乘$\{\bar{a}_2\}^T$與$u_{22}\,\{q_2\}^T$可得

$$\{\bar{a}_2\} = \{a_2\} - \{q_1\}\,u_{12} = \{q_2\}\,u_{22}$$
$$\{\bar{a}_2\}^T\{\bar{a}_2\} = u_{22}\,\{q_2\}^T\{q_2\}\,u_{22} = u_{22}^2$$
$$u_{22} = \sqrt{\{\bar{a}_2\}^T\{\bar{a}_2\}} \tag{9.17}$$

同理可得

$$u_{13} = \{q_1\}^T\{a_3\}\,, \qquad u_{23} = \{q_2\}^T\{a_3\} \tag{9.18}$$
$$\{\bar{a}_3\} = \{a_3\} - \{q_1\}\,u_{13} - \{q_2\}\,u_{23} = \{q_3\}\,u_{33}$$
$$u_{33} = \sqrt{\{\bar{a}_3\}^T\{\bar{a}_3\}} \tag{9.19}$$

其一般式可寫成

$$u_{ij} = \{q_i\}^T\{a_j\}\,, \qquad i = 1, 2, \ldots, j-1 \tag{9.20}$$
$$\{\bar{a}_j\} = \{a_j\} - \sum_{i=1}^{j-1}\{q_i\}\,u_{ij} \tag{9.21}$$
$$u_{jj} = \sqrt{\{\bar{a}_j\}^T\{\bar{a}_j\}} \tag{9.22}$$

將$[A] = [Q]\,[U]$代入式(9.7)可得$[U]^T[U]\{X\} = [U]^T[Q]^T\{B\}$，或

$$[U]\,\{X\} = [Q]^T\{B\} \tag{9.23}$$

由上式利用反向代入即可求得$\{X\}$。

9.4 曲線近似法

在許多實驗分析中，常希望從如[表一]所示之$n+1$組數據中，得到二個參數y與x間的關係如下：

$$y = c_m x^m + \cdots + c_1 x + c_o \tag{9.24}$$

[表一] 已知實驗數據

x_i	x_o	x_1	x_2	$\cdots$	x_n
y_i	y_o	y_1	y_2	$\cdots$	y_n

式(9.24)為x之m次多項式。前章曾探討過，在x, y平面內，通過$m+1$點可以找出唯一的一條m次曲線如式(9.24)所示。如果$n>m$，則不一定能找到一條m次函數曲線通過所有n點，但是可以有一條m次函數曲線：

$$Y(x) = c_m x^m + \cdots + c_1 x + c_o \tag{9.25}$$

使該曲線上之點與n已知點間之誤差平方和或誤差加權平方和

$$S = \sum_{i=o}^{n} (Y_i - y_i)^2 \qquad \text{或} \qquad S = \sum_{i=o}^{n} w_i (Y_i - y_i)^2 \tag{9.26}$$

為最小。其中$Y_i = Y(x_i)$。此問題相當於在$n>m$時求解下列聯立方程式

$$\begin{bmatrix} x_o^m & \cdots & x_o & 1 \\ x_1^m & \cdots & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^m & \cdots & x_n & 1 \end{bmatrix} \begin{Bmatrix} c_m \\ \vdots \\ c_1 \\ c_o \end{Bmatrix} = \begin{Bmatrix} y_o \\ y_1 \\ \vdots \\ y_n \end{Bmatrix} \tag{9.27}$$

之$\{c_m, \ldots, c_1, c_o\}$，使各式的誤差平方和或加權平方和為最小。若令

$$[A] = \begin{bmatrix} x_o^m & \cdots & x_o & 1 \\ x_1^m & \cdots & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_n^m & \cdots & x_n & 1 \end{bmatrix}, \quad \{X\} = \begin{Bmatrix} c_m \\ \vdots \\ c_1 \\ c_o \end{Bmatrix}, \quad \{B\} = \begin{Bmatrix} y_o \\ y_1 \\ \vdots \\ y_n \end{Bmatrix} \tag{9.28}$$

即可照前節所述之過程求得$\{X\}$，即$\{c_m, \ldots, c_1, c_o\}$。

9.5 指數函數之近似

除前節之多項式函數可以由線性聯立式求得函數之係數外，另外一種常用以近似[表二]之實驗數據之指數形式函數，如下式所示，亦可由線性聯立式求得其中之指數a與b及係數c。不過採用之誤差改為$r_i = \ln Z_i - \ln z_i$；非前節所用之$r_i = Z_i - z_i$。其中$Z_i = Z(x_i, y_i)$。

$$Z(x,y) = c\,x^a\,y^b \tag{9.29}$$

對上式兩邊取對數，得

$$\ln Z(x,y) = a\,\ln x + b\,\ln y + \ln c \tag{9.30}$$

[表二] 已知實驗數據

x_i	x_o	x_1	x_2	$\cdots$	x_n
y_i	y_o	y_1	y_2	$\cdots$	y_n
z_i	z_o	z_1	z_2	$\cdots$	z_n

若欲使曲線之近似值之$\ln Z_i$與實驗值之$\ln z_i$之差之平方和或加權平方和為最小，即

$$S = \sum_{i=o}^{n} (\ln Z_i - \ln z_i)^2 \qquad \text{或} \qquad S = \sum_{i=o}^{n} w_i\,(\ln Z_i - \ln z_i)^2 \tag{9.31}$$

則相當於求解下列聯立方程式

$$\begin{bmatrix} \ln x_o & \ln y_o & 1 \\ \ln x_1 & \ln y_1 & 1 \\ \vdots & \vdots & \vdots \\ \ln x_n & \ln y_n & 1 \end{bmatrix} \begin{Bmatrix} a \\ b \\ \ln c \end{Bmatrix} = \begin{Bmatrix} \ln z_o \\ \ln z_1 \\ \vdots \\ \ln z_n \end{Bmatrix} \tag{9.32}$$

之$\{a, b, \ln c\}$，使各式的誤差平方和或加權平方和為最小。若令

$$[A] = \begin{bmatrix} \ln x_o & \ln y_o & 1 \\ \ln x_1 & \ln y_1 & 1 \\ \vdots & \vdots & \vdots \\ \ln x_n & \ln y_n & 1 \end{bmatrix}, \quad \{X\} = \begin{Bmatrix} a \\ b \\ \ln c \end{Bmatrix}, \quad \{B\} = \begin{Bmatrix} \ln z_o \\ \ln z_1 \\ \vdots \\ \ln z_n \end{Bmatrix} \tag{9.33}$$

則亦可照前節所述之過程求得$\{X\}$，即$\{a, b, \ln c\}$，而$c = e^{\ln c}$。

9.6 副程式$QUSOL$

[表三]為以最小二乘法求式(9.1)之近似解之副程式。其中

$A(NN,M)$	=	$n\times m$之$[A]$矩陣。
$B(N)$	=	$\{B\}$向量。
$Q(NN,M)$	=	用以儲存$[Q]$之臨時矩陣。
$U(MM,M)$	=	用以儲存$[U]$之臨時向量。
$X(M)$	=	$[A]\{X\}=\{B\}$之近似解。
N	=	n。
M	=	m。
NN	=	$[A]$矩陣之宣告列數。
MM	=	$[U]$矩陣之宣告列數。

[表四] $QUSOL$副程式

```
************************************************************************
      SUBROUTINE QUSOL(A,B,Q,U,X,N,M,NN,MM)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NN,M),B(N),Q(NN,M),U(MM,M),X(M)
      DATA EPS/1.0D-10/
C**   ================================================================    **
C*I   A(N,M),B(N),N,M,NN                                                  **
C*O   Q(N,M),U(M,M),X(M) (Q(N,M) & A(N,M) can be the same matrix)         **
C**   ----------------------------------------------------------------    **
C**   Solve A'(M,N)*A(N,M)*X(M) = A'(M,N)*B(N)                            **
C**   By orthogonal decomposition A(N,M) = Q(N,M)*U(M,M)                  **
C**   Where Q(N,M) is an orthogonal matrix Q'(M,N)*Q(N,M) = I(M,M)        **
C**    and  U(M,M) is an upper-triangular matrix                          **
C**   Get X(M) by backward substitution U(M,M)*X(M) = Q'(M,N)*B(N)        **
C**   ================================================================    **
      DO 70 J=1,M
      XSQS=0.0
      DO 10 I=1,N
      XSQS=XSQS+A(I,J)**2
   10 Q(I,J)=A(I,J)
C**   +--------------------------------------------------------+          **
C**   | Compute Q(I,J) & U(J,J)                                |          **
C**   +--------------------------------------------------------+          **
      DO 30 I=1,J-1
      UIJ=0.0
      DO 20 K=1,N
   20 UIJ=UIJ+Q(K,I)*Q(K,J)
      U(I,J)=UIJ
      DO 25 K=1,N
   25 Q(K,J)=Q(K,J)-Q(K,I)*UIJ
   30 CONTINUE
C**   ----------------------------------------------------------------    **
      UJJ=0.0
      DO 40 K=1,N
```

```
   40 UJJ=UJJ+Q(K,J)**2
      IF(UJJ.GT.XSQS*EPS) THEN
        UJJ=DSQRT(UJJ)
        U(J,J)=UJJ
        DO 50 K=1,N
   50   Q(K,J)=Q(K,J)/UJJ
      ELSE
        U(J,J)=1.0
        DO 55 K=1,N
   55   Q(K,J)=0.0
        WRITE(*,'(I5,A)') J,'-th vector is not independent, set X(j)=0'
      ENDIF
C**   +-------------------------------------------------------------+    **
C**   | Compute Q(1:N,J)*B(1:N) --> X(J)  :  [Q]'{B}                 |    **
C**   +-------------------------------------------------------------+    **
      X(J)=0.0
      DO 60 K=1,N
   60 X(J)=X(J)+Q(K,J)*B(K)
   70 CONTINUE
C**   +-------------------------------------------------------------+    **
C**   | Solve U(M,M)*X(M) = Q(N,M)*B(N)  :  [U]{X}=[Q]'{B}           |    **
C**   +-------------------------------------------------------------+    **
      DO 90 J=M,1,-1
      X(J)=X(J)/U(J,J)
      DO 80 I=1,J-1
   80 X(I)=X(I)-U(I,J)*X(J)
   90 CONTINUE
C**   ---------------------------------------------------------------    **
C     DO 95 J=1,M
C  95 WRITE(*,'('' U:'',1P6E12.5)') (U(I,J),I=1,J)
C     WRITE(*,'('' X:'',1P6E12.5)') X
      RETURN
      END
*************************************************************************
```

[表四]之主程式，係用以讀入矩陣$[A],\{B\}$與$[W]$，計算$[W]^{1/2}[A]$與$[W]^{1/2}\{B\}$，並呼叫副程式$QUSOL$。求得$\{X\}$後並將結果印出。主程式後為二個算例之輸入資料與計算結果。

[表五] 最小二乘法試用程式

```
*************************************************************************
      PROGRAM LSQMN
C**   =============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(100,10),B(100),W(100)
      DIMENSION Q(100,10),C(100),U(10,10),X(10)
      DATA NMAX/100/,MMAX/10/
C**   =============================================================    **
C**   Program for solving [A]{X}={B} by least squares method
C**   where [A] is a N x M matrix, and N > M
C**   =============================================================    **
   10 READ(*,'(2I4)') N,M
      IF(N.EQ.0.OR.M.EQ.0.OR.N.GT.NMAX.OR.M.GT.MMAX) STOP
      DO 30 I=1,N
      READ(*,'(12F8.0)') (A(I,J),J=1,M),B(I),W(I)
      IF(W(I).LE.0.0) W(I)=1.0
```

```
      WSQ=DSQRT(W(I))
      C(I)=B(I)*WSQ
      DO 20 J=1,M
      Q(I,J)=A(I,J)*WSQ
   20 CONTINUE
   30 CONTINUE
C**   ------------------------------------------------------------    **
      CALL QUSOL(Q,C,Q,U,X,N,M,NMAX,MMAX)
C**   ------------------------------------------------------------    **
      WRITE(*,'(/'' Variables X(M) :''//(1X,1P6E12.5))') (X(J),J=1,M)
      WRITE(*,'(/4X,A/)') 'I       {B}        [A]{X}   {R}=[A]{X}-{B}'
      DO 50 I=1,N
      YI=0.0
      DO 40 J=1,M
      YI=YI+A(I,J)*X(J)
   40 CONTINUE
      RI=YI-B(I)
      WRITE(*,'(I5,1P3E13.5)') I,B(I),YI,RI
   50 CONTINUE
      GO TO 10
      END
*************************************************************************
   5   3
      2.      1.      1.      1.
      3.      2.      1.      3.
      1.      2.      2.      3.
      1.      2.      3.      2.
      1.      2.      1.      4.
   3   3
      2.      1.      1.      1.
      3.      2.      1.      3.
      1.      2.      2.      3.

=========================================================================
Variables X(M) :

-4.18994E-01 2.65363E+00-9.49721E-01

   I       {B}        [A]{X}   {R}=[A]{X}-{B}

   1  1.00000E+00  8.65922E-01 -1.34078E-01
   2  3.00000E+00  3.10056E+00  1.00559E-01
   3  3.00000E+00  2.98883E+00 -1.11732E-02
   4  2.00000E+00  2.03911E+00  3.91061E-02
   5  4.00000E+00  3.93855E+00 -6.14525E-02

Variables X(M) :

-3.33333E-01 2.33333E+00-6.66667E-01

   I       {B}        [A]{X}   {R}=[A]{X}-{B}

   1  1.00000E+00  1.00000E+00  1.33227E-15
   2  3.00000E+00  3.00000E+00  0.00000E+00
   3  3.00000E+00  3.00000E+00  0.00000E+00
=========================================================================
```

9.7 主程式$CURMN$

[表五]之主程式，係用以讀入$\{X\},\{Y\}$及$[W]$，安排式(9.27)，並前乘$[W]^{1/2}$後，呼叫副程式$QUSOL$，以計算$\{c_m,\ldots,c_1,c_o\}$而存於$\{C\}$中，並將結果印出。主程式後為算例之輸入資料與計算結果。

[表五] 曲線近似副程式

```
************************************************************************
      PROGRAM CURMN
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(100),Y(100),W(100)
      DIMENSION A(100,10),B(100),Q(100,10),U(10,10),C(10)
      DATA NMAX/100/,MMAX/10/
C**   ============================================================    **
C**   Program for curve fitting by least squares method
C**   ============================================================    **
   10 READ(*,'(2I4)') N,M
      IF(N.EQ.0.OR.M.EQ.0.OR.N.GT.NMAX.OR.M.GT.MMAX) STOP
      XSUM=0.0
      DO 20 I=1,N
      READ(*,'(3F8.0)') X(I),Y(I),W(I)
      IF(W(I).LE.0.0) W(I)=1.0
      XSUM=XSUM+X(I)
      B(I)=DSQRT(W(I))*Y(I)
   20 A(I,M)=DSQRT(W(I))
      XMEAN=XSUM/N
      DO 30 I=1,N
      DO 30 J=M-1,1,-1
   30 A(I,J)=A(I,J+1)*(X(I)-XMEAN)
C**   ------------------------------------------------------------    **
      CALL QUSOL(A,B,Q,U,C,N,M,NMAX,MMAX)
C**   ------------------------------------------------------------    **
      WRITE(*,'(/'' Coefficients C(M) for C(j)*(X(i)-X_mean)**(M-j)'')')
      WRITE(*,'(/(1X,1P10E12.5))') (C(J),J=1,M)
      WRITE(*,'(/A5,A10,2A13,A14/)') 'I','{x}','{y}','{Y}','{Y}-{y}'
      DO 50 I=1,N
      XI=X(I)-XMEAN
      YI=0.0
      DO 40 J=1,M
      YI=YI*XI+C(J)
   40 CONTINUE
      RI=YI-Y(I)
      WRITE(*,'(I5,1P4E13.5)') I,X(I),Y(I),YI,RI
   50 CONTINUE
      GO TO 10
      END
************************************************************************
  15   3
    0.20 .198669
    0.40 .389418
    0.60 .564642
    0.80 .717356
    1.00 .841471
    1.20 .932039
```

```
    1.40 .985450
    1.60 .999574
    1.80 .973848
    2.00 .909298
    2.20 .808496
    2.40 .675463
    2.60 .515501
    2.80 .334988
    3.00 .141120

=============================================================================
Coefficients C(M) for C(j)*(X(i)-X_mean)**(M-j)

-4.25656E-01-2.31912E-02 9.83645E-01

   I       {x}          {y}          {Y}       {Y}-{y}

   1  2.00000E-01  1.98669E-01  1.81827E-01 -1.68417E-02
   2  4.00000E-01  3.89418E-01  3.98530E-01  9.11220E-03
   3  6.00000E-01  5.64642E-01  5.81181E-01  1.65386E-02
   4  8.00000E-01  7.17356E-01  7.29778E-01  1.24225E-02
   5  1.00000E+00  8.41471E-01  8.44324E-01  2.85292E-03
   6  1.20000E+00  9.32039E-01  9.24817E-01 -7.22214E-03
   7  1.40000E+00  9.85450E-01  9.71257E-01 -1.41927E-02
   8  1.60000E+00  9.99574E-01  9.83645E-01 -1.59287E-02
   9  1.80000E+00  9.73848E-01  9.61981E-01 -1.18672E-02
  10  2.00000E+00  9.09298E-01  9.06264E-01 -3.03412E-03
  11  2.20000E+00  8.08496E-01  8.16494E-01  7.99844E-03
  12  2.40000E+00  6.75463E-01  6.92673E-01  1.72095E-02
  13  2.60000E+00  5.15501E-01  5.34798E-01  1.92971E-02
  14  2.80000E+00  3.34988E-01  3.42871E-01  7.88325E-03
  15  3.00000E+00  1.41120E-01  1.16892E-01 -2.42281E-02
=============================================================================
```

習題

1. 試寫一主程式讀入N及N組(x_i, y_i)，並安排類似式(9.33)之矩陣後呼叫$LSTSQ$以計算$\{a, \ln b\}$之值，使

$$S = \sum_{i=o}^{N-1} (\ln Y_i - \ln y_i)^2$$

為最小，其中$Y_i = b\, x_i^a$。

2. 利用上題程式，以下列數據計算a與b。

x_i	1.22	1.65	2.72	4.48	7.39	12.2	20.1	33.1	54.6	90.0
y_i	13.3	15.5	20.0	25.9	33.3	42.4	54.7	70.0	90.0	114.

第十章

多項式之根

10.1 前言

本章主要介紹多項式根之求法。本節將先介紹幾個有關基本操作。對於下列n次多項式

$$P_n(z) = a_1 z^n + a_2 z^{n-1} + \cdots + a_n z + a_{n+1} \tag{10.1}$$

設以向量$A(m)$存其係數，即$A(i) = a_i,\ i = 1,2,...,m$，其中$m = n+1$。以後各項運算將以$n = 4$為例並輔以程式片斷說明之。

10.1.1 根比例放大（或縮小）：$\bar{z} = rz$

即多項式$\bar{P}(\bar{z})$之根$\bar{z}$為多項式$P(z)$之根z之r倍。令$\bar{P}(\bar{z}) = P(z)$及$\bar{z} = rz$，由式(10.1)可得

$$\begin{aligned} \bar{P}_4(\bar{z}) &= a_1 r^{-4}\bar{z}^4 + a_2 r^{-3}\bar{z}^3 + a_3 r^{-2}\bar{z}^2 + a_4 r^{-1}\bar{z} + a_5 & (10.2)\\ &= \bar{a}_1\bar{z}^4 + \bar{a}_2\bar{z}^3 + \bar{a}_3\bar{z}^2 + \bar{a}_4\bar{z} + \bar{a}_5 & (10.3) \end{aligned}$$

其中$\bar{a}_i = a_i r^{-(m-i)}$。程式片斷為：

```
C**   INPUT  : M=N+1, A(I), R
C**   OUTPUT : A(I)
      P=1.0
      DO 10 I=M,1,-1
      A(I)=A(I)*P
   10 P=P/R
```

注意上列程式片斷執行前之$A(I)$為$P(z)$之係數a_i；執行後之$A(I)$為$\bar{P}(\bar{z})$之係數$\bar{a}_i$。

10.1.2 根向左（或向右）平移：$\bar{z} = z - s$

即多項式 $\bar{P}(\bar{z})$ 之根 $\bar{z}$ 為多項式 $P(z)$ 之根 z 減 s 。令 $\bar{P}(\bar{z}) = P(z)$ 及 $\bar{z} = z - s$ ，由式(10.1)可得

$$\begin{aligned}
\bar{P}_4(\bar{z}) &= a_1(\bar{z}+s)^4 + a_2(\bar{z}+s)^3 + a_3(\bar{z}+s)^2 + a_4(\bar{z}+s) + a_5 && (10.4)\\
&= \bar{a}_1(z-s)^4 + \bar{a}_2(z-s)^3 + \bar{a}_3(z-s)^2 + \bar{a}_4(z-s) + \bar{a}_5 && (10.5)\\
&= \bar{a}_1\bar{z}^4 + \bar{a}_2\bar{z}^3 + \bar{a}_3\bar{z}^2 + \bar{a}_4\bar{z} + \bar{a}_5 && (10.6)
\end{aligned}$$

上列運算可連續利用綜合除法四回合，如下式

$$\begin{aligned}
P_4(z) &= a_1z^4 + a_2z^3 + a_3z^2 + a_4z + a_5 && (10.7)\\
&= (z-s)(b_1z^3 + b_2z^2 + b_3z + b_4) + b_5 && (10.8)\\
&= (z-s)((z-s)(c_1z^2 + c_2z + c_3) + c_4) + b_5 && (10.9)\\
&= (z-s)((z-s)((z-s)(d_1z + d_2) + d_3) + c_4) + b_5 && (10.10)\\
&= (z-s)((z-s)((z-s)((z-s)e_1 + e_2) + d_3) + c_4) + b_5 && (10.11)\\
&= e_1(z-s)^4 + e_2(z-s)^3 + d_3(z-s)^2 + c_4(z-s) + b_5 && (10.12)
\end{aligned}$$

列成運算表則為

	a_1	a_2	a_3	a_4	a_5
$+$		sb_1	sb_2	sb_3	sb_4
	b_1	b_2	b_3	b_4	b_5
$+$		sc_1	sc_2	sc_3	
	c_1	c_2	c_3	c_4	
$+$		sd_1	sd_2		
	d_1	d_2	d_3		
$+$		se_1			
	e_1	e_2			

其中

$$\begin{aligned}
&e_1 = d_1 = c_1 = b_1 = a_1 && (10.13)\\
&b_i = a_i + sb_{i-1}, \quad i = 2,3,4,5 && (10.14)\\
&c_i = b_i + sc_{i-1}, \quad i = 2,3,4 && (10.15)\\
&d_i = c_i + sd_{i-1}, \quad i = 2,3 && (10.16)\\
&e_i = d_i + se_{i-1}, \quad i = 2 && (10.17)
\end{aligned}$$

因此可得$\bar{P}(\bar{z})$之係數$\bar{a}_i$依次為e_1, e_2, d_3, c_4, b_5。其對應程式片斷則為：

```
C**   INPUT  : M=N+1, A(I), S
C**   OUTPUT : A(I)
      DO 10 K=M,2,-1
      DO 10 I=2,K
   10 A(I)=A(I)+S*A(I-1)
```

根向左平移s之運算，除可視為根之位置在原座標系內向左平移s之距離外；亦可視為根之位置不變，但是令座標軸向右平移s之距離，而同樣可得到根之位置相對於新座標軸$\hat{z}$較相對於原座標軸z減少了s值。

10.1.3 根平方變號：$w = -z^2$

即多項式$G(w)$之根w為多項式$P(z)$之根z之平方之負值$(-z^2)$。計算多項式$G(w)$之係數並不能如前述二種情形以直接代入之方式求得，而是由二個多項式相乘而得，其中一個為原多項式$P(z)$，另一個為多項式$P(-z)$，即其根與原多項式之根互為等值異號。由式(10.1)可得

$$P_n(-z) = a_1(-z)^n + a_2(-z)^{n-1} + \cdots + a_n(-z) + a_{n+1} \tag{10.18}$$

故

$$\begin{aligned} P_n(z)P_n(-z) &= a_1^2(-z^2)^n + (a_2^2 - 2a_1a_3)(-z^2)^{n-1} + \cdots \\ &\quad + (a_i^2 + 2\sum_{k=1}^{min(i-1,n+1-i)}(-1)^k a_{i-k}a_{i+k})(-z^2)^{n-i+1} + \\ &\quad \cdots + (a_n^2 - 2a_{n-1}a_{n+1})(-z^2) + a_{n+1}^2 \end{aligned} \tag{10.19}$$

注意，上列多項式無z之奇次方項，可令$w = -z^2$而得

$$\begin{aligned} G_n(w) &= P_n(z)P_n(-z) \\ &= \bar{a}_1 w^n + \bar{a}_2 w^{n-1} + \cdots + \bar{a}_n w + \bar{a}_{n+1} \end{aligned} \tag{10.20}$$

其中

$$\bar{a}_i = a_i^2 + 2\sum_{k=1}^{min(i-1,n+1-i)}(-1)^k a_{i-k}a_{i+k} \tag{10.21}$$

求$\bar{a}_i$=B(I)之程式片斷為：

```
C**   INPUT  : M=N+1, A(I)
C**   OUTPUT : B(I)
      DO 20 I=1,M
      S=2.0
      AA=0.0
      DO 10 K=1,MIN(I-1,M-I)
      S=-S
   10 AA=A(I-K)*A(I+K)-AA
   20 B(I)=A(I)*A(I)+S*AA
```

若$P_n(x)=0$之根為$\alpha_1,\alpha_2,...,\alpha_n$，則

$$P_n(z) = (-1)^n a_1(\alpha_1-z)(\alpha_2-z)\cdots(\alpha_n-z) \tag{10.22}$$

$$P_n(-z) = (-1)^n a_1(\alpha_1+z)(\alpha_2+z)\cdots(\alpha_n+z) \tag{10.23}$$

因此

$$P_n(z)P_n(-z) = a_1^2(\alpha_1^2-z^2)(\alpha_2^2-z^2)\cdots(\alpha_n^2-z^2) \tag{10.24}$$

亦即

$$G_n(w) = a_1^2(\alpha_1^2+w)(\alpha_2^2+w)\cdots(\alpha_n^2+w) \tag{10.25}$$

故得$G_n(w)=0$之根為$-\alpha_1^2,-\alpha_2^2,...,-\alpha_n^2$。

10.2 葛雷非法

葛雷非(Graeffe)法為重覆利用前述根平方變號之操作，使原來大小不相等之二根之比值（小值比大值）經多次（r次，$r\to\infty$）平方後，愈來愈小。而當各大小不相等之根之比值均很小時，各根之大小即可很簡單地由多項式之係數求得。以下說明假設$a_1=1$。由

$$G_n^{(r)}(w) = (\alpha_1^{2^r}+w)(\alpha_2^{2^r}+w)\cdots(\alpha_n^{2^r}+w) \tag{10.26}$$

可知w^{n-i}之係數為

$$a_{i+1}^{(r)} = \sum \alpha_{\beta_1}^{2^r}\alpha_{\beta_2}^{2^r}\cdots\alpha_{\beta_i}^{2^r} = \sum\prod_{k=1}^{i}\alpha_{\beta_k}^{2^r} \tag{10.27}$$

其中$(\beta_1,\beta_2,...,\beta_i)$為自$(1,2,...,n)$等$n$個整數中任取$i$個之組合；$\sum$則為對所有可能之不同組合之和。以$n=4$為例：$i=1$時$(\beta_1)$有$(1),(2),(3),(4)$

等四種組合；$i=2$ 時 (β_1,β_2) 有 $(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)$ 等六種組合；$i=3$ 時 $(\beta_1,\beta_2,\beta_3)$ 有 $(1,2,3),(1,2,4),(1,3,4),(2,3,4)$ 等四種組合；$i=4$ 時 $(\beta_1,\beta_2,\beta_3,\beta_4)$ 僅有 $(1,2,3,4)$ 一種組合。

以下將分三種情況說明，當 $r\to\infty$ 時根 $\alpha_i^{2^r}$ 與係數 $a_i^{(r)}$ 間之關係：

(1) α_i 為相異實數根，且 $|\alpha_1|>|\alpha_2|>\cdots>|\alpha_n|$ 之情形：

$$a_{i+1}^{(r)}=\sum\prod_{k=1}^{i}\alpha_{\beta_k}^{2^r}=\alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_i^{2^r} \tag{10.28}$$

上式第二等號右邊僅取 $\sum$ 中之最大一項，因該項遠大於其他各項。根據上式可以看出下列二項重要之關係：

$$\frac{a_i^{(r)}}{(a_i^{(r-1)})^2}=1 \tag{10.29}$$

$$\frac{a_{i+1}^{(r)}}{a_i^{(r)}}=\alpha_i^{2^r} \tag{10.30}$$

由式 (10.29) 可判知 r 是否已足夠大，若是即可由式 (10.30) 求得 $|\alpha_i|$。

(2) α_i 為實數根，但 α_{k+1} 有 ν 個根 $(\nu>1)$ 之絕對值相等，即 $|\alpha_1|>|\alpha_2|>\cdots>|\alpha_k|>|\alpha_{k+1}|=\cdots=|\alpha_{k+\nu}|>|\alpha_{k+\nu+1}|>\cdots>|\alpha_n|$。則

$$a_{i+1}^{(r)}=\alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_i^{2^r},\quad i=1,2,...,k \tag{10.31}$$

$$a_{k+l+1}^{(r)}=C_l^{\nu}\alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_k^{2^r}(\alpha_{k+1}^{2^r})^l,\quad l=1,2,...,\nu \tag{10.32}$$

$$a_{i+1}^{(r)}=\alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_k^{2^r}(\alpha_{k+1}^{2^r})^{\nu}\alpha_{k+\nu+1}^{2^r}\cdots\alpha_i^{2^r},\quad i=k+\nu+1,...,n \tag{10.33}$$

在式 (10.32) 中之 $C_l^{\nu}=\frac{\nu!}{l!(\nu-l)!}$，該值除 $l=\nu$ 者外並不等於 1。因此式 (10.29) 於 $i=k+1,...,k+\nu-1$ 時不再成立，而式 (10.30) 於 $i=k+1,...,k+\nu$ 時亦不成立。亦即 α_{k+1} 有 ν 個重根時有連續 $\nu-1$ 個關係式 (10.29) 不成立。而於得知 α_{k+1} 之重根數目後，$|\alpha_{k+1}|$ 之值須用下式求得

$$\frac{a_{k+\nu+1}^{(r)}}{a_{k+1}^{(r)}}=(\alpha_{k+1}^{2^r})^{\nu} \tag{10.34}$$

(3) α_{k+1} 為 α_{k+2} 之共軛複數根，設其絕對值為 ρ_{k+1}，即 $\alpha_{k+1}=\rho_{k+1}e^{i\theta}$，$\alpha_{k+2}=\rho_{k+1}e^{-i\theta}$，且 $|\alpha_1|>|\alpha_2|>\cdots>|\alpha_k|>\rho_{k+1}>|\alpha_{k+3}|>\cdots>|\alpha_n|$，則

$$a_{i+1}^{(r)} = \alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_i^{2^r}, \quad i = 1, 2, ..., k \tag{10.35}$$

$$a_{k+2}^{(r)} = \alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_k^{2^r}\rho_{k+1}^{2^r}(2\cos(2^r\theta)) \tag{10.36}$$

$$a_{k+3}^{(r)} = \alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_k^{2^r}(\rho_{k+1}^{2^r})^2 \tag{10.37}$$

$$a_{i+1}^{(r)} = \alpha_1^{2^r}\alpha_2^{2^r}\cdots\alpha_k^{2^r}(\rho_{k+1}^{2^r})^2\alpha_{k+3}^{2^r}\cdots\alpha_i^{2^r}, \quad i = k+3, ..., n \tag{10.38}$$

因此式(10.29)於$i = k+1$時不再成立，而式(10.30)於$i = k+1, k+2$時亦不成立。而ρ_{k+1}之值須用下式求得

$$\frac{a_{k+3}^{(r)}}{a_{k+1}^{(r)}} = (\rho_{k+1}^{2^r})^2 \tag{10.39}$$

一般而言，若有連續$\nu - 1$個i值之式(10.29)不成立，則可判知有ν個根之絕對值相等，其值可用式(10.34)求得。對於正或負實數根，可分別以其正負值代入原多項式驗証何者為多項式之根。至於複數根，則須用其他方法計算其幅角，有關方法將於下節介紹。

10.3 複數根之幅角之求法

前節介紹之Graeffe法可同時求得多項式之所有根之絕對值，本節則將僅介紹已知根之絕對值後，求根之幅角之方法。

10.3.1 *FFT*法

利用FFT可求得絕對值為ρ之複數根$\rho e^{i\theta}$之幅角θ。將$z = \rho e^{i\theta}$代入式(10.1)可得

$$P_n(\rho e^{i\theta}) = a_1\rho^n e^{i\theta n} + a_2\rho^{n-1}e^{i\theta(n-1)} + \cdots + a_n\rho e^{i\theta} + a_{n+1} \tag{10.40}$$

令$\theta = (2\pi/N)k$得

$$G_k = P_n(\rho e^{\frac{2\pi i}{N}k}) = a_1\rho^n(e^{\frac{2\pi i}{N}})^{kn} + a_2\rho^{n-1}(e^{\frac{2\pi i}{N}})^{k(n-1)} + \cdots + a_n\rho(e^{\frac{2\pi i}{N}})^k + a_{n+1} \tag{10.41}$$

再令

$$g_l = a_{n+1-l}\rho^l, \quad l = 0, 1, ..., n \tag{10.42}$$

$$g_l = 0, \quad l = n+1, ..., N-1 \tag{10.43}$$

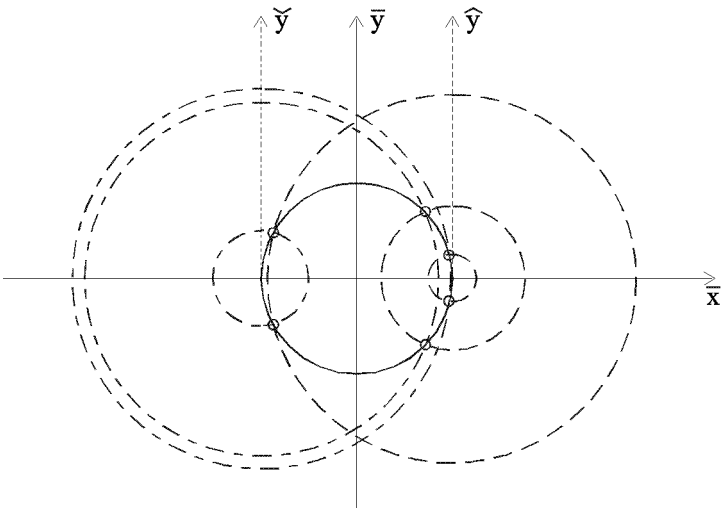
可得

$$G_k = \sum_{l=0}^{N-1} g_l (e^{\frac{2\pi i}{N}})^{kl}, \quad k = 0, 1, ..., N-1 \tag{10.44}$$

上式可用FFT法快速求得N個G_k之值。再由較小之$|G_k|$值可得對應之$\theta = \frac{2\pi}{N}k$，最後再用Bairstow法（詳第10.4節）求更精確之根。利用FFT法之好處為可同時快速算出許多（例如N=1024）θ值之函數值。

10.3.2 圓交點法

Graeffe法之優點為可同時求出所有根之絕對值，缺點為複數根之幅角無法直接求得。但若將多項式之根做平移（或根之位置不變而將座標原點平移），再用Graeffe法求平移後之根之絕對值，則因平移前根之絕對值ρ為根之位置$C(x,y)$至原座標原點O之距離$\overline{OC}$，而平移後根之絕對值$\hat{\rho}$為根之位置$C(\hat{x},\hat{y})$至新座標原點$\hat{O}$之距離$\overline{\hat{O}C}$，若以O為圓心，ρ為半徑畫圓，再以$\hat{O}$為圓心，$\hat{\rho}$為半徑畫圓，則根之位置C必在二圓之交點上。注意C點之座標相對於原座標軸為(x,y)；相對於新座標軸為$(\hat{x},\hat{y})$。因此僅利用Graeffe法所求出根之絕對值即可求出所有實數根及複數根。但因平移之操作常引進相當大之誤差，故須針對絕對值大小不同之根分別做平移。以下為實際之作法（詳圖一）



圖一 根在同心圓之交點

1. 令$\bar{z}=\rho^{-1}z$得$\bar{P}_n(\bar{z})=P_n(\rho\bar{z})$。

2. 令$\hat{z}=\bar{z}-1$得$\hat{P}_n(\hat{z})=\bar{P}_n(\hat{z}+1)$。

3. 以Graeffe法求$\hat{P}_n(\hat{z})=0$之根之絕對值為$\hat{\rho}_i$。

4. 以$\hat{z}$之原點$\hat{z}=(0,0)$即$\bar{z}=(1,0)$為圓心，以$\hat{\rho}_i$為半徑畫同心圓。這些同心圓與圓心為$\bar{z}=(0,0)$半徑為1之單位圓之交點即可能為複數根：$\bar{\theta}=2\sin^{-1}(\hat{\rho}_i/2),\ \bar{x}=1-\hat{\rho}_i^2/2,\ \bar{y}=\hat{\rho}_i\sqrt{1-\hat{\rho}_i^2/4}$。可直接代入$\bar{P}_n(\bar{z})$驗証是否為0，如是可再用Bairstow法（詳第10.4節）求更精確之根。由於較大之$\hat{\rho}_i$所得交點位置之精度較差，因此步驟2-4只求單位圓上最靠近$\bar{z}=(1,0)$之三分之一圓周上之根（即$\hat{\rho}_i\le 1$者）。

5. 令$\check{z}=\bar{z}+1$得$\check{P}_n(\check{z})=\bar{P}_n(\check{z}-1)$。以類似步驟3-4之做法求單位圓上最靠近$\bar{z}=(-1,0)$之三分之一圓周上之根。

6. 單位圓上最靠近$\bar{z}=(0,\pm 1)$之上下各六分之一圓周上之根如經一次之根平方變號之操作，這些根均會轉移到最靠近$\bar{z}=(1,0)$之三分之一圓周上。再利用步驟2-4即可求得這些根。

7. 單位圓上最靠近$\bar{z}=(0,\pm 1)$之上下各六分之一圓周上之根，雖亦可令$\tilde{z}=\bar{z}\mp i$做平移，但所得$\tilde{P}_n(\tilde{z})$之係數含複數值，須增加次數為二倍以使係數變為實數（詳第10.6節），則所需儲存空間及運算時間均較多，故不採用。

10.3.3 *Bareiss* 法

另一個求複數根之方法為利用下式（設複數根之絕對值已調為1）

$$\begin{aligned}\bar{P}(\bar{z}) &= \bar{a}_1\bar{z}^n+\bar{a}_2\bar{z}^{n-1}+\cdots+\bar{a}_n\bar{z}+\bar{a}_{n+1} \qquad (10.45)\\ &= (\bar{z}^2-p\bar{z}+1)(b_1\bar{z}^{n-2}+b_2\bar{z}^{n-3}+\cdots+b_{n-1})+(\bar{z}-p)b_n+b_{n+1}\end{aligned}$$

可將b_i以已知之$\bar{a}_i$及未知之p表示，因此可得b_n及b_{n+1}分別為p之$n-1$次及n次多項式函數。解$b_n(p)=0$或$b_{n+1}(p)=0$之實數根即可求得p。因p為實數故仍可利用Graeffe法求得。但此法精度在p較小時反求根之幅角較準；在p較大時反求根之幅角較不準，但均遠較圓交點法為差，故不採用，有興趣者可參閱文獻1及2。以下僅將求$b_n(p)$之係數BN(I)及$b_{n+1}(p)$之係數BM(I)之程式片斷列供參考：

```
C**   INPUT  : M=N+1, (A(I), I=1,M)
C**   OUTPUT : (BN(I), I=1,N), (BM(I), I=1,M)
      BN(1)=A(1)
      BM(1)=A(1)
      BM(2)=A(2)
      DO 10 K=3,M
      BM(K)=A(K)
      DO 10 I=K,3,-1
      BN(I-1)=BM(I-1)
   10 BM(I)=BM(I)-BN(I-2)
```

以上三個方法須先已知ρ，但ρ由Graeffe法求得而難免有些許誤差，此誤差對以上三法有不同程度之影響，特簡述於下

1. FFT法ρ不準時仍可由$|G_k|$之最小值找到近似幅角，方法可靠。

2. 圓交點法ρ不準時仍可得$\hat{\rho}$，方法最可靠。

3. Bareiss法ρ不準時可能找不到實數根之p，故較不可靠。

10.4 貝爾斯脫法

貝爾斯脫(Bairstow)法係先求出實數係數多項式$P_n(z)$之二次因式，再由該二次因式求$P_n(z)=0$之二個實數根或一對共軛複數根。此法可以求得非常精確之根，但須有近似之初始值，否則不一定會收斂。因此頗適合於配合前述各法之後續微調之用。

以一任意之二次式(z^2-pz+q)除多項式$P_n(z)$可得商式為$Q_{n-2}(z)$及餘式$Rz+S$。即

$$P_n(z)=(z^2-pz+q)Q_{n-2}(z)+Rz+S \tag{10.46}$$

此項運算可利用如下表之綜合除法

$$\begin{array}{cccccccc}
 & a_1 & a_2 & a_3 & \cdots & a_{n-1} & a_n & a_{n+1} \\
 & & pb_1 & pb_2 & \cdots & pb_{n-2} & pb_{n-1} & pb_n \\
+ & & & -qb_1 & \cdots & -qb_{n-3} & -qb_{n-2} & -qb_{n-1} \\
\hline
 & b_1 & b_2 & b_3 & \cdots & b_{n-1} & b_n & b_{n+1}
\end{array}$$

令$b_{-1}=b_0=0$，則表中

$$b_i=a_i+pb_{i-1}-qb_{i-2},\qquad i=1,2,...,n+1 \tag{10.47}$$

因此得

$$\begin{aligned} Q_{n-2}(z) &= b_1z^{n-2}+b_2z^{n-3}+\cdots+b_{n-1} && (10.48)\\ Rz+S &= b_nz+(b_{n+1}-pb_n)=(z-p)b_n+b_{n+1} && (10.49)\end{aligned}$$

對應之程式片斷為：

```
C**   INPUT  : M=N+1, A(I), P, Q
C**   OUTPUT : A(I)
C**   INPUT  A(I): (Sum of A(I)*X**(M-I)) =
C**   OUTPUT A(I): (X*X-P*X+Q)(Sum of A(I)*X**(M-2-I)) +
C**                (X-P)*A(M-1)+A(M)
      A(2)=A(2)+P*A(1)
      DO 10 I=3,M
   10 A(I)=A(I)+P*A(I-1)-Q*A(I-2)
```

則(z^2-pz+q)為$P_n(z)$之因式之條件為$S=0$及$R=0$，亦可為$b_{n+1}=0$及$b_n=0$。因b_{n+1}及b_n（及$Q_{n-2}(z)$之係數b_i）將隨p與q改變，可視其為p與q之函數，則(z^2-pz+q)為$P_n(z)$之因式之條件即為

$$b_{n+1}(p,q)=0 \tag{10.50}$$

$$b_n(p,q)=0 \tag{10.51}$$

上式為二元聯立非線性方程式，可用牛頓法由假設之初值$p^{(0)}$與$q^{(0)}$按下式依$i=0,1,2,...$反覆求解。

$$\begin{aligned} p^{(i+1)} &= p^{(i)}+dp^{(i)} && (10.52)\\ q^{(i+1)} &= q^{(i)}+dq^{(i)} && (10.53)\end{aligned}$$

其中之$dp^{(i)}$與$dq^{(i)}$由下列聯立方程式解之

$$\begin{aligned} \frac{\partial b_{n+1}}{\partial p}\,dp^{(i)}+\frac{\partial b_{n+1}}{\partial q}\,dq^{(i)} &= -b_{n+1} && (10.54)\\ \frac{\partial b_n}{\partial p}\,dp^{(i)}+\frac{\partial b_n}{\partial q}\,dq^{(i)} &= -b_n && (10.55)\end{aligned}$$

以下說明$\frac{\partial b_{n+1}}{\partial p}$, $\frac{\partial b_{n+1}}{\partial q}$, $\frac{\partial b_n}{\partial p}$, $\frac{\partial b_n}{\partial q}$之求法。由式(10.46)至(10.49)所得之

$$P_n(z) = (z^2 - pz + q)Q_{n-2}(z) + (z-p)b_n + b_{n+1} \tag{10.56}$$

分別對p, q偏微分，注意$P_n(z)$為固定之多項式與p, q無關，故$\frac{\partial P}{\partial p} = \frac{\partial P}{\partial q} = 0$

$$\begin{aligned} 0 &= -zQ_{n-2}(z) - b_n \\ &\quad +(z^2 - pz + q)\frac{\partial Q_{n-2}(z)}{\partial p} + (z-p)\frac{\partial b_n}{\partial p} + \frac{\partial b_{n+1}}{\partial p} \qquad (10.57) \\ 0 &= Q_{n-2}(z) + (z^2 - pz + q)\frac{\partial Q_{n-2}(z)}{\partial q} + (z-p)\frac{\partial b_n}{\partial q} + \frac{\partial b_{n+1}}{\partial q} \qquad (10.58) \end{aligned}$$

或寫成

$$\begin{aligned} zQ_{n-2}(z) + b_n &= (z^2 - pz + q)\frac{\partial Q_{n-2}(z)}{\partial p} + (z-p)\frac{\partial b_n}{\partial p} + \frac{\partial b_{n+1}}{\partial p} \qquad (10.59) \\ -Q_{n-2}(z) &= (z^2 - pz + q)\frac{\partial Q_{n-2}(z)}{\partial q} + (z-p)\frac{\partial b_n}{\partial q} + \frac{\partial b_{n+1}}{\partial q} \qquad (10.60) \end{aligned}$$

上二式之型式與式(10.56)相當，故亦可利用綜合除法，令$c_{-1} = c_0 = 0$計算

$$c_i = b_i + pc_{i-1} - qc_{i-2}, \qquad i = 1, 2, ..., n \tag{10.61}$$

比照式(10.48)及(10.49)可得

$$\begin{aligned} \frac{\partial Q_{n-2}(z)}{\partial p} &= c_1 z^{n-3} + c_2 z^{n-4} + \cdots + c_{n-2} \qquad (10.62) \\ \frac{\partial b_n}{\partial p} &= c_{n-1} \qquad (10.63) \\ \frac{\partial b_{n+1}}{\partial p} &= c_n \qquad (10.64) \end{aligned}$$

及

$$\begin{aligned} -\frac{\partial Q_{n-2}(z)}{\partial q} &= c_1 z^{n-4} + c_2 z^{n-5} + \cdots + c_{n-3} \qquad (10.65) \\ -\frac{\partial b_n}{\partial q} &= c_{n-2} \qquad (10.66) \\ -\frac{\partial b_{n+1}}{\partial q} &= c_{n-1} \qquad (10.67) \end{aligned}$$

將式(10.63)、(10.64)、(10.66)與(10.67)代入式(10.54)與(10.55)中可得

$$\begin{aligned} c_n\, dp^{(i)} - c_{n-1}\, dq^{(i)} &= -b_{n+1} \qquad (10.68) \\ c_{n-1}\, dp^{(i)} - c_{n-2}\, dq^{(i)} &= -b_n \qquad (10.69) \end{aligned}$$

或

$$dp^{(i)} = (b_n c_{n-1} - b_{n+1} c_{n-2})/(c_n c_{n-2} - c_{n-1}^2) \tag{10.70}$$

$$dq^{(i)} = (b_n c_n - b_{n+1} c_{n-1})/(c_n c_{n-2} - c_{n-1}^2) \tag{10.71}$$

以下為Bairstow法所需之二回合綜合除法運算表（其中p, q為$p^{(k)}, q^{(k)}$省略上標）

	a_1	a_2	a_3	$\cdots$	a_{n-2}	a_{n-1}	a_n	a_{n+1}
		pb_1	pb_2	$\cdots$	pb_{n-3}	pb_{n-2}	pb_{n-1}	pb_n
$+$			$-qb_1$	$\cdots$	$-qb_{n-4}$	$-qb_{n-3}$	$-qb_{n-2}$	$-qb_{n-1}$
	b_1	b_2	b_3	$\cdots$	b_{n-2}	b_{n-1}	b_n	b_{n+1}
		pc_1	pc_2	$\cdots$	pc_{n-3}	pc_{n-2}	pc_{n-1}	
$+$			$-qc_1$	$\cdots$	$-qc_{n-4}$	$-qc_{n-3}$	$-qc_{n-2}$	
	c_1	c_2	c_3	$\cdots$	c_{n-2}	c_{n-1}	c_n	

Bairstow法之程式簡單（詳副程式BAIRST）且於正確根附近收斂很快為二次逼近。但若初始值$p^{(0)}$與$q^{(0)}$不夠接近正確根時，收斂較慢且有時不一定會收斂至附近之正確根。 因此Bairstow法常須配合前述Graeffe法及FFT法或圓交點法計算較精確之根。

10.5 牛頓法

牛頓法不但可求一般非線性方程式之根，用以求多項式方程式之根亦頗簡單。如初始值為實數$x^{(0)}$，反覆利用下列運算式，通常可收斂至正確之實數根

$$x^{(k+1)} = x^{(k)} - \frac{P_n(x^{(k)})}{P_n'(x^{(k)})} \tag{10.72}$$

其中$P_n(x^{(k)})$及$P_n'(x^{(k)})$可先用二回合綜合除法求

$$b_i = x^{(k)} b_{i-1} + a_i, \qquad i = 1, 2, ..., n+1 \tag{10.73}$$

$$c_i = x^{(k)} c_{i-1} + b_i, \qquad i = 1, 2, ..., n \tag{10.74}$$

即可得$P_n(x^{(k)}) = b_{n+1}$及$P_n'(x^{(k)}) = c_n$。上列運算相當於以$s = x^{(k)}$代入式(10.8)及(10.9)中之結果。通常b_i及c_i可不必儲存，而可用下列程式片斷，其計算所得之B及C即分別為$P_n(x)$及$P_n'(x)$。

```
C**   INPUT  : M=N+1, A(I), X
C**   OUTPUT : B=Pn(X),  C=Pn'(X)
      C=0
      B=A(1)
      DO 10 I=2,M
      C=X*C+B
   10 B=X*B+A(I)
```

如初始值為複數 $z^{(0)} = x^{(0)} + iy^{(0)}$ ，則用牛頓法亦可求得複數根，但須用複數計算下列運算式

$$x^{(k+1)} + iy^{(k+1)} = x^{(k)} + iy^{(k)} - \frac{P_n(x^{(k)} + iy^{(k)})}{P_n'(x^{(k)} + iy^{(k)})} \tag{10.75}$$

上式雖僅是一個複數運算式，實際上是二個實數變數 (x, y) 之二個聯立實數運算式。

令 $P_n(z) = P_n(x + iy)$ 之實數部分為 $R(x, y)$ ，虛數部分為 $I(x, y)$ ，即 $P_n(z) = P_n(x + iy) = R(x, y) + iI(x, y)$ 。令 $R_x = \partial R/\partial x$ ， $R_y = \partial R/\partial y$ ， $I_x = \partial I/\partial x$ ， $I_y = \partial I/\partial y$ ，由 Cauchy 等式知 $I_y = R_x$ ， $R_y = -I_x$ ，因此 $P_n'(z) = R_x + iI_x = I_y - iR_y$ ，故

$$\frac{P_n(z)}{P_n'(z)} = \frac{R + iI}{R_x + iI_x} = \frac{(R_x - iI_x)(R + iI)}{R_x^2 + I_x^2} \tag{10.76}$$

$$= \frac{(R_xR + I_xI) + i(R_xI - I_xR)}{R_x^2 + I_x^2} \tag{10.77}$$

$$= \frac{(I_yR - R_yI) + i(R_xI - I_xR)}{R_xI_y - I_xR_y} \tag{10.78}$$

上式代入式 (10.75) 後實數部分與虛數部分分別計算如下

$$x^{(k+1)} = x^{(k)} - \frac{I_y^{(k)}R^{(k)} - R_y^{(k)}I^{(k)}}{R_x^{(k)}I_y^{(k)} - I_x^{(k)}R_y^{(k)}} \tag{10.79}$$

$$y^{(k+1)} = y^{(k)} - \frac{R_x^{(k)}I^{(k)} - I_x^{(k)}R^{(k)}}{R_x^{(k)}I_y^{(k)} - I_x^{(k)}R_y^{(k)}} \tag{10.80}$$

上式與牛頓法求解 $R(x, y) = 0$ ， $I(x, y) = 0$ 之 x, y 之下列反覆計算式相同

$$\begin{Bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{Bmatrix} = \begin{Bmatrix} x^{(k)} \\ y^{(k)} \end{Bmatrix} - \begin{bmatrix} R_x^{(k)} & R_y^{(k)} \\ I_x^{(k)} & I_y^{(k)} \end{bmatrix}^{-1} \begin{Bmatrix} R^{(k)} \\ I^{(k)} \end{Bmatrix} \tag{10.81}$$

10.6 複數係數多項式

前面各節所介紹之求根法僅適用於實數係數多項式。對於複數係數多項式，若將次數增為二倍，可得一實數係數多項式。方法介紹如下：

設$P_n(z)=a_1z^n+a_2z^{n-1}+\cdots+a_{n+1}=0$之根為$\alpha_1,\alpha_2,...,\alpha_n$。令$a_i$之共軛複數為$\bar{a}_i$，$\alpha_i$之共軛複數為$\bar{\alpha}_i$，則$\bar{P}_n(\bar{z})=\bar{a}_1\bar{z}^n+\bar{a}_2\bar{z}^{n-1}+\cdots+\bar{a}_{n+1}=0$之根為$\bar{\alpha}_1,\bar{\alpha}_2,...,\bar{\alpha}_n$。因此若令

$$G_{2n}(z)=P_n(z)\bar{P}_n(z)=b_1z^{2n}+b_2z^{2n-1}+\cdots+b_{2n+1} \tag{10.82}$$

式中

$$b_{2i-1} = 2\sum_{k=1}^{min(i-1,n-i+1)} Real(a_{i-k}\bar{a}_{i+k})+a_i\bar{a}_i,\ i=1,2,...,n+1 \tag{10.83}$$

$$b_{2i} = 2\sum_{k=0}^{min(i-1,n-i)} Real(a_{i-k}\bar{a}_{i+k+1}),\qquad i=1,2,...,n \tag{10.84}$$

則$G_{2n}(z)=0$之根如為實數根必有偶數個，如為複數根必為共軛複數對。$G_{2n}(z)$多項式之係數亦必為實數。故可按實數係數多項式之求根法求解。二個共軛複數中應取何者為原多項式之根，可直接代入原多項式$P_n(z)$驗証是否為0而得知。求b_i之程式片斷為：

```
C**   INPUT  : M2=2*(N+1), (A(I), I=1,M2)
C**   OUTPUT : MM=2*N+1, (B(I), I=1,MM)
      MM=M2-1
      DO 50 I=1,MM,2
      IX=I
      IY=I
      SR=0.0
      SI=0.0
   30 IF(IX.LE.0.OR.IY.GE.MM) GO TO 40
      IY=IY+2
      SI=SI+A(IX)*A(IY)+A(IX+1)*A(IY+1)
      IF(IX.LE.1) GOTO 40
      IX=IX-2
      SR=SR+A(IX)*A(IY)+A(IX+1)*A(IY+1)
      GOTO 30
   40 B(I)=A(I)*A(I)+A(I+1)*A(I+1)+2.0*SR
   50 B(I+1)=2.0*SI
```

10.7 求多項式根之程式

****

```
      PROGRAM POLY
C**   ================================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION A(5100)
      DATA EPSX,EPSF/1.0D-15,1.0D-5/
      DATA ITM,ITS,IP,NF/15,4,21,0/,NDM/5100/
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Input coefficients in A(1:M) --> A(I) * X**(M-I)              |    **
C**   | M     = Number of real coefficients = Poly. order + 1        |    **
C**   |       = 2 * Number of complex coefficients                   |    **
C**   | ITM   = Maximum numbers of iterations allowed                |    **
C**   | ITS   = Max. # of iterations for no additional # of RR=1     |    **
C**   | IP    = Debug output control (1+2+4+8+16+32+64+128)          |    **
C**   | NF    > 0 : By FFT method (NF=data points used in FFT)       |    **
C**   |       = 0 : By root shift (Itersections of circles)          |    **
C**   |       < 0 : For complex coefficients                         |    **
C**   +--------------------------------------------------------------+    **
   10 CALL INPEQU(A,M,NF,EPSX,EPSF,ITM,ITS,IP)
      NO=1
      NX=NO+M                ! A(NO) for AO(M)
      NY=NX+M                ! A(NX) for AX(M)
      NR=NY+M                ! A(NY) for AY(M)
C**   +--------------------------------------------------------------+    **
C**   | By FFT method                                                |    **
C**   +--------------------------------------------------------------+    **
      IF(NF.LE.0) GOTO 30
      NM=NR+M*4+NF+2         ! A(NR) for Working array AR(4*M+NF+2)
      IF(NM.GT.NDM) STOP
      CALL POLYFT(A(NO),A(NX),A(NY),A(NR),M,EPSX,EPSF,ITM,ITS,IP,NF)
      GOTO 10
C**   +--------------------------------------------------------------+    **
C**   | By root shift (intersections of circles)                     |    **
C**   +--------------------------------------------------------------+    **
   30 IF(NF.LT.0) GOTO 50
      NM=NR+M*13             ! A(NR) for Working array AR(13*M)
      IF(NM.GT.NDM) STOP
      CALL POLYRT(A(NO),A(NX),A(NY),A(NR),M,EPSX,EPSF,ITM,ITS,IP)
      GOTO 10
C**   +--------------------------------------------------------------+    **
C**   | For complex coefficients                                     |    **
C**   +--------------------------------------------------------------+    **
   50 NM=NR+M*14             ! A(NR) for Working array AR(14*M)
      IF(NM.GT.NDM) STOP
      CALL POLYCT(A(NO),A(NX),A(NY),A(NR),M,EPSX,EPSF,ITM,ITS,IP)
      GOTO 10
      END
**********************************************************************
      SUBROUTINE INPEQU(AO,M,NF,EPSX,EPSF,ITM,ITS,IP)
C**   ================================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      LOGICAL ANSWER
      DIMENSION AO(1)
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Input data from terminal                                     |    **
C**   +--------------------------------------------------------------+    **
   10 CALL TERMR(' EPSX  :',EPSX)
      CALL TERMR(' EPSF  :',EPSF)
      CALL TERMI(' ITM   :',ITM)
```

```
      CALL TERMI(' ITS   :',ITS)
      CALL TERMI(' IDEBUG:',IP)
      CALL TERMI(' NF-FFT:',NF)
      CALL TERMI(' M     :',M)
C**   +--------------------------------------------------------+    **
C**   | Input coefficients of the polynomial                   |    **
C**   +--------------------------------------------------------+    **
      I=0
   20 I=I+1
      CALL TERMI(' I     :',I)
      IF(I.GT.M) GOTO 30
      CALL TERMR(' AO(I) :',AO(I))
      GOTO 20
C**   ------------------------------------------------------------    **
   30 IF(ANSWER(' WANT TO RETYPE (Y/N) ? ','?')) GOTO 10
      IF(M.LE.0) STOP
      RETURN
      END
*************************************************************************
      SUBROUTINE GRAEFF(AA,KA,BB,KB,RR,KR,AO,MO,EPS,ITM,ITS,IP)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AA(MO),KA(MO),BB(MO),KB(MO),RR(MO),KR(MO),AO(MO)
      DATA BASE/1024.0D0/,KPOS,KNEG/7,-8/,KMAX/1073741822/
C**   ============================================================    **
C**   BASE**( KPOS-1) .GT. 2**BitR8      (10*(7-1).GT.56)             **
C**   BASE**(-KNEG-2) .GT. 2**BitR8      (10*(8-2).GT.56)             **
C**   KMAX*2+3 .LE. 2**MIN0(BitI4,2*BitI4)-1    (2**30-2)            **
C**   ------------------------------------------------------------    **
C**   Input   : AO(MO),MO,EPS,ITM,ITS,IP                            **
C**   Output  : RR(MO),KR(MO)                                       **
C**   Working : AA(MO),KA(MO),BB(MO),KB(MO)                         **
C**   ------------------------------------------------------------    **
C*I   AO(I) = Coefficient of X**(M-I)                               **
C*I   MO    = Number of coefficients = Polynormial order + 1        **
C*I   EPS   = Error tolerance for RR=1 as |RR-1| <= EPS             **
C*I   ITM   = Maximum numbers of iterations allowed                 **
C*I   ITS   = Max. # of iterations for no additional # of RR=1      **
C*I   IP    = Debug output control (1+2+4+8+16+32+64+128)          **
C**   ------------------------------------------------------------    **
C*O   If KR(I).GT.0 then RR(I) is the magnitude of the roots       **
C**                      KR(I) is the multiplicity                 **
C**   ------------------------------------------------------------    **
C**   By Graeffe's root squaring method                             **
C**   ============================================================    **
C**   +------------------------------------------------------------+    **
C**   | Adjust M such that A(M).NE.0 and AO(M+1:MO)=0             |    **
C**   +------------------------------------------------------------+    **
      M=MO
   10 IF(M.LE.1) GOTO 90
      IF(DABS(AO(M)).LT.1000.0D0*EPS) THEN
        M=M-1
        GOTO 10
      ENDIF
C**   +------------------------------------------------------------+    **
C**   | Normalize : AO(1)=|AO(M)|=1, AO(II)=AA(II)*BASE**KA(II)  |    **
C**   +------------------------------------------------------------+    **
   15 AI=AO(1)
      AM=DEXP(DLOG(DABS(AO(M)/AI))/(M-1))
      DO 20 II=1,M
```

```
      AA(II)=AO(II)/AI
      KA(II)=0
      CALL NORMAL(AA(II),KA(II))
      AK=KA(II)
      IF(IP/4*2.NE.IP/2)
     *WRITE(2,'(I8,1PE16.8,0PF20.0,1PE24.16)') II,AA(II),AK,AO(II)
   20 AI=AI*AM
      AA(M)=AA(M)/DABS(AA(M))
      KA(M)=0
      IF(IP/4*2.NE.IP/2) WRITE(2,'(2I4)') ITM,ITS
C**   +--------------------------------------------------------+    **
C**   | Root squaring                                          |    **
C**   +--------------------------------------------------------+    **
      KM=0
      I1O=0
      DO 50 IT=1,ITM+ITS*2
      DO 30 II=1,M
      IL=II
      IR=II
      S=2.0D0
      BB(II)=AA(II)*AA(II)
      KB(II)=KA(II)+KA(II)
      CALL NORMAL(BB(II),KB(II))
   25 IF(IL.LE.1.OR.IR.GE.M) GO TO 30
      IL=IL-1
      IR=IR+1
      S=-S
      SA=S*AA(IL)*AA(IR)
      IF(SA.NE.0.0D0) THEN
        KI=KA(IL)+KA(IR)-KB(II)
        IF(KI.GE.KPOS.OR.BB(II).EQ.0.0D0) THEN
          BB(II)=SA
          KB(II)=KA(IL)+KA(IR)
          CALL NORMAL(BB(II),KB(II))
        ELSE IF(KI.GT.KNEG) THEN
          BB(II)=BB(II)+SA*BASE**KI
          CALL NORMAL(BB(II),KB(II))
        ENDIF
      ENDIF
      GOTO 25
   30 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Move BB(M),KB(M) --> AA(M),KA(M)                       |    **
C**   | Count I1N = Number of II with AA(II)**2/BB(II)=1       |    **
C**   +--------------------------------------------------------+    **
      I1N=0
      DO 40 II=1,M
      IF(BB(II).NE.0.0D0.AND.AA(II).NE.0.0D0) THEN
        RR(II)=AA(II)*AA(II)/BB(II)*BASE**(KA(II)+KA(II)-KB(II))
      ELSE
        RR(II)=0.0D0
      ENDIF
      AA(II)=BB(II)
      KA(II)=KB(II)
      IF(DABS(RR(II)-1.0D0).LE.EPS) I1N=I1N+1
      IF(KA(II).GT.KM) KM=KA(II)
      IF(IP/4*2.NE.IP/2)
     *WRITE(2,'(I8,1PE16.8,I8,E24.16)') II,AA(II),KA(II),RR(II)
   40 CONTINUE
C**   +--------------------------------------------------------+    **
```

```
C**   | Set  MS  = Number of squaring has been performed         |    **
C**   | I1N/I1O  = Number of RR(*)=1 at this/last cycle          |    **
C**   | IT-ITO+1 = The number of cycles which have the same I1N  |    **
C**   +----------------------------------------------------------+    **
      IF(I1N.NE.I1O) ITO=IT
      I1O=I1N
      MS=IT
      IF(IP/4*2.NE.IP/2) WRITE(2,'(4I4)') IT,I1N,I1O,ITO
C**   +----------------------------------------------------------+    **
C**   | Convergence test                                         |    **
C**   +----------------------------------------------------------+    **
      IF(KM.GT.KMAX) GOTO 60
      IF(IT.GE.ITM.AND.IT-ITO.GE.ITS) GOTO 60
   50 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find MM = II-IL = Multiplicity of root S, where RR(II)=1 |    **
C**   +----------------------------------------------------------+    **
   60 IL=1
      DO 80 II=2,M
      IF(DABS(RR(II)-1.0D0).GT.EPS) GOTO 80
      S=AA(II)/AA(IL)
      KK=KA(II)-KA(IL)
      CALL RSQRT(S,KK,MS)
      MM=II-IL
      MQ=KK/MM
      MR=KK-MQ*MM
      IF(MM.GT.1) S=DEXP(DLOG(S*BASE**MR)/MM)
      S=S*BASE**MQ*AM
C**   +----------------------------------------------------------+    **
C**   | Set RR(IL:IL+MM-1)=S, KR(IL)=MM, KR(IL+1:IL+MM-1)=-MM    |    **
C**   +----------------------------------------------------------+    **
      MX=MM
      DO 70 IR=1,MM
      RR(IL)=S
      KR(IL)=MX
      IF(IP/2*2.NE.IP)
     *WRITE(2,'(I8,1PE24.16,I8,E24.16)') IL,RR(IL),KR(IL),RR(IL+1)
      MX=-MM
      IL=IL+1
   70 CONTINUE
   80 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Return RR(M),KR(M) (RR(M+1:MO)=0, KR(M+1)=MO-M, if M<MO) |    **
C**   +----------------------------------------------------------+    **
   90 KR(MO)=-MO
      IF(M.EQ.MO) RETURN
      DO 95 II=M,MO-1
      RR(II)=0.0D0
   95 KR(II)=M-MO
      KR(M)=MO-M
      RETURN
      END
************************************************************************
      SUBROUTINE NORMAL(A,K)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DATA BASE/1024.0D0/
C**   ============================================================    **
C**   Normalize : A*BASE**K such that BASE > |A| >= 1                 **
C**   ============================================================    **
```

```
      X=DABS(A)
      IF(X.LE.0.0D0) THEN
        K=0
        RETURN
      ENDIF
   10 IF(X.GE.BASE) THEN
        X=X/BASE
        K=K+1
        GO TO 10
      ENDIF
   20 IF(X.LT.1.0D0) THEN
        X=X*BASE
        K=K-1
        GOTO 20
      ENDIF
      A=DSIGN(X,A)
      RETURN
      END
*****************************************************************************
      SUBROUTINE RSQRT(A,K,MM)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DATA BASE/1024.0D0/
C**   ============================================================    **
C**   Square root of A*BASE**K  MM times  (A is positive)             **
C**   ============================================================    **
      DO 10 I=1,MM
      IF(K/2*2.NE.K) THEN
        A=DSQRT(A*BASE)
        K=(K-1)/2
      ELSE
        A=DSQRT(A)
        K=K/2
      ENDIF
   10 CONTINUE
      RETURN
      END
*****************************************************************************
      SUBROUTINE POLYRT(AO,RR,KR,A,M,EPSX,EPSF,ITM,ITS,IP)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AO(M),RR(M),KR(M),A(M,13)
C**   ============================================================    **
      CALL POLYNM(AO,RR,KR,A(1,1),A(1,2),A(1,3),A(1,4),A(1,5),A(1,6)
     *           ,A(1,7),A(1,8),A(1,9),A(1,10),A(1,11),A(1,12),A(1,13)
     *           ,M,EPSX,EPSF,ITM,ITS,IP)
      RETURN
      END
*****************************************************************************
      SUBROUTINE POLYNM(AO,RR,KR,AA,KA,BB,KB,AR,AL,AC,PR,PL,PC,NR,NL,NC
     *                 ,M,EPSX,EPSF,ITM,ITS,IP)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AO(M),RR(M),KR(M),AA(M),KA(M),BB(M),KB(M),BUF(14)
      DIMENSION AR(M),AL(M),AC(M),PR(M),PL(M),PC(M),NR(M),NL(M),NC(M)
      DATA BASE/1024.0D0/,ZERO/0.0D0/,IM/2/
C**   ============================================================    **
C**   Input   : AO(M),M,EPSX,EPSF,ITM,ITS,IP                          **
C**   Output  : RR(M),KR(M)                                           **
C**   Working : AA(M),KA(M),BB(M),KB(M)                               **
```

```
C**             AR(M),AL(M),AC(M),PR(M),PL(M),PC(M),NR(M),NL(M),NC(M) **
C**    ------------------------------------------------------------    **
C**    AO(I) = Coefficient of X**(M-I)                                 **
C**    ------------------------------------------------------------    **
C**    If KR(I).GT.0 and KR(I+1).LT.0 then                             **
C**         RR(I) = Real root                                          **
C**         KR(I) = Its multiplicity                                   **
C**    ------------------------------------------------------------    **
C**    If KR(I).GT.0 and KR(I+1).EQ.0 then                             **
C**         (RR(I),+-RR(I+1)) = Complex roots                          **
C**         KR(I) = Its multiplicity (always even)                     **
C**    ------------------------------------------------------------    **
C**    By Graeffe's method and root shift and Bairstow's method        **
C**    ============================================================    **
       CALL GRAEFF(AA,KA,BB,KB,RR,KR,AO,M,EPSX,ITM,ITS,IP)
C**    ------------------------------------------------------------    **
       II=1
       DO 500 IL=1,M-1
       IF(KR(IL).LE.0.OR.IL.LT.II) GO TO 500
       IR=IL+KR(IL)-1
C**    ------------------------------------------------------------    **
       II=IL
       X=RR(II)
       IF(X.EQ.0.0D0) GOTO 500
C**    +-----------------------------------------------------------+    **
C**    | Scale the II-th root to unit magnitude                    |    **
C**    +-----------------------------------------------------------+    **
       KFM=0
       XX=1.0D0
       KX=0
       DO 20 I=M,1,-1
       AA(I)=AO(I)*XX
       KA(I)=KX
       CALL NORMAL(AA(I),KA(I))
       IF(KA(I).GT.KFM.AND.AA(I).NE.0.0D0) KFM=KA(I)
       XX=XX*X
    20 CALL NORMAL(XX,KX)
C**    +-----------------------------------------------------------+    **
C**    | FFM = Sum of the abs of the coefficients                  |    **
C**    +-----------------------------------------------------------+    **
       FFM=0.0D0
       DO 30 I=1,M
       KI=KA(I)-KFM
       IF(KI.NE.0.AND.AA(I).NE.0.0D0) AA(I)=AA(I)*BASE**KI
    30 FFM=FFM+DABS(AA(I))
       EFM=EPSF*FFM
       EPS=EPSX*100.0D0
C**    +-----------------------------------------------------------+    **
C**    | Square the roots for pure imag roots                      |    **
C**    +-----------------------------------------------------------+    **
       DO 210 I=1,M
       AL(I)=AA(I)
       AR(I)=AA(I)
       IX=I
       IY=I
       S=2.0D0
       SA=0.0D0
   205 IF(IX.GT.1.AND.IY.LT.M) THEN
       IX=IX-1
       IY=IY+1
```

```
      S=-S
      SA=AA(IX)*AA(IY)-SA
      GOTO 205
      ENDIF
  210 AC(I)=AA(I)*AA(I)+S*SA
C**   +--------------------------------------------------------+    **
C**   | Shift the roots                                        |    **
C**   +--------------------------------------------------------+    **
      DO 220 K=M,2,-1
      DO 220 I=2,K
      AR(I)=AR(I)+AR(I-1)
      AL(I)=AL(I)-AL(I-1)
  220 AC(I)=AC(I)+AC(I-1)
C**   +--------------------------------------------------------+    **
C**   | Find  M-MR = # of real roots (+X,0)                    |    **
C**   | Find  M-ML = # of real roots (-X,0)                    |    **
C**   | Find  M-MC = # of imag roots (0,+X) or (0,-X)          |    **
C**   +--------------------------------------------------------+    **
      DO 230 K=M-KR(IL),M
      IF(DABS(AR(K)).GT.EFM) MR=K
      IF(DABS(AL(K)).GT.EFM) ML=K
      IF(DABS(AC(K)).GT.EFM) MC=K
  230 CONTINUE
C**   ---------------------------------------------------------    **
      CALL PUTX( X,ZERO,M-MR,RR,KR,II,EPS)
      IF(II.GT.IR) GOTO 500
      CALL PUTX(-X,ZERO,M-ML,RR,KR,II,EPS)
      IF(II.GT.IR) GOTO 500
      CALL PUTX( ZERO,X,M-MC,RR,KR,II,EPS)
      IF(II.GT.IR) GOTO 500
C**   +--------------------------------------------------------+    **
C**   | Check complex root for Err < EFM (EFM increase with KKK) |    **
C**   +--------------------------------------------------------+    **
      DO 400 KKK=1,3
C**   +--------------------------------------------------------+    **
C**   | Find complex roots : real parts between +0.5X and +X   |    **
C**   +--------------------------------------------------------+    **
      IF(KKK.EQ.1)
     *CALL GRAEFF(AR,KA,BB,KB,PR,NR,AR,MR,EPSX,ITM,ITS/2,IP)
      DO 310 IV=MR-1,1,-1
      IF(NR(IV).LE.0.OR.PR(IV).GT.2.005D0) GOTO 310
      P=2.0D0-PR(IV)**2
      Q=1.0D0
      DO 300 ISTEP=1,20
      CALL SYNTH2(AA,BB,M,P,Q,IM)
      CALL PQERR(BB,M,P,Q,IM,ERS,FX)
      PO=P
      P=XMIN(P,FX,0.001D0,ISTEP,IVY,BUF)
      IF(DABS(P-PO).LE.EPSX) GOTO 305
  300 CONTINUE
  305 CALL BAIRST(AA,BB,M,P,Q,IM,EPSX,EFM,IP)
      IF(DABS(Q-1.0).GT.0.125.OR.DABS(P).GT.2.25) GO TO 310
      CALL CHECK2(AA,BB,M,MB,P,Q,IM,EFM,IP)
      IF(M.EQ.MB) GOTO 310
      CALL PUTX(0.5D0*P*X,DSQRT(Q-0.25D0*P*P)*X,M-MB,RR,KR,II,EPS)
      NR(IV)=0
      IF(II.GT.IR) GOTO 500
  310 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Find complex roots : real parts between -X and -0.5X   |    **
```

```
C**   +--------------------------------------------------------+    **
      IF(KKK.EQ.1)
     *CALL GRAEFF(AL,KA,BB,KB,PL,NL,AL,ML,EPSX,ITM,ITS/2,IP)
      DO 330 IV=ML-1,1,-1
      IF(NL(IV).LE.0.OR.PL(IV).GE.2.005D0) GOTO 330
      P=PL(IV)**2-2.0D0
      Q=1.0D0
      DO 320 ISTEP=1,20
      CALL SYNTH2(AA,BB,M,P,Q,IM)
      CALL PQERR(BB,M,P,Q,IM,ERS,FX)
      PO=P
      P=XMIN(P,FX,0.001D0,ISTEP,IVY,BUF)
      IF(DABS(P-PO).LE.EPSX) GOTO 325
  320 CONTINUE
  325 CALL BAIRST(AA,BB,M,P,Q,IM,EPSX,EFM,IP)
      IF(DABS(Q-1.0).GT.0.125.OR.DABS(P).GT.2.25) GO TO 330
      CALL CHECK2(AA,BB,M,MB,P,Q,IM,EFM,IP)
      IF(M.EQ.MB) GOTO 330
      CALL PUTX(0.5D0*P*X,DSQRT(Q-0.25D0*P*P)*X,M-MB,RR,KR,II,EPS)
      NL(IV)=0
      IF(II.GT.IR) GOTO 500
  330 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Find complex roots : real parts between -0.5X and +0.5X  |    **
C**   +--------------------------------------------------------+    **
      EFM=EFM*4.0D0
      IF(KKK.EQ.1)
     *CALL GRAEFF(AC,KA,BB,KB,PC,NC,AC,MC,EPSX,ITM,ITS/2,IP)
      DO 350 IV=MC-1,1,-1
      IF(NC(IV).LE.0.OR.PC(IV).GT.2.010D0) GOTO 350
      P=PC(IV)
      Q=1.0D0
      DO 333 ISTEP=1,20
      CALL SYNTH2(AA,BB,M,P,Q,IM)
      CALL PQERR(BB,M,P,Q,IM,ERS,FX)
      PO=P
      P=XMIN(P,FX,0.001D0,ISTEP,IVY,BUF)
      IF(DABS(P-PO).LE.EPSX) GOTO 335
  333 CONTINUE
  335 DO 340 KK=1,2
      CALL BAIRST(AA,BB,M,P,Q,IM,EPSX,EFM,IP)
      IF(DABS(Q-1.0).GT.0.125.OR.DABS(P).GT.2.25) GO TO 340
      CALL CHECK2(AA,BB,M,MB,P,Q,IM,EFM,IP)
      IF(M.EQ.MB) GOTO 340
      CALL PUTX(0.5D0*P*X,DSQRT(Q-0.25D0*P*P)*X,M-MB,RR,KR,II,EPS)
      NC(IV)=0
      IF(II.GT.IR) GOTO 500
  340 P=-P
  350 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Increase the error tolerance EFM                       |    **
C**   +--------------------------------------------------------+    **
      EFM=EFM*2.5D0
      IF(II.GT.IR) GOTO 500
  400 CONTINUE
      WRITE(2,'(1X,A,1PE16.8)') 'NO ENOUGH ACCURATE ROOTS FOUND FOR',X
  500 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Output the answers                                     |    **
C**   +--------------------------------------------------------+    **
```

```
  700 WRITE(2,'(/A8,2A24,A8/)') 'I','Real.X(I)','Imag.X(I)','No'
      DO 710 II=1,M-1
      IF(KR(II).LE.0) GOTO 710
      IF(KR(II+1).NE.0) THEN
        WRITE(2,'(I8,1PE24.16,24X,I8)') II,RR(II),KR(II)
      ELSE
        WRITE(2,'(I8,1P2E24.16,I8)') II,RR(II),RR(II+1),KR(II)
      ENDIF
  710 CONTINUE
      RETURN
      END

************************************************************************
      SUBROUTINE BAIRST(AA,BB,M,P,Q,IM,EPS,EFM,IP)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AA(M),BB(M)
C**   ============================================================    **
C**   Input   : AA(M),M,P,Q,IM,EPS,EFM,IP                             **
C**   Output  : P,Q IN THE FACTOR (X*X-P*X+Q)                         **
C**   Working : BB(M)                                                 **
C**   ------------------------------------------------------------    **
C**   By Bairstow's method                                            **
C**   ============================================================    **
      IF(M.LE.3) RETURN
      DO 40 I=1,20
C**   +----------------------------------------------------------+    **
C**   | [X*X-P*X+Q][B(1)*X**(M-3)+..+B(M-2)]+B(M-1)*(X-P)+B(M)   |    **
C**   +----------------------------------------------------------+    **
      CALL SYNTH2(AA,BB,M,P,Q,2)
C**   ------------------------------------------------------------    **
      B0=BB(M)
      B1=BB(M-1)
      EE=DSQRT(DABS(B0*B0-P*B0*B1+Q*B1*B1))
C**   +----------------------------------------------------------+    **
C**   | [X*X-P*X+Q][C(1)*X**(M-4)+..+C(M-3)]+C(M-2)*(X-P)+C(M-1) |    **
C**   +----------------------------------------------------------+    **
      CALL SYNTH2(BB,BB,M-1,P,Q,2)
C**   +----------------------------------------------------------+    **
C**   | / C(M-1)  C(M-2) \ / -DP \   / B( M ) \                  |    **
C**   | |                | |     | = |       |                  |    **
C**   | \ C(M-2)  C(M-3) / \ +DQ /   \ B(M-1) /                  |    **
C**   +----------------------------------------------------------+    **
      D=BB(M-1)*BB(M-3)-BB(M-2)**2
      IF(DABS(D).LT.EPS) WRITE(2,'(1X,A)') 'BAIRST : |D| < EPS'
      DP=(BB(M-2)*B1-BB(M-3)*B0)/D
      DQ=(BB(M-1)*B1-BB(M-2)*B0)/D
      P=P+DP
      Q=Q+DQ
      IF(IP/32*2.NE.IP/16)
     *WRITE(2,'('' BAIRST '',1PE24.16,3E14.5)') P,Q,DP,DQ,D,EE,B1,B0
      IF(DABS(DP)+DABS(DQ).LE.EPS) RETURN
C**   +----------------------------------------------------------+    **
C**   | For general use following statement should be deleted    |    **
C**   +----------------------------------------------------------+    **
      IF(DABS(Q-1.0).GT.0.125.OR.DABS(P).GT.2.25) RETURN
   40 CONTINUE
      WRITE(2,'(1X,A)') 'BAIRST : NOT CONVERGE IN 20 ITERATION'
      RETURN
      END
```

```
*************************************************************************
      SUBROUTINE PUTX(XR,XI,N,RR,KR,II,EPS)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION RR(1),KR(1)
C**   ============================================================    **
C**   Test if (XR,XI) already exist in  RR(1:II-1)                    **
C**   ============================================================    **
      IF(N.LE.0) RETURN
      DO 30 I=1,II-1
      IF(KR(I).LE.0) GOTO 30
      IF(KR(I+1).NE.0) THEN
         IF(DABS(XR-RR(I))+DABS(XI-RR(I+1)).LE.EPS) RETURN
      ELSE
         IF(DABS(XR-RR(I)).LE.EPS) RETURN
      ENDIF
   30 CONTINUE
C**   ------------------------------------------------------------    **
      RR(II)=XR
      KR(II)=N
      IF(XI.NE.0.0D0) THEN
        RR(II+1)=XI
        KR(II+1)=0
      ENDIF
      II=II+N
      WRITE(2,'('' PUTX '',I8,1P2E24.16)') II,XR,XI
      RETURN
      END
*************************************************************************
      SUBROUTINE CHECK2(AA,BB,MA,MB,P,Q,IM,EFM,IP)
C**   ============================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AA(MA),BB(MA)
C**   ============================================================    **
C**   Check # of the factor (X*X - P*X + Q)   --->  (MB-MA)/2         **
C**            or the factor (X - P)          --->  (MB-MA)           **
C**   ------------------------------------------------------------    **
C**   Input   : AA(MA),MA,P,Q,IM,EFM,IP                               **
C**   Output  : MB                                                    **
C**   Working : BB(MA)                                                **
C**   ============================================================    **
      MB=MA
      DO 10 M=MA,IM+1,-IM
      IF(M.EQ.MA) THEN
        CALL SYNTH2(AA,BB,MB,P,Q,IM)
      ELSE
        CALL SYNTH2(BB,BB,MB,P,Q,IM)
      ENDIF
      CALL PQERR(BB,MB,P,Q,IM,ERS,ERR)
      IF(IP/64*2.NE.IP/32)
     *WRITE(2,'(I8,1P2E16.8)') (I,AA(I),BB(I),I=1,MB)
      IF(IP/32*2.NE.IP/16)
     *WRITE(2,'('' CHECK '',I5,1PE24.16,3E12.4)') MB,P,Q,ERR,EFM
      IF(ERR.GT.EFM) RETURN
      MB=MB-IM
   10 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE SYNTH2(AA,BB,M,P,Q,IM)
```

```
C**   ================================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AA(M),BB(M)
C**   ================================================================    **
C*I   A(1)*X**(M-1)+..+A(M)                                               **
C*O   =  (X-P)  *  (B(1)*X**(M-2)+..+B(M-1)) + B(M)                       **
C*O   =(X*X-P*X+Q)*(B(1)*X**(M-3)+..+B(M-2)) + B(M-1)*(X-P) + B(M)         **
C**   ----------------------------------------------------------------    **
C**   Input   : AA(M),M,P,Q,IM  (IM=1 or 2)                               **
C**   Output  : BB(M)                                                     **
C**   ================================================================    **
      BB(1)=AA(1)
      IF(IM.EQ.1) THEN
        DO 20 I=2,M
   20   BB(I)=AA(I)+BB(I-1)*P
      ELSE
        BB(2)=AA(2)+P*BB(1)
        DO 80 I=3,M
   80   BB(I)=AA(I)+BB(I-1)*P-BB(I-2)*Q
      ENDIF
      RETURN
      END
************************************************************************
      SUBROUTINE PQERR(BB,M,P,Q,IM,ES,ER)
C**   ================================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION BB(M)
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | |(X-P)*B1+B0)|**2=((X-P)*B1+B0)*((conj.X-P)*B1+B0)         |    **
C**   | =(B0-P*B1)**2+(X+conj.X)*(B0-P*B1)*B1+|X|**2*B1**2         |    **
C**   | =(B0-P*B1)**2+    P     *(B0-P*B1)*B1+ Q    *B1**2         |    **
C**   | =B0**2-P*B0*B1+Q*B1**2                                     |    **
C**   +--------------------------------------------------------------+    **
      IF(IM.EQ.1) THEN
        ES=BB(M)
      ELSE
        ES=BB(M)**2-P*BB(M)*BB(M-1)+Q*BB(M-1)**2
      ENDIF
      ER=DABS(ES)
      RETURN
      END
************************************************************************
      SUBROUTINE POLYFT(AO,RR,KR,A,M,EPSX,EPSF,ITM,ITS,IP,NF)
C**   ================================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AO(M),RR(M),KR(M),A(M,10)
C**   ================================================================    **
      K=NF/2+2
      CALL FFTANG(AO,RR,KR,A(1,1),A(1,2),A(1,3),A(1,4)
     *           ,A(1,5),A(K,5),M,EPSX,EPSF,ITM,ITS,IP,NF)
      RETURN
      END
************************************************************************
      SUBROUTINE FFTANG(AO,RR,KR,AA,KA,BB,KB,FF,LL
     *                 ,M,EPSX,EPSF,ITM,ITS,IP,NF)
C**   ================================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AO(M),RR(M),KR(M),AA(M),KA(M),BB(M),KB(M)
      DIMENSION FF(NF),LL(NF),BUF(14)
```

```
      DATA BASE/1024.0D0/,INV/-2/
C**   ============================================================    **
C**   Input   : AO(M),M,EPSX,EPSF,ITM,ITS,IP,NF                       **
C**   Output  : RR(M),KR(M)                                           **
C**   Working : AA(M),KA(M),BB(M),KB(M),FF(NF+2),LL(NF/2+1)           **
C**   ------------------------------------------------------------    **
C**   AO(I) = Coefficient of X**(M-I)                                 **
C**   ------------------------------------------------------------    **
C**   If KR(I).GT.0 and KR(I+1).NE.0 then                             **
C**        RR(I) = Real root                                          **
C**        KR(I) = Its multiplicity                                   **
C**   ------------------------------------------------------------    **
C**   If KR(I).GT.0 and KR(I+1).EQ.0 then                             **
C**        (RR(I),+-RR(I+1)) = complex roots                          **
C**        KR(I) = Its multiplicity (always even)                     **
C**   ------------------------------------------------------------    **
C**   By Graeffe's method and FFT and Bairstow's method               **
C**   ============================================================    **
      CALL GRAEFF(AA,KA,BB,KB,RR,KR,AO,M,EPSX,ITM,ITS,IP)
C**   ------------------------------------------------------------    **
      II=1
      DO 85 IL=1,M-1
      IF(KR(IL).LE.0.OR.IL.LT.II) GO TO 85
      IR=IL+KR(IL)-1
C**   ------------------------------------------------------------    **
      II=IL
      X=RR(II)
      IF(X.EQ.0.0D0) GOTO 85
C**   ------------------------------------------------------------    **
      DO 15 I=1,NF+2
   15 FF(I)=0.0D0
C**   +----------------------------------------------------------+    **
C**   | Scale the II-th root to unit magnitude                   |    **
C**   +----------------------------------------------------------+    **
      KFM=0
      XX=1.0D0
      KX=0
      DO 20 I=M,1,-1
      AA(I)=AO(I)*XX
      KA(I)=KX
      CALL NORMAL(AA(I),KA(I))
      IF(KA(I).GT.KFM.AND.AA(I).NE.0.0D0) KFM=KA(I)
      XX=XX*X
   20 CALL NORMAL(XX,KX)
C**   +----------------------------------------------------------+    **
C**   | FFM = Sum of the abs of the coefficients                 |    **
C**   +----------------------------------------------------------+    **
      FFM=0.0D0
      DO 30 I=1,M
      KI=KA(I)-KFM
      IF(KI.NE.0.AND.AA(I).NE.0.0D0) AA(I)=AA(I)*BASE**KI
      FFM=FFM+DABS(AA(I))
   30 FF(I)=AA(I)
      EFM=EPSF*FFM
      EPS=EPSX*100.0D0
C**   +----------------------------------------------------------+    **
C**   | Find real roots +X or -X                                 |    **
C**   +----------------------------------------------------------+    **
      IM=1
      P=1.0D0
```

```
      Q=0.0D0
      DO 40 K=1,2
      CALL CHECK2(AA,BB,M,MB,P,Q,IM,EFM,IP)
      CALL PUTX(P*X,Q,M-MB,RR,KR,II,EPS)
      IF(II.GT.IR) GOTO 85
   40 P=-P
C**   +--------------------------------------------------------+    **
C**   | Compute function values along the unit circle by FFT   |    **
C**   +--------------------------------------------------------+    **
      CALL FFTR(FF,FF,NF,INV)
      DO 50 I=1,NF/2+1
      AM=DSQRT(FF(2*I-1)**2+FF(2*I)**2)
      IF(IP/256*2.NE.IP/128)
     *WRITE(2,'(I8,1P3E16.8)') I,AM,FF(2*I-1),FF(2*I)
   50 FF(I)=AM
C**   +--------------------------------------------------------+    **
C**   | Find locations with local minimum errors               |    **
C**   +--------------------------------------------------------+    **
      IN=0
      DL=FF(2)-FF(1)
      DO 60 I=2,NF/2
      DR=FF(I+1)-FF(I)
      IF(DL.GT.0.0D0.OR.DR.LT.0.0D0) GOTO 60
      IN=IN+1
      LL(IN)=I
      IF(IP/128*2.NE.IP/64)
     *WRITE(2,'(I8,1P3E16.9)') I,FF(I-1),FF(I),FF(I+1)
   60 DL=DR
C**   +--------------------------------------------------------+    **
C**   | Index sort according to errors                         |    **
C**   +--------------------------------------------------------+    **
      IS=IN
   65 IE=IS-1
      DO 70 I=1,IE
      IU=LL(I)
      IV=LL(I+1)
      IF(FF(IU).LE.FF(IV)) GOTO 70
      LL(I)=IV
      LL(I+1)=IU
      IS=I
   70 CONTINUE
      IF(IS.LE.IE) GOTO 65
C**   +--------------------------------------------------------+    **
C**   | Refine the complex roots by minimizing the errors      |    **
C**   | Then by Bairstow's method                              |    **
C**   +--------------------------------------------------------+    **
      IM=2
      DX=2.0D0/NF
      DO 80 IV=1,IN
      XX=DX*(LL(IV)-1)
      DO 77 ISTEP=1,12
      P=2.0D0*DCOS(XX*3.1415926535897932D0)
      Q=1.0D0
      CALL SYNTH2(AA,BB,M,P,Q,IM)
      CALL PQERR(BB,M,P,Q,IM,ERS,FX)
      XO=XX
      XX=XMIN(XX,FX,0.5*DX,ISTEP,IVY,BUF)
      IF(DABS(XX-XO).LE.EPSX) GOTO 78
   77 CONTINUE
C**   ----------------------------------------------------------    **
```

```
   78 CALL BAIRST(AA,BB,M,P,Q,IM,EPSX,EFM,IP)
      IF(DABS(Q-1.0).GT.0.125.OR.DABS(P).GT.2.25) GO TO 80
      CALL CHECK2(AA,BB,M,MB,P,Q,IM,EFM,IP)
      IF(M.EQ.MB) GOTO 80
      CALL PUTX(0.5D0*P*X,DSQRT(Q-0.25D0*P*P)*X,M-MB,RR,KR,II,EPS)
      IF(II.GT.IR) GOTO 85
   80 CONTINUE
   85 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Output the answers                                    |    **
C**   +-------------------------------------------------------+    **
   90 WRITE(2,'(/A8,2A24,A8/)') 'I','Real.X(I)','Imag.X(I)','No'
      DO 100 II=1,M-1
      IF(KR(II).LE.0) GOTO 100
      IF(KR(II+1).NE.0) THEN
        WRITE(2,'(I8,1PE24.16,24X,I8)') II,RR(II),KR(II)
      ELSE
        WRITE(2,'(I8,1P2E24.16,I8)') II,RR(II),RR(II+1),KR(II)
      ENDIF
  100 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE POLYCT(AO,RR,KR,A,M2,EPSX,EPSF,ITM,ITS,IP)
C**   ===================================================    **
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION AO(M2),RR(M2),KR(M2),A(1)
C**   ===================================================    **
C**   Input   : AO(M2),M2,EPSX,EPSF,ITM,ITS,IP               **
C**   Output  : RR(M2),KR(M2)                                **
C**   Working : A(MM*14)         (MM=M2-1, M=M2/2, M2 is even)   **
C**   -------------------------------------------------------    **
C**   (AO(2*I-1),AO(2*I)) = Complex coefficient of X**(M-I)      **
C**   -------------------------------------------------------    **
C**   A(I) = Coefficient of Y**(MM-I)                         **
C**        Half # of roots Y = Roots X                        **
C**        Half # of roots Y = Conjugate of roots X           **
C**   -------------------------------------------------------    **
C**   If KR(I).GT.0 and KR(I+1).LT.0 then                     **
C**        RR(I) = Real root                                  **
C**        KR(I) = Its multiplicity                           **
C**   -------------------------------------------------------    **
C**   If KR(I).GT.0 and KR(I+1).EQ.0 then                     **
C**        (RR(I),+-RR(I+1)) = complex roots                  **
C**        KR(I) = Its multiplicity (always even)             **
C**   -------------------------------------------------------    **
C**   Double the degree of the polynomial such that           **
C**   The roots include the original roots and their conjugates    **
C**   So the coefficients are all real                        **
C**   ===================================================    **
      M=M2/2
      MM=M2-1
      DO 150 I=1,MM,2
      IX=I
      IY=I
      SR=0.0D0
      SI=0.0D0
  135 IF(IX.GT.0.AND.IY.LT.MM) THEN
        IY=IY+2
        SI=SI+AO(IX)*AO(IY)+AO(IX+1)*AO(IY+1)
```

```
        IX=IX-2
        IF(IX.GT.0) THEN
        SR=SR+AO(IX)*AO(IY)+AO(IX+1)*AO(IY+1)
        ENDIF
        GO TO 135
      ENDIF
      A(I  )=2.0D0*SR+AO(I)*AO(I)+AO(I+1)*AO(I+1)
      A(I+1)=2.0D0*SI
  150 CONTINUE
C**   ------------------------------------------------------------    **
      CALL POLYRT(A,RR,KR,A(MM+1),MM,EPSX,EPSF,ITM,ITS,IP)
C**   ------------------------------------------------------------    **
      WRITE(2,'(/A8,2A24,A8/)') 'I','Real.X(I)','Imag.X(I)','No'
      DO 300 II=1,MM-1
      IF(KR(II).LE.0) GOTO 300
      XR=RR(II)
      IF(KR(II+1).EQ.0) THEN
        XI=RR(II+1)
        IX=2
      ELSE
        XI=0.0D0
        IX=1
      ENDIF
C**   ------------------------------------------------------------    **
      DO 260 IR=1,IX
      DO 210 I=1,M2
  210 A(I)=AO(I)
      DO 230 K=MM,3,-2
      DO 220 I=3,K,2
      A(I  )=A(I  )+A(I-2)*XR-A(I-1)*XI
  220 A(I+1)=A(I+1)+A(I-2)*XI+A(I-1)*XR
      IF(DSQRT(A(K)**2+A(K+1)**2).GT.EPSF) GOTO 250
  230 CONTINUE
C**   ------------------------------------------------------------    **
  250 N=(MM-K)/2
      WRITE(2,'(I8,1P2E24.16,I8)') II,XR,XI,N
      XI=-XI
  260 CONTINUE
  300 CONTINUE
      RETURN
      END
*************************************************************************
```

習題

1. 試以圓交點法求下列多項式之根。

   (a) $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

   (b) $x^5 + 2x^4 + 4x^3 + x^2 + 2x + 4 = 0$

   (c) $x^7 + 1 = 0$

2. 試以牛頓法求題1多項式之實數根。初始值可利用題1之近似根。

3. 試以Bairstow法求題1多項式之複數根。初始值可利用題1之近似根。

參考文獻

1. Ralston, A. and Rabinowitz, P., *A First Course in Numerical Analysis,* 2nd ed., McGraw-Hill, 1978.

2. Bareiss, E.H., "Resultant Procedure and the Mechanization of the Graeffe Process," J. Assoc. Comput. Mach., Vol.7, pp.346-386, 1960.

第十一章

誤差分析

11.1 前言

工程問題之分析或求解方法可分為二大類：一為解析方法，其解稱為解析解；一為數值方法，其解稱為數值解。這二種方法各有其優缺點，亦有其適用場合，二者有時需相輔相成。工程上難免有些問題，事實上應該是大部分問題，需用數值方法求解。而數值方法離不開數值之運算，但數值無法在賴以計算之計算機內正確無誤地表示，亦即數值難免含有誤差。而根據有誤差的數值，運算後所得的數值亦難免含有誤差。有些數值方法會使運算結果之誤差太大而使結果不能用；而有些數值方法不會或比較不會有此現象。 誤差分析之目的是希望了解某一數值方法之運算方式或過程對誤差消長之特性，因此而可避免採用會產生太大誤差之方法，或找出造成誤差過大之原因及其解決之道。任何問題之分析都不希望計算的結果因誤差太大而不能用，更不希望用了誤差太大的錯誤結果而不知悉。因此誤差分析在選擇計算方法時，就顯得相當重要。了解造成誤差的原因方能在決定運算過程時隨時避凶趨吉，因此為程式設計時重要的基本知識。本章將先介紹誤差之來源及產生方式。再介紹誤差如何影響其後續數值結果之誤差，即所謂誤差傳播之原理與分析方法。最後，介紹一些常會造成誤差過大問題的範例，並分析其造成之原因及提供一些解決之道。

11.2 誤差來源

欲了解誤差之消長，須先掌握誤差之來源或成因。誤差之來源一般分為三類：一為固有(Inherent)誤差，一為截項(Truncation)誤差，一為捨入(Roundoff)誤差。 固有誤差為原始數據之誤差，對於一些由度量取得之物理量，一方面由於物理量本身的不確定性，一方面由於度量儀器的精準度，其數值即含有某一程度之誤差。 再者若資料取錯，所造成的固有誤差則更為嚴重。此項誤差與程式設計關係不大，因此不再詳加討論，不過有句老話還是要提一下：垃圾進垃圾出(garbage in garbage out)。

截項誤差為數值方法之誤差， 若數值方法在理論上須考慮無窮項，或須經無窮個步驟之運算方能求得正確值，而實際計算僅能考慮有限項或僅做有限步驟，由此所造成之誤差謂之。須考慮無窮項的數值方法很多，有用泰勒級數 (Tayler series)直接計算各種函數值(如$\sin x$)者，或用泰勒級數推導數值方法之運算式者， 如倫伯格積分法或解常微分方程式之郎吉庫達法或亞當氏法。 須經無窮個步驟的數值方法亦很多，如求非線性方程式之根或極值之反覆試算法。幾乎所有非線性方程式之求根或求極值均無有限步驟的方法，有些線性方程式之根亦以反覆試算法求解。可知反覆試算法在數值方法之普遍性。所幸，只要反覆試算會收斂，大部分不會有誤差過大之問題。

捨入誤差為計算工具之誤差，係因計算機僅能存放有限長度之位元所造成。計算機所能存放之位元長度，稱有效位數t。 令計算機之基底為B (大部分計算機$B=2$，$REAL*4$之單精度實數$t=24$，$REAL*8$之倍精度實數$t=53$；人腦則習慣於$B=10$，對應於單精度實數約$t=7$，倍精度實數約$t=16$。本章若提到之$t<20$則其$B=10$，若$t\geq 20$則其$B=2$，應不至於混淆，不過會以$B=10$為主說明較為習慣)，若以所謂4捨5入方式處理不能存放的值，則此誤差稱捨入誤差。其可能最大相對誤差為$\epsilon_M=\frac{1}{2}B^{-t+1}$；另一種方式為全部捨去不能存放的值，則此誤差稱捨去誤差。其可能最大相對誤差為$\epsilon_M=B^{-t+1}$，為前者之二倍，目前幾乎沒有計算機採用。注意10進位的數值，例如0.1，用2進位數即不能正確表示，即使採用倍精度亦是如此，因此10個0.1相加不一定會等於1.0，這跟$\frac{1}{3}$不能用10進位數正確表示，一樣都是因為有限之有效位數所造成。因每一運算之結果，均受計算機有限之有效位數之限制而無法全部存放，故每一運算步驟均可能引進捨入誤差，此舉常造成誤差分析之困難。

11.3 誤差之表示法

誤差之表示法有二：一為絕對誤差，一為相對誤差。若某數之近似值為$\bar{x}$，正確值為x，則絕對誤差δ_x定義於式(11.1)，相對誤差ϵ_x定義於式(11.2)。但因一般之x並未確知，故將相對誤差改定義或近似為第二式。即使如此定義，因x未知而使正確的絕對誤差或相對誤差仍然不得而知。

$$\delta_x = x - \bar{x} \tag{11.1}$$

$$\epsilon_x = \delta_x / x \doteq \delta_x / \bar{x} \tag{11.2}$$

前節計算機之有效位數之含義，亦可用來表示數值之誤差，此時有效位數為數值正確之位元長度。通常數值之有效位數不會大於計算機之有效位數。因此有效位數即數值中有效位元之數目。有效位元當然指較高位元之前幾個正確位元，一般認為最後之有效位元之最大絕對誤差為該位元為1之值之一半，亦即與計算機之有效位數之捨入誤差類似。若某數之有效位數為t，其近似值$\bar{x}$以式(11.3)表示，其正確值x以式(11.4)表示，則其絕對誤差δ_x與相對誤差ϵ_x及其上限值分別如式(11.5)與式(11.6)所示。

$$\bar{x} = fB^e, \qquad B^{-1} \le f < 1 \tag{11.3}$$

$$x = fB^e + gB^{e-t}, \qquad |g| \le \frac{1}{2} \tag{11.4}$$

$$\delta_x = gB^{e-t} \le \frac{1}{2}B^{e-t} \tag{11.5}$$

$$\epsilon_x = (g/f)B^{-t} \le \frac{1}{2}B^{-t+1} \tag{11.6}$$

因以有效位數表示誤差具有直覺之效，後繼討論誤差將會經常用有效位數表示，且若以符號$ccccbbb$表示某一個數值時，表示該數值之近似值即計算機存放之值為$\bar{x} = 0.ccccbbb * B^e$，其$f = 0.ccccbbb$，計算機之有效位數為7，數值之有效位數為$t = 4$，其最大絕對誤差為$\delta_x = 0.0001000B^e/2 = \frac{1}{2}B^{e-4}$，最大相對誤差為$\epsilon_x = \frac{1}{2}B^{-3}$。注意未顯示在符號中者為$e$與$c$及$b$之真正數值，均為討論時不必考慮之值。不過在討論二值之和時會將二數同位元之值對齊如下所示：

$$\begin{array}{r} ccccbbb \\ \quad ccccbbb \\ \hline ccccccb \end{array} \qquad \begin{array}{r} ccccbbb \\ ccccbbb \\ \hline ccbbbbb \end{array}$$

11.4 誤差傳播

二個有誤差的數值經過運算後之結果亦會有誤差存在，此種由原有數值之誤差而導致運算結果之誤差之現象稱為誤差傳播。數值之運算主要由加減乘除之四則運算達成，因此該四則運算結果之誤差與原始數值 x 與 y 間之誤差關係，為誤差傳播之最基本關係：

$$z = x + y \tag{11.7}$$

$$\delta_{x+y} = \delta_x + \delta_y \tag{11.8}$$

$$\epsilon_{x+y} = \frac{\bar{x}}{\bar{x}+\bar{y}}\,\epsilon_x + \frac{\bar{y}}{\bar{x}+\bar{y}}\,\epsilon_y \tag{11.9}$$

$$z = x - y \tag{11.10}$$

$$\delta_{x-y} = \delta_x - \delta_y \tag{11.11}$$

$$\epsilon_{x-y} = \frac{\bar{x}}{\bar{x}-\bar{y}}\,\epsilon_x - \frac{\bar{y}}{\bar{x}-\bar{y}}\,\epsilon_y \tag{11.12}$$

$$z = xy \tag{11.13}$$

$$\delta_{xy} = \bar{y}\,\delta_x + \bar{x}\,\delta_y \tag{11.14}$$

$$\epsilon_{xy} = \epsilon_x + \epsilon_y \tag{11.15}$$

$$z = x/y \tag{11.16}$$

$$\delta_{x/y} = \frac{1}{\bar{y}}\,\delta_x - \frac{\bar{x}}{\bar{y}^2}\,\delta_y \tag{11.17}$$

$$\epsilon_{x/y} = \epsilon_x - \epsilon_y \tag{11.18}$$

式(11.8)(11.11)(11.14)(11.17)為絕對誤差間之關係，可由 $x = \bar{x} + \delta_x$ ，$y = \bar{y} + \delta_y$ 代入式(11.7)與式(11.10)得 $z = x \pm y = (\bar{x} + \delta_x) \pm (\bar{y} + \delta_y)$ ，故 $\delta_z = \delta_{x\pm y} = (x \pm y) - (\bar{x} \pm \bar{y}) = \delta_x \pm \delta_y$ ，因此得式(11.8)與式(11.11)之關係。注意這些關係之推導與微分式之推導類似(又如 $\delta_{\sin\theta} = \cos\theta\,\delta_\theta$ 等)，因此式(11.14)與式(11.17)之推導從略。相對誤差關係式可簡單地由絕對誤差關係式分別除以 $\bar{x}+\bar{y}$ 、$\bar{x}-\bar{y}$ 、$\bar{x}\bar{y}$ 與 $\bar{x}/\bar{y}$ ，並引用後列之定義式而得。$\epsilon_{x+y} = \delta_{x+y}/(\bar{x}+\bar{y})$ 、$\epsilon_{x-y} = \delta_{x-y}/(\bar{x}-\bar{y})$ 、$\epsilon_{xy} = \delta_{xy}/(\bar{x}\bar{y})$ 、$\epsilon_{x/y} = \delta_{x/y}/(\bar{x}/\bar{y})$ 、$\epsilon_x = \delta_x/\bar{x}$ 與 $\epsilon_y = \delta_y/\bar{y}$ 。

由以上之誤差關係可看出下面二個簡單關係：

(1)二數值之積或商之相對誤差為二數值者之和或差。通常誤差有正有負，其上限應取二相對誤差之絕對值之和。因此二數值相乘之積或相除之

商之相對誤差會接近二數值中之相對誤差之較大者。以有效位數而言：一個乘數如果只有p個有效位元，則其乘積之有效位數不可能超過p位。(2)二數值之和或差之絕對誤差為二數值者之和或差。通常誤差有正有負，其上限應取二絕對誤差之絕對值之和。二數值之和差運算如用下列符號式表示，即可清楚看出各值之有效位數間之下述關係：

(2a)二個加數之同位元之數值必須均正確，和數之同位元之數值才會正確。因此和數中由左向右第一個b之位置會在二個加數中最左邊之b處。

(2b)和差運算後之和數之有效位數有可能大量減少。當x與y相當接近時，$|x-y| << |x| \doteq |y|$，由相對誤差之關係式知ϵ_x放大$|x|/(|x-y|)$倍，ϵ_y放大$|y|/(|x-y|)$倍，而$|x-y|$之值可以很小甚至為零，因此相對誤差可能會被放大至無窮大。參考下列右邊符號式，如果二原始值有前q個位元相等則其差值之有效位數即會減少q位，相當於其相對誤差放大約B^q倍。

$$\begin{array}{r} ccccbbb \\ -\ ccccbbb \\ \hline ccccccb \end{array} \qquad \begin{array}{r} cccccbb \\ -\ ccccbbb \\ \hline ccbbbbb \end{array}$$

因此很接近之二個絕對值相減(簡稱近值相減)之運算，都會造成運算結果之誤差嚴重放大。如果這些嚴重放大之誤差又不斷影響至最後結果，則會造成結果之誤差過大而沒有用處。由於和差運算為最基本之運算，如果未加注意，即很可能會遇到近值相減之運算。如果計算過程中必然會有近值相減之運算，則此數值方法極有可能造成誤差過大之問題，因此必須設法避免之。但和差運算可能佔整個運算量之一大半，有時也會有防不勝防的情形，因此而造成誤差分析之困難。另外一種運算為一系列有正有負之數值之和，如泰勒級數和，若因正負值抵消而使級數和之絕對值遠小於運算過程最大值，也會造成誤差嚴重放大之現象。 運算過程最大值，指運算過程中之部分和或單項值之最大絕對值。此時級數和之絕對誤差約等於運算過程最大值之捨入絕對誤差(級數和之絕對誤差應約等於運算過程中之部分和或單項值之最大絕對誤差，但該等值不易取得，且若單項值之有效位數均等於計算機之有效位數，則採用運算過程最大值之捨入誤差應屬合理)。因此其相對誤差之放大倍數約為運算過程最大值與級數和之比。注意級數於每加一項時之部分和，不一定會減少很多位有效位數，亦即每次運算之相對誤差放大不多，但相對誤差經很多次放大之相乘效果，最後一樣會將相對誤差嚴重放大。此種情形，基本上與近值相減之特性相似。表面上雖無二個近值相減，但實際上可視

為二個約等於運算過程最大值之近值相減。

理論上，n 個值相加之和數之絕對誤差最大上限值為加數之絕對誤差之絕對值之和。但計算機採用捨入誤差，誤差有正有負，其平均值為零，故和數之絕對誤差會互相抵消，因此以上限值做為和數之誤差即顯得太大而不切實際。以統計方法估計和數之絕對誤差應較合理。根據中心極限定理，和數之絕對誤差 δ_n 為加數之絕對誤差 δ_{x_i} 之和，其分布為常態分布 (Normal distribution)。若和數之絕對誤差 δ_n 之平均值為 0，標準偏差為 σ_n。則 $|\delta_n| < 2\sigma_n$ 之機率為 95.45%，$|\delta_n| < 3\sigma_n$ 之機率為 99.73%。可見絕大多數情況下和數之絕對誤差在 $3\sigma_n$ 之內。若加數之絕對誤差之標準偏差為 σ_{x_i}，則和數之絕對誤差之標準偏差為 $\sigma_n = \sqrt{\sum_{i=1}^{n} \sigma_{x_i}^2}$。若二數之絕對誤差均為常態分布，則絕對誤差之比等於其標準偏差之比。因此和數之絕對誤差約為 $\delta_n = \sqrt{\sum_{i=1}^{n} \delta_{x_i}^2}$。例如計算 $n = 100$ 個絕對誤差大約相同之值之和時，和數之絕對誤差之上限為 $n = 100$ 倍加數之絕對誤差，但以統計方法計算，和數之絕對誤差約為 $\sqrt{n} = 10$ 倍加數之絕對誤差。考慮計算機之捨入誤差，設其相對誤差 ϵ 為均勻分布於 $\pm\epsilon_M$ 之間，$\epsilon_M = \frac{1}{2}B^{-t+1}$，則其平均值為 0，標準偏差為 $\sigma_M = \sqrt{1/3}\ \epsilon_M$。若以常態分布近似均勻分布，則 $|\epsilon| < \epsilon_M = \sqrt{3}\ \sigma_M$ 之機率為 91.67%，注意該值已很接近均勻分布時之正確機率 100%。另一參考值為 $|\epsilon| < \sqrt{3}\ \epsilon_M = 3\,\sigma_M$ 之機率為 99.73%。

11.5 誤差過大之範例

本節將列舉幾個會產生過大誤差之範例並分析其原因及解決之道。

例一：計算下列正負項交替之無窮級數之和：

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \cdots \tag{11.19}$$

以泰勒級數計算 $\sin x$ 時，理論上該級數對任意大小之 x 均會收斂。雖然計算無窮級數和難免有截項誤差存在，但其誤差一般都可以控制在可接受之範圍內，不太容易造成誤差過大之問題。此例主要問題是由下列二項因素所造成：(1) 計算機有限之有效位數。(2) 無窮級數有正及負項，因正負值抵消而使級數和之絕對值遠小於運算過程最大值。以 $\theta = 1830^o$ 為例，$x = 31.93953$，數列中最大值之項為 $\frac{1}{31!}x^{31} = 0.5237940 * 10^{13}$，最大部分和為 $0.2655370 * 10^{13}$，故運算過程最大值為 $0.5237940 * 10^{13}$，而級數和之理論值為 0.5，因此相對誤差之放大倍數高達 10^{13}。若計算機之有效位數

為7，則運算過程最大值之捨入絕對誤差可達$0.5*10^6$，級數和之絕對誤差亦可達$0.5*10^6$。因此實際由計算機所得之答案為$0.665275*10^5$應是在預料之中。若計算機之有效位數為16，則運算過程最大值之捨入絕對誤差可達$0.5*10^{-3}$，級數和之絕對誤差亦可達$0.5*10^{-3}$。此時實際由計算機所得之答案為0.499905也是在預料之中。採用倍精度通常可以解決大部分問題，不過還是需要對級數之最大部分和或最大項有所了解，以確定該級數之解是否可用。

$$\begin{array}{r} ccccccc \\ ccccccc \\ ccccccc \\ ccccccc \\ ccccccc \\ ccccccc \\ \hline bbbbbbb \end{array} \qquad \begin{array}{r} cccccccccccccccc \\ cccccccccccccccc \\ cccccccccccccccc \\ cccccccccccccccc \\ cccccccccccccccc \\ cccccccccccccccc \\ \hline ccbbbbbbbbbbbbbb \end{array}$$

例二：求下列二次多項式之根：

$$ax^2 - 2bx + c = 0 \tag{11.20}$$

$$x_1 = \frac{b+\sqrt{b^2-ac}}{a}, \qquad x_2 = \frac{b-\sqrt{b^2-ac}}{a} = \frac{c}{b+\sqrt{b^2-ac}} = \frac{c}{ax_1} \tag{11.21}$$

設$b>0$，如果$b^2 >> ac$，則b與$\sqrt{b^2-ac}$幾乎相等，則x_2由上述公式第一式計算誤差會很大，應改用第二式或第三式計算較為準確。以$a=1,\ b=100,\ c=1$為例，假設用三位有效位數計算，可得$x_1=200$，以第一式計算得$x_2=0$，以第二式或第三式計算得$x_2=0.0500$，顯然較第一式算得之$x_2=0$精確。以賈柯比法求對稱矩陣之特徵值及特徵向量時，即應考慮到此情形。

例三：利用下列之遞迴 (recurrence) 式(11.22)計算$J_n(x),\ n=0,1,2,\cdots$：

$$J_{n-1}(x) + J_{n+1}(x) = (2n/x)J_n(x) \tag{11.22}$$

$$J_o(x) + 2\sum_{n=1}^{\infty} J_{2n}(x) = 1 \tag{11.23}$$

當$n < x$時，可由式(11.22)按$n = 0,1,2,\cdots$之順序計算。否則嚴重的捨入誤差會得不到準確的結果。此時應該改按n值由大到小的順序計算。

以$x = 1$為例，令$kJ_8(1) = 0,\ kJ_7(1) = 1$，然後由式(11.22)可得下表之各$kJ_n(1)$之值。再利用式(11.23)之關係可得$k = 668648$。該值除第二行即得第三行之各$J_n(1)$值。至於第四行之$J_n^*(1)$係由$J_o(1)$與$J_1(1)$之值按式(11.22)依$n = 1, 2, \cdots$之順序計算所得，注意$n > 4$以後之值誤差愈變愈大。此處誤差變大之原因為$J_{n+1}(x)$係由幾乎相等之二值$(2n/x)J_n(x)$與$J_{n-1}(x)$相減而得，因此誤差會被放大。反向計算時$J_{n-1}(x)$則係由較大之值$(2n/x)J_n(x)$與較小之值$J_{n+1}(x)$相減而得，因此誤差不會被放大。凡是利用類似式(11.22)之遞迴式時，均可能有一個方向之計算會有誤差放大之現象(事實上式(11.22)為二階差分式，亦可按第十二章之方法分析是否有不穩定之寄生根)，可按本例做法改變方向計算之。

n	$kJ_n(1)$	$J_n(1)$	$J_n^*(1)$
8	0	0.00000	−.96202
7	1	0.00000	−.06913
6	14	0.00002	−.00579
5	167	0.00025	−.00034
4	1656	0.00248	.00240
3	13081	0.01956	.01955
2	76830	0.11490	.11490
1	294239	0.44005	.44005
0	511648	0.76520	.76520

例四：計算下式之奇異值積分：

$$I = \int_{-a}^{a} \frac{f(x)dx}{x} = \int_{0}^{a} \frac{[f(x) - f(-x)]dx}{x} \tag{11.24}$$

設$f(x) > 0$，若以第一式計算積分時，則當積分式由x=$-a$積至x=0時，積分值趨近於負無窮大，雖然繼續積分會被$x > 0$之正積分值抵消而為有限值，但因計算機有限之有效位數使誤差達最大之部分積分值之$\frac{1}{2}B^{-t+1}$，但該值會遠大於最後之積分值致使積分值誤差過大而不能使用。若改用第二式計算，只要在計算$f(x) - f(-x)$時略加注意，即不會有誤差過大之問題。

例五：考慮下列各計算式：

$$ab - ac = a(b - c)\,, \qquad \tfrac{b}{a} - \tfrac{c}{a} = \tfrac{b-c}{a} \tag{11.25}$$

$$\frac{b^2-c^2}{b-c} = b+c\ , \qquad (b+d)-b = d \tag{11.26}$$

設 b 與 c 幾乎相等，$d << b$。則上列式中，最好採用等號右邊之算式為宜。式 (11.26) 之情形可明顯看出左式會造成誤差過大之問題。由下列之 $(b+d)-b$ 之符號式亦可明顯看出 d 之有效位數減少情形。

$$\begin{aligned} b: &\quad cccccc \\ d: &\quad\ \ cccccc \\ b: &\quad cccccc \\ \hline (b+d)-b: &\quad\ \ ccccbbb \end{aligned}$$

式 (11.25) 之情形並不十分明顯，茲以上節所示方法分析之。

$$\epsilon_{ab-ac} = \frac{b}{b-c}(\epsilon_a+\epsilon_b+\epsilon_{m1}) - \frac{c}{b-c}(\epsilon_a+\epsilon_c+\epsilon_{m2}) + \epsilon_{s1} \tag{11.27}$$

$$\epsilon_{a(b-c)} = \frac{b}{b-c}\epsilon_b - \frac{c}{b-c}\epsilon_c + \epsilon_{s2} + \epsilon_a + \epsilon_{m3} \tag{11.28}$$

注意上式計算誤差上限時應取各項絕對值之和。其中 ϵ_{m1} 等為積數之捨入誤差，ϵ_{s1} 等為差數之捨入誤差。在 ϵ_{ab-ac} 中積數之捨入誤差有 ϵ_{m1} 與 ϵ_{m2} 二項，二值不一定會相等或同號，且又經分別放大 $\frac{b}{b-c}$ 或 $\frac{c}{b-c}$ 倍，機率上，顯然比 $\epsilon_{a(b-c)}$ 中僅有一項未經放大之 ϵ_{m3} 者為大。下式 $\epsilon_{b/a-c/a}$ 之情形亦與前者類似，不再贅述。

$$\epsilon_{b/a-c/a} = \frac{b}{b-c}(\epsilon_b-\epsilon_a+\epsilon_{m1}) - \frac{c}{b-c}(\epsilon_c-\epsilon_a+\epsilon_{m2}) + \epsilon_{s1} \tag{11.29}$$

$$\epsilon_{(b-c)/a} = \frac{b}{b-c}\epsilon_b - \frac{c}{b-c}\epsilon_c + \epsilon_{s2} - \epsilon_a + \epsilon_{m3} \tag{11.30}$$

例六：考慮下列計算標準偏差之計算式：

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{11.31}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2 \tag{11.32}$$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i^2 - 2\mu x_i + \mu^2)$$

$$= \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \mu^2 \tag{11.33}$$

當n個x_i值相當接近時，σ^2將可能遠比μ^2為小，亦即$\frac{1}{n}\sum_{i=1}^{n} x_i^2$與$\mu^2$幾乎相等，以式(11.33)計算$\sigma^2$將會使誤差太大，因此應採用式(11.32)為宜。

例七：考慮下列最小二乘法之計算式：

$$y_i = k_o + k_1 x_i\,, \quad i = 1, 2, \cdots, n \tag{11.34}$$

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{Bmatrix} k_o \\ k_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{Bmatrix} \tag{11.35}$$

上式前乘係數矩陣之轉置矩陣可得式(11.36)。將式中之$\sum_{i=1}^{n} x_i$用$n\mu$取代後，再消去第二式中k_o之係數可得式(11.37)。注意其第二式k_1之係數與式(11.33)之計算式相當，可見在x_i接近相等時，該係數之誤差亦會太大。

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{Bmatrix} k_o \\ k_1 \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{Bmatrix} \tag{11.36}$$

$$\begin{bmatrix} n & n\mu \\ 0 & \sum_{i=1}^{n} x_i^2 - n\mu^2 \end{bmatrix} \begin{Bmatrix} k_o \\ k_1 \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i - \mu \sum_{i=1}^{n} y_i \end{Bmatrix} \tag{11.37}$$

若將式(11.34)改為式(11.39)再以最小二乘法求K_o, K_1，即可得類似計算式(11.32)之式(11.40)。比較式(11.40)與式(11.37)，其第二式相同，故$K_1 = k_1$；比較第一式，知$K_o = k_o + \mu k_1$。

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{11.38}$$

$$y_i = K_o + K_1(x_i - \mu)\,, \quad i = 1, 2, \cdots, n \tag{11.39}$$

$$\begin{bmatrix} n & 0 \\ 0 & \sum_{i=1}^{n} (x_i - \mu)^2 \end{bmatrix} \begin{Bmatrix} K_o \\ K_1 \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} (x_i - \mu) y_i \end{Bmatrix} \tag{11.40}$$

例八：考慮下列聯立線性方程式之解：

$$\begin{aligned} x_1 + \quad 5x_2 &= 7.0 \\ 5x_1 + 25.01x_2 &= 35.01 \end{aligned} \tag{11.41}$$

上式之解為$x_1 = 2.0$，$x_2 = 1.0$。若將其中之常數35.01改為35.02，則其解變成$x_1 = -3.0$，$x_2 = 2.0$。雖然常數項僅增加不到0.03%，但結果之x_2居然增加一倍，而x_1甚至改變符號。產生這種誤差的原因，也是由於二個幾乎相等的值相減所造成。將第一式乘5減第二式可得$0.01x_2 = 0.01$。注意其中二個數值0.01一個由25.01減25.00而來；另一個由35.01減35.00而來。二個數值之有效位數都少了3位，也就是相對誤差放大了1000倍以上。事實上一個放大$25.01/0.01 = 2501$倍；另一個放大$35.01/0.01 = 3501$倍。因此即使僅0.03%的相對誤差經放大3500倍後，可以造成解答有100%的差異。

解聯立線性方程式會造成誤差過大之主要原因有二：一為聯立方程式未知數N很大使誤差累積過多。二為和數變小使相對誤差嚴重放大。若係數矩陣之半帶寬為M，將其分解為下三角矩陣與上三角矩陣之乘積時，其對角元素須經M個數值相加而得，其絕對誤差為這些加數中之最大之絕對誤差，乘以最多為$\sqrt{M}$之值。但這些加數也是經過類似的和算之三個數值相乘除之積。因此最後一個對角元素約須經過NM^2個數值之間接或直接之相加而得。若僅考慮誤差之累積，且以統計方法計算，其誤差可達原始數值誤差之$\sqrt{NM^2}$倍。因此當NM^2超過一百萬時應以倍精度計算為宜。和數變小指和數之值遠小於最大之部分和或最大之加數(均指絕對值)，因此使相對誤差放大。此種情形常發生於接近奇異之矩陣。

因樞紐元素之相對誤差會傳給被其所除之元素，其影響範圍較其他元素大，故應儘量選擇相對誤差小之元素做為樞紐元素。一般徹底尋樞紐法選擇絕對值最大之元素做為樞紐元素，應該是在不知道各元素之誤差之情況下之最佳選擇，其理由有二：

(1)和數變小會使相對誤差放大。和數變的愈小成為最大值之機會愈小。
(2)一個加數之絕對誤差，不論傳給大的和數或小的和數均相等。但同樣大小之絕對誤差除以和數後之相對誤差即與和數之大小成反比，因此愈小的和數被感染之相對誤差愈大。故最大值較不可能有大的相對誤差。

注意(2)項所指之小和數為其加數本來就很小；有別於(1)項所指之和數變小係由大的幾乎相等之加數互相抵消之結果。對於不尋樞紐而不做行或列對調之分解，如能對於對角元素之最後值，與運算過程最大值比較其大小，可了解對角元素之有效位數是否有減少，應該對判斷結果之誤差有幫助。若要真正知道結果之誤差範圍，則須要記錄各元素之誤差，再利用誤差傳播公式，追蹤各元素運算後之誤差，以做為尋找樞紐元素時

之依據。但記錄各元素之誤差既費時又須存放空間，故幾乎沒有程式考慮之。因此計算結果之準確性有時還需要由使用者加以判斷。

例九：考慮下列n個數值x_i之乘積p：

$$p = \prod_{i=1}^{n} x_i \ , \qquad \epsilon_p = \sum_{i=1}^{n} \epsilon_{x_i} \ , \qquad |\epsilon_p| \le \sum_{i=1}^{n} |\epsilon_{x_i}| \tag{11.42}$$

$$\sigma^2_{\epsilon_p} = \sum_{i=1}^{n} \sigma^2_{\epsilon_{x_i}} \doteq m\, \sigma^2_{\epsilon_{x^*}} \ , \qquad \sigma_{\epsilon_p} \doteq \sqrt{m}\, \sigma_{\epsilon_{x^*}} \tag{11.43}$$

由上節之誤差傳播分析，二數值相乘或相除，其相對誤差為二數者之和或差。但誤差有正有負，故相對誤差之絕對值有時相加有時相減，與乘或除無對應關係。$|\epsilon_p|$之上限為$|\epsilon_{x_i}|$之和。但該值常太大而不切實際。在此改用統計方法計算相對誤差的標準偏差如式(11.43)。式中x^*表示相對誤差較大之數值，m為相對誤差與x^*者相當之數值之總數，該值介於n與1之間。如果n個值之相對誤差相當或有效位數相同，則$m = n$，以十進位$B = 10$而言：大約100個有效位數相同的數值相乘或除之積或商之有效位數會少一位數。如果n個值中只有一個值的相對誤差特別大，則$m = 1$，即乘積之相對誤差等於乘數中之最大相對誤差。

例十：考慮下列n個數值x_i之和s：

$$s = \sum_{i=1}^{n} x_i \ , \qquad \delta_s = \sum_{i=1}^{n} \delta_{x_i} \ , \qquad |\delta_s| \le \sum_{i=1}^{n} |\delta_{x_i}| \tag{11.44}$$

$$\sigma^2_{\delta_s} = \sum_{i=1}^{n} \sigma^2_{\delta_{x_i}} \doteq m\, \sigma^2_{\delta_{x^*}} \ , \qquad \sigma_{\delta_s} \doteq \sqrt{m}\, \sigma_{\delta_{x^*}} \tag{11.45}$$

由上節之誤差傳播分析，二數值相加或相減，其絕對誤差為二數者之和或差。但誤差有正有負，故絕對誤差之絕對值有時相加有時相減，與加或減無對應關係。$|\delta_s|$之上限為$|\delta_{x_i}|$之和。但該值常太大而不切實際。在此亦用統計方法計算絕對誤差的標準偏差如式(11.45)。式中x^*表示絕對誤差較大之數值，m為絕對誤差與x^*者相當之數值之總數，該值介於n與1之間。如果n個值之絕對誤差大小相當，則$m = n$，大約100個誤差值相同的數值相加或減之和或差之絕對誤差約會增大十倍。如果n個值中只有一個值的絕對誤差特別大，則$m = 1$，即和之絕對誤差等於加數中之最大絕對誤差。

在本例中並未考慮每做一次加後部分和存放時引進之捨入誤差$\epsilon_M = \frac{1}{2} B^{-t+1}$。如果運算過程中部分和之有效位數已經比計算機之有效位數少

時，部分和之捨入誤差即可忽略不計。一般當n很大時，部分和因誤差累積，其有效位數即很可能小於計算機之有效位數。如果計算機有效位數等於部分和之有效位數，則前述m值最多改為n。並請參考例十一對此情形所做之分析。

例十一：考慮下列4個數值x_i之和s：

$$\begin{aligned} s &= ((x_1+x_2)+x_3)+x_4 \\ \delta_s &= \delta_{x_1}+\delta_{x_2}+(x_1+x_2)\epsilon_M+\delta_{x_3}+(x_1+x_2+x_3)\epsilon_M+\delta_{x_4}+(x_1+x_2+x_3+x_4)\epsilon_M \\ &= \sum_{i=1}^{n}\delta_{x_i}+(3x_1+3x_2+2x_3+x_4)\epsilon_M \end{aligned} \tag{11.46}$$

本例即考慮當加數誤差很小，部分和之有效位數等於計算機之有效位數，而針對計算機之捨入誤差加以分析。現考慮下列二種計算方式：

(1)將加數先依由大到小之順序排列再相加，即排成$x_1 > x_2 > x_3 > x_4$。
(2)將加數先依由小到大之順序排列再相加，即排成$x_1 < x_2 < x_3 < x_4$。

設δ_{x_i}很小可以忽略，則由式(11.46)之結果，注意式中各加數x_i之係數先加者較大而後加者較小。因此方式(1)先加之大數x_1乘較大之係數3，比方式(2)後加之大數x_4乘較小之係數1為大。故方式(1)之和數之誤差比方式(2)者大。可知運算值之大小順序會影響誤差之大小。這種現象也可由下列符號式說明：方式(1)之左式一開始二數相加即有捨入誤差，且其誤差及後繼所有每次加算之部分和之捨入誤差都會一直保留給和數s，相當於最先加入之大數x_1之係數為$n-1=3$。方式(2)之右式一開始相加也有捨入誤差，但其誤差不會留給和數s。或從另一角度看：和數之有效位數範圍內之數值，在前三項較小加數之部分和中並無捨入誤差產生。換言之，其捨入誤差遠小於和數之最後一位有效數值，因此只有最後一次捨入誤差留給和數s，相當於最後加入之大數x_4之係數為1。

$$\begin{array}{rcc} x_1: & ccccccc & ccccccc \\ x_2: & ccccccc & ccccccc \\ x_3: & ccccccc & ccccccc \\ x_4: & ccccccc & ccccccc \\ \hline s: & ccccccc & ccccccc \end{array}$$

由以上分析可知加數由小漸大較由大漸小可以減少計算機之捨入誤差。但此做法並不容易實行，一種簡單做法是部分和採用倍精度。則計算機

對部分和造成之捨入誤差就會遠比加數之絕對誤差小(僅約10^{-8}倍)，因此即可不計此捨入誤差，而可不受加數大小順序之影響。事實上最後之和數將不含部分和之捨入誤差，其所有誤差均得自加數之誤差(含單精度之捨入誤差)與最後結果存為單精度之捨入誤差。如果所有加數已經用倍精度，則因有效位數已夠多，大小順序的影響即可不必太計較。

注意例九之積，與例十之和，二者之論點十分相似，二者之誤差均為乘數或加數者之和，故其標準偏差之計算及討論相同，所不同者為一個用相對誤差一個用絕對誤差。也正因為乘積用相對誤差，計算機之有限之有效位數產生之相對誤差約為固定值，而積數不會對相對誤差做嚴重之放大，故乘算較不會造成相對誤差過大之問題。反之，和數用絕對誤差就不如積數那麼幸運，因為和數會比加數小，甚至為零，但絕對誤差只增不減，因此和數之相對誤差可以變成無窮之大。以相對誤差而言，和數之相對誤差經常可能較加數之相對誤差大很多倍，有如相對誤差被放大。本章所有範例之過大誤差幾乎都是由和算所造成。事實既然如此，利用計算機做數值計算，必然會由捨入誤差引進一定量之相對誤差，而和算又偏偏對相對誤差具有放大作用，只好步步為營，擅自小心，不要誤入陷井，相信即能避凶趨吉。

例十二：考慮下列複數(complex number)之求絕對值與相除運算：

$$|a+bi| = \sqrt{a^2+b^2} = a\sqrt{1+(b/a)^2} \tag{11.47}$$

$$\frac{c+di}{a+bi} = \frac{ac+bd+(ad-bc)i}{a^2+b^2} = \frac{c+(b/a)d+(d-(b/a)c)i}{a(1+(b/a)^2)} \tag{11.48}$$

本例應該不屬於誤差分析之討論範圍，不過有必要提一下，因此順便在此討論。上列問題不在計算機之有限之有效位數之捨入誤差；而是在計算機之最大數值上之限制。以下討論假設$|a|>|b|$。如果a值很大，其平方值超過計算機之最大值，則用第一式計算a^2時會造成溢位 (overflow)之錯誤。如果改用第二式計算就不會發生此溢位錯誤。其他類似情形如式(11.48)，亦可也應採用類似方式處理。不過這些處理方式只有在實數與虛數分開儲存，自行寫計算式做有關之運算時才需要。符傳程式之內部複數運算應已考慮及此，且其內部運算可將基底指數扣除二者之平均值後再運算，即可避免溢位錯誤。最後再經指數調整即得運算結果，應較上述做法簡單。

習題

1. 試寫一程式以下列級數計算$\sin 1830^o$之值。分別以單精度及倍精度計算，並列印部分和與每一項之值。

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \cdots$$

2. 試以部分積分法（integration by parts），証明下列積分有其後所列之遞迴關係，並計算$n = 0, 1, 2, \cdots, 10$之值。必要時可用$I_o = 1 - e^{-1}$。

$$I_n = \int_0^1 x^n e^{x-1}\, dx\ , \quad n = 0, 1, 2, \cdots$$
$$I_n = 1 - nI_{n-1}$$

3. 試分析下列函數之二種算法之誤差，何者較好？

$$\begin{aligned} f_1(x) &= ((a*x)*x)*x + (b*x)*x + c*x + d \\ f_2(x) &= ((a*x+b)*x+c)*x + d \end{aligned}$$

4. 下列聯立方程式是否會有誤差過大之問題？試用程式求解並比較單精度與倍精度之結果。若其中之係數與常數用四位有效位數讀入時，其結果又如何？

$$\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 25/12 \\ 77/60 \\ 19/20 \\ 319/420 \end{Bmatrix}$$

第十二章

常微分方程之數值解法

12.1 前言

微分方程式以自變數之多寡區分二大類：只有一個自變數的為常微分方程；二個或二個以上自變數的為偏微分方程；二者均為工程上常會遇上之問題。本章僅介紹常微分方程之數值解法；偏微分方程之數值解法則另以一章介紹。工程應用上常微分方程式之自變數以時間t為最常見，因此本章之各項說明均將以t做為自變數。一般常微分方程式之解析解均含一定數目的積分常數，這些常數須由等數目的已知條件來決定，由於積分常數的數目與該因變數在微分方程式中之最高微分階數相等，最常見的條件為$t = 0$時（或$t = t_o$時）之自變數值及其比最高微分階數小之各階微分值，此種條件稱為初始條件。例如2階微分方程式之初始條件為$y(t_o) = y_o,\ y'(t_o) = v_o$。這些條件亦可分散在不同之$t$處，若條件在不只一個$t$處，此種條件稱為邊界條件。注意在不同$t$處可採用相同微分階數之值為已知條件。例如2階微分方程式之邊界條件可為$y(t_o) = y_o,\ y(t_1) = y_1$，或為$y'(t_o) = v_o,\ y'(t_1) = v_1$。初始條件為常微分方程式中最常用也是最基本的條件，因此本章以探討初始條件之數值解為主，再說明如何以初始條件之解法求邊界條件之解。並略述以差分式求解邊界值問題之方法。

由於高階常微分方程式均可轉化為一階常微分方程式，而多個因變數之解法與一個因變數之解法類似，因此有關之公式推導均以一個因變數之一階常微分方程式為之。

12.2 高階轉化為一階之方法

一般常微分方程式之數值解法均以聯立一階常微分方程式處理，對於高階之常微分方程式均可仿照下述方法將其轉化為聯立一階常微分方程式。以因變數$y(t)$之下列三階微分方程式與初始條件為例：

$$A(t)\frac{d^3y}{dt^3}+B(t)\frac{d^2y}{dt^2}+C(t)\frac{dy}{dt}+D(t)y=F(t) \tag{12.1}$$

$$y(t_o)=y_o,\quad y'(t_o)=v_o,\quad y''(t_o)=a_o \tag{12.2}$$

若令$y^{(1)}(t)=y,\ y^{(2)}(t)=\frac{dy}{dt},\ y^{(3)}(t)=\frac{d^2y}{dt^2}$，上式即可轉化為三個因變數$y^{(1)}(t),\ y^{(2)}(t),\ y^{(3)}(t)$之下列三式聯立一階常微分方程式與初始條件。注意表面上$y^{(1)},\ y^{(2)},\ y^{(3)}$係視為三個獨立之因變數，其間之實質微分關係則係利用式(12.3)與式(12.4)之條件達成。

$$\frac{dy^{(1)}}{dt}=y^{(2)} \tag{12.3}$$

$$\frac{dy^{(2)}}{dt}=y^{(3)} \tag{12.4}$$

$$\frac{dy^{(3)}}{dt}=-\frac{B(t)}{A(t)}y^{(3)}-\frac{C(t)}{A(t)}y^{(2)}-\frac{D(t)}{A(t)}y^{(1)}+\frac{F(t)}{A(t)} \tag{12.5}$$

$$y^{(1)}(t_o)=y_o,\quad y^{(2)}(t_o)=v_o,\quad y^{(3)}(t_o)=a_o \tag{12.6}$$

再以式(12.7)之n自由度結構動力 之n式聯立二階常微分方程式為例，可轉化為式(12.8)之$2n$式聯立一階常微分方程式。其一階解法與二階直接解法另詳第12.6節。

$$[M]\{y''\}+[C]\{y'\}+[K]\{y\}=\{p\} \tag{12.7}$$

$$\begin{Bmatrix}\{y'\}\\ \{v'\}\end{Bmatrix}=\begin{bmatrix}[0] & [I]\\ -[M]^{-1}[K] & -[M]^{-1}[C]\end{bmatrix}\begin{Bmatrix}\{y\}\\ \{v\}\end{Bmatrix}+\begin{Bmatrix}\{0\}\\ [M]^{-1}\{p\}\end{Bmatrix} \tag{12.8}$$

由以上討論可知高階常微分方程式均可化為如下之標準n式聯立一階常微分方程式：

$$\frac{dy^{(1)}}{dt}=f^{(1)}(y^{(1)},y^{(2)},\cdots,y^{(n)},t)$$

$$\frac{dy^{(2)}}{dt} = f^{(2)}(y^{(1)}, y^{(2)}, \cdots, y^{(n)}, t) \tag{12.9}$$

$$\cdots$$

$$\frac{dy^{(n)}}{dt} = f^{(n)}(y^{(1)}, y^{(2)}, \cdots, y^{(n)}, t)$$

$$y^{(1)}(0) = y_o^{(1)}, \quad y^{(2)}(0) = y_o^{(2)}, \quad \cdots, \quad y^{(n)}(0) = y_o^{(n)} \tag{12.10}$$

12.3 各種數值解法簡介與特性比較

為簡化符號與容易說明，以下將以一個因變數之一階常微分方程式探討常微分方程式之數值解法。事實上這些方法很容易延用於多式聯立之情形。

$$\frac{dy}{dt} = f(y, t) \tag{12.11}$$

$$y(0) = y_o \tag{12.12}$$

數值解主要由$t = 0$時之已知初始值$y(0) = y_o$依$j = 1, 2, \cdots$之次序計算$t_j = j \cdot \Delta t = j \cdot h$時各點之因變數值$y(t_j) = y_j$。以下討論係假設已經計算至$y_j$而欲計算下一點$y_{j+1}$之情形為例。常微分方程之數值解法一般分為二種類型：一為郎吉庫達類型(Runge–Kutta type)，如四階精度之式(12.13)；一為亞當氏類型(Adams type)，該型又分為開顯式(Open formula)與閉隱式(Closed formula)二種，如四階精度之式(12.14)一般稱為亞當貝士福滋(Adams–Bashforth)式屬開顯式， 式(12.15)一般稱為亞當莫爾頓(Adams–Moulton)式屬閉隱式。 郎吉庫達類型之特點為：計算y_{j+1}時只用到$t \geq t_j$時之$y(t)$與$f(y, t)$。亞當氏類型之特點為： 開顯式計算y_{j+1}時會用到$t \leq t_j$時之$y(t)$與$f(y, t)$。閉隱式則還需用到$f_{j+1} = f(y_{j+1}, t_{j+1})$之值，但式中等號右邊用到之$y_{j+1}$為待求之未知值，因此須以反覆試算方式求$y_{j+1}$之值，即式(12.15)等號右邊之$y_{j+1}$用第$k$次近似值$y_{j+1}^{(k)}$代入，所得等號左邊之$y_{j+1}$為其第$k+1$次近似值$y_{j+1}^{(k+1)}$。閉隱式須要反覆試算為其缺點，但所得結果較開隱式準確，實用的有效做法是以開顯式所得之y_{j+1}做為閉隱式之起始試算值$y_{j+1}^{(o)}$，如此可減少閉隱式之反覆試算次數，甚至可只用閉隱式計算一次。因此開顯式又稱預測式(Predictor)，閉隱式又稱改正式(Corrector)。有時為了能更精確地預測起始值，常將開顯式之預測值$y_{j+1}^{(o)}$再經式(12.16)之修正式計算修正值$\tilde{y}_{j+1}^{(o)}$，方做為閉隱式之起始試算值

。修正式係假設閉隱式與開顯式所得值之差於t_{j+1}與t_j處相等，即假設$y_{j+1}-y_{j+1}^{(o)}=y_j-y_j^{(o)}$而得。（一般修正式用$\tilde{y}_{j+1}^{(o)}=y_{j+1}^{(o)}+\frac{251}{270}(y_j-y_j^{(o)})$係根據式(12.87)而得。但改正式之收斂值為$y_{j+1}^{(\infty)}$，並非$y_{j+1}$。故式(12.16)之修正值應更接近收斂值$y_{j+1}^{(\infty)}$，而使收斂較快。所幸收斂值不受修正值之影響，故採用何式做修正均可）注意亞當氏類型在最開始數點因其所需用到之$y_{-1}, y_{-2}, \cdots$各值為未知而無法使用，通常最開始數點可利用郎吉庫達類型計算。郎吉庫達類型不必用到$t<t_j$以前之$y(t)$因此一開始即可使用，為其最大優點。因此亦稱此類型為可自行起始。 又因其未用到$t<t_j$以前之$y(t)$可不必受限於先前使用之步長h而隨時可以依照精度之要求及當時之函數變化情況調整適當之步長。因此較亞當氏類型適合於函數變化較大之常微分方程式。但郎吉庫達類型需另外計算其所需之$f_j^a, {}_j^b, \cdots$等值，並不像亞當氏類型大部分採用已有之前數點之值，而使得郎吉庫達類型計算效率略差。

$$\begin{aligned} y_{j+1} &= y_j+\frac{h}{6}(f_j+2f_j^a+2f_j^b+f_j^c) \qquad (12.13)\\ f_j &= f(y_j,\ t_j)\\ f_j^a &= f(y_j+\frac{1}{2}f_j\,h,\ t_j+\frac{1}{2}h)\\ f_j^b &= f(y_j+\frac{1}{2}f_j^a h,\ t_j+\frac{1}{2}h)\\ f_j^c &= f(y_j+f_j^b h,\ t_j+h) \end{aligned}$$

$$y_{j+1} = y_j+h(\frac{55}{24}f_j-\frac{59}{24}f_{j-1}+\frac{37}{24}f_{j-2}-\frac{9}{24}f_{j-3}) \qquad (12.14)$$

$$y_{j+1} = y_j+h(\frac{9}{24}f_{j+1}+\frac{19}{24}f_j-\frac{5}{24}f_{j-1}+\frac{1}{24}f_{j-2}) \qquad (12.15)$$

$$\tilde{y}_{j+1}^{(o)} = y_{j+1}^{(o)}+(y_j-y_j^{(o)}) \qquad (12.16)$$

12.4 各種數值解法之推導

亞當氏類型之推導較簡單先行介紹。首先將y_{j+1}對t_j點用泰勒級數展開如式(12.17)，再引用式(12.11)之關係即$y_j'=f_j, y_j''=f_j', y_j'''=f_j'', \cdots$等即得式(12.18)。

$$y_{j+1}=y_j+hy_j'+\frac{h^2}{2!}y_j''+\frac{h^3}{3!}y_j'''+\frac{h^4}{4!}y_j''''+\frac{h^5}{5!}y_j^{(v)}+\frac{h^6}{6!}y_j^{(vi)}+\cdots \qquad (12.17)$$

$$= y_j + hf_j + \frac{h^2}{2!}f'_j + \frac{h^3}{3!}f''_j + \frac{h^4}{4!}f'''_j + \frac{h^5}{5!}f''''_j + \frac{h^6}{6!}f^{(v)}_j + \cdots \qquad (12.18)$$

將 f_j 之各階微分用下列反向差分代入即可得式 (12.19) 之 亞當氏類型之各階開顯式。（以下各反向差分式之証明：將式中之各 f_{j-k} 以其泰勒式 $f_j - khf'_j + \frac{k^2h^2}{2!}f''_j - \frac{k^3h^3}{3!}f'''_j + \cdots$ 代入即可証得。）

$$\begin{aligned}
f'_j &= \frac{f_j - f_{j-1}}{h} + \frac{h}{2!}f''_j - \frac{h^2}{3!}f'''_j + \frac{h^3}{4!}f''''_j - \frac{h^4}{5!}f^{(v)}_j + \cdots \\
f''_j &= \frac{f_j - 2f_{j-1} + f_{j-2}}{h^2} + hf'''_j - \frac{7h^2}{12}f''''_j + \frac{h^3}{4}f^{(v)}_j - \cdots \\
f'''_j &= \frac{f_j - 3f_{j-1} + 3f_{j-2} - f_{j-3}}{h^3} + \frac{3h}{2}f''''_j - \frac{5h^2}{4}f^{(v)}_j + \cdots \\
f''''_j &= \frac{f_j - 4f_{j-1} + 6f_{j-2} - 4f_{j-3} + f_{j-4}}{h^4} + 2hf^{(v)}_j - \cdots
\end{aligned}$$

$$\begin{aligned}
& y_{j+1} - y_j \qquad (12.19) \\
&= h(\frac{3}{2}f_j - \frac{1}{2}f_{j-1}) + \frac{5h^3}{12}f''_j - \frac{h^4}{24}f'''_j + \frac{7h^5}{240}f''''_j - \frac{h^6}{360}f^{(v)}_j + \cdots \\
&= h(\frac{23}{12}f_j - \frac{16}{12}f_{j-1} + \frac{5}{12}f_{j-2}) + \frac{9h^4}{24}f'''_j - \frac{77h^5}{360}f''''_j + \frac{73h^6}{720}f^{(v)}_j - \cdots \\
&= h(\frac{55}{24}f_j - \frac{59}{24}f_{j-1} + \frac{37}{24}f_{j-2} - \frac{9}{24}f_{j-3}) + \frac{251h^5}{720}f''''_j - \frac{529h^6}{1440}f^{(v)}_j + \cdots \\
&= h(\frac{1901}{720}f_j - \frac{2774}{720}f_{j-1} + \frac{2616}{720}f_{j-2} - \frac{1274}{720}f_{j-3} + \frac{251}{720}f_{j-4}) + \frac{475h^6}{1440}f^{(v)}_j - \cdots
\end{aligned}$$

閉隱式之推導則將 y_j 對 t_{j+1} 點用泰勒級數展開，移項後如式 (12.20)，再引用式 (12.11) 之關係即得式 (12.21)。

$$y_{j+1} = y_j + hy'_{j+1} - \frac{h^2}{2!}y''_{j+1} + \frac{h^3}{3!}y'''_{j+1} - \frac{h^4}{4!}y''''_{j+1} + \frac{h^5}{5!}y^{(v)}_{j+1} - \frac{h^6}{6!}y^{(vi)}_{j+1} + \cdots \qquad (12.20)$$

$$= y_j + hf_{j+1} - \frac{h^2}{2!}f'_{j+1} + \frac{h^3}{3!}f''_{j+1} - \frac{h^4}{4!}f'''_{j+1} + \frac{h^5}{5!}f''''_{j+1} - \frac{h^6}{6!}f^{(v)}_{j+1} + \cdots \qquad (12.21)$$

將 f_{j+1} 之各階微分用下列反向差分代入即可得式 (12.22) 之各階閉隱式。

$$\begin{aligned}
f'_{j+1} &= \frac{f_{j+1} - f_j}{h} + \frac{h}{2!}f''_{j+1} - \frac{h^2}{3!}f'''_{j+1} + \frac{h^3}{4!}f''''_{j+1} - \frac{h^4}{5!}f^{(v)}_{j+1} + \cdots \\
f''_{j+1} &= \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2} + hf'''_{j+1} - \frac{7h^2}{12}f''''_{j+1} + \frac{h^3}{4}f^{(v)}_{j+1} - \cdots \\
f'''_{j+1} &= \frac{f_{j+1} - 3f_j + 3f_{j-1} - f_{j-2}}{h^3} + \frac{3h}{2}f''''_{j+1} - \frac{5h^2}{4}f^{(v)}_{j+1} + \cdots \\
f''''_{j+1} &= \frac{f_{j+1} - 4f_j + 6f_{j-1} - 4f_{j-2} + f_{j-3}}{h^4} + 2hf^{(v)}_{j+1} - \cdots
\end{aligned}$$

$$
\begin{aligned}
& y_{j+1}-y_j \qquad (12.22)\\
&= h(\frac{1}{2}f_{j+1}+\frac{1}{2}f_j)-\frac{h^3}{12}f''_{j+1}+\frac{h^4}{24}f'''_{j+1}-\frac{h^5}{80}f''''_{j+1}+\frac{h^6}{360}f^{(v)}_{j+1}-\cdots\\
&= h(\frac{5}{12}f_{j+1}+\frac{8}{12}f_j-\frac{1}{12}f_{j-1})-\frac{h^4}{24}f'''_{j+1}+\frac{13h^5}{360}f''''_{j+1}-\frac{13h^6}{720}f^{(v)}_{j+1}+\cdots\\
&= h(\frac{9}{24}f_{j+1}+\frac{19}{24}f_j-\frac{5}{24}f_{j-1}+\frac{1}{24}f_{j-2})-\frac{19h^5}{720}f''''_{j+1}+\frac{49h^6}{1440}f^{(v)}_{j+1}-\cdots\\
&= h(\frac{251}{720}f_{j+1}+\frac{646}{720}f_j-\frac{264}{720}f_{j-1}+\frac{106}{720}f_{j-2}-\frac{19}{720}f_{j-3})-\frac{27h^6}{1440}f^{(v)}_{j+1}+\cdots
\end{aligned}
$$

亞當氏類型公式亦可由下列積分式導得。若其中之$F(t)$為通過下列n點$(t_{j-n+1},f_{j-n+1}),\ (t_{j-n+2},f_{j-n+2}),\cdots,(t_{j-1},f_{j-1}),\ (t_j,f_j)$之$n-1$次多項式，可得$n$階開顯式。若$F(t)$改為通過下列$n$點$(t_{j-n+2},f_{j-n+2}),\ (t_{j-n+3},f_{j-n+3}),\cdots,(t_j,f_j),\ (t_{j+1},f_{j+1})$之$n-1$次多項式，則得$n$階閉隱式。

$$y_{j+1}=y_j+\int_{t_j}^{t_{j+1}}F(t)\,dt \qquad (12.23)$$

上式之積分下限亦可改為t_{j-k}，當然y_j亦應配合改為y_{j-k}，以下之三階米爾內(Milne)預測式 (12.24) 與改正式 (12.25)即由此導出。該法誤差項較小但為不穩定 (Unstable)為其缺點。

$$y_{j+1} = y_{j-3}+\frac{4h}{3}(2f_j-f_{j-1}+2f_{j-2})+\frac{14h^5}{45}y^{(v)}(\xi) \qquad (12.24)$$

$$y_{j+1} = y_{j-1}+\frac{h}{3}(f_{j+1}+4f_j+f_{j-1})-\frac{h^5}{90}y^{(v)}(\eta_1) \qquad (12.25)$$

$$y_{j+1} = \frac{1}{8}(9y_j-y_{j-2})+\frac{3h}{8}(f_{j+1}+2f_j-f_{j-1})-\frac{h^5}{40}y^{(v)}(\eta_2) \qquad (12.26)$$

漢民(Hamming)則以下述待定係數法推得另一改正式(12.26)以取代米爾內不穩定之改正式，該式之誤差比米爾內公式大，但比亞當氏同階之閉隱式小，因此亦頗常被採用。

$$y_{j+1}=\sum_{k=0}^{p}(A_ky_{j-k}+hB_kf_{j-k})+hB_{-1}f_{j+1} \qquad (12.27)$$

上式中若不含f_{j+1}項，即$B_{-1}=0$，則屬開顯式；若含f_{j+1}，即$B_{-1}\neq 0$，則屬閉隱式。欲求p階之公式可將下列之y與f代入上式，並令各任意常

數c_m之係數為零，即得式(12.30)至式(12.33)，如此可使上式能正確算得y為t之p次多項式時之解。

$$y(t)=\sum_{m=0}^{p}c_m(t-t_j)^m,\qquad f(t)=y'(t)=\sum_{m=0}^{p}c_m m(t-t_j)^{m-1} \tag{12.28}$$

$$y_{j-k}=\sum_{m=0}^{p}c_m(-k)^m h^m,\qquad f_{j-k}=\sum_{m=0}^{p}c_m m(-k)^{m-1}h^{m-1} \tag{12.29}$$

$$1 = \sum_{k=0}^{p}A_k \tag{12.30}$$

$$-1 = \sum_{k=0}^{p}kA_k-\sum_{k=-1}^{p}B_k \tag{12.31}$$

$$1 = \sum_{k=0}^{p}k^2A_k-2\sum_{k=-1}^{p}kB_k \tag{12.32}$$

$$\cdots$$

$$(-1)^p = \sum_{k=0}^{p}k^pA_k-p\sum_{k=-1}^{p}k^{p-1}B_k \tag{12.33}$$

上式之條件比待定係數少，大部分係數可設為零，除上述條件外，對於閉隱式還須符合穩定之條件。穩定條件將於第12.5節說明。

郎吉庫達類型之推導較繁瑣：在此僅以常用之四階公式為例。首先將式(12.34)中微分項以式(12.35)至式(12.37)表示。式中f'為對t之全微分，f_t為對t之偏微分，f_y為對y之偏微分。以下各等式右邊各函數或各階偏微分均表示t_j時之值，為簡化符號均省略下標j。

$$y_{j+1}=y_j+hf_j+\frac{h^2}{2!}f'_j+\frac{h^3}{3!}f''_j+\frac{h^4}{4!}f'''_j+O(h^5) \tag{12.34}$$

$$f'_j = f_t+ff_y \tag{12.35}$$

$$\begin{aligned} f''_j &= (f_{tt}+ff_{ty})+(f_tf_y+ff_y^2+ff_{yt}+f^2f_{yy}) \\ &= (f_{tt}+2ff_{ty}+f^2f_{yy})+(f_tf_y+ff_y^2) \end{aligned} \tag{12.36}$$

$$\begin{aligned} f'''_j &= (f_{ttt}+ff_{tty})+2(f_tf_{ty}+ff_yf_{ty}+ff_{tty}+f^2f_{tyy}) \\ &\quad +(2ff_tf_{yy}+2f^2f_yf_{yy}+f^2f_{tyy}+f^3f_{yyy}) \\ &\quad +(f_{tt}f_y+ff_{ty}f_y+f_tf_{ty}+ff_tf_{yy}) \\ &\quad +(f_tf_y^2+ff_y^3+2ff_yf_{ty}+2f^2f_yf_{yy}) \\ &= (f_{ttt}+3ff_{tty}+3f^2f_{tyy}+f^3f_{yyy})+(f_{tt}+2ff_{ty}+f^2f_{yy})f_y \\ &\quad +3(f_tf_{ty}+ff_yf_{ty}+ff_tf_{yy}+f^2f_yf_{yy})+(f_tf_y^2+ff_y^3) \end{aligned} \tag{12.37}$$

將式(12.40)至式(12.42)中之$f_j^a,\ f_j^b,\ f_j^c$亦以式(12.43)至式(12.45)之泰勒式表示。為簡化符號令$\beta_{23}=\beta_2+\beta_3$，$\beta_{456}=\beta_4+\beta_5+\beta_6$。

$$y_{j+1} = y_j + h(\gamma_o f_j + \gamma_1 f_j^a + \gamma_2 f_j^b + \gamma_3 f_j^c) \tag{12.38}$$

$$f_j = f(y_j,\ t_j) \tag{12.39}$$
$$f_j^a = f(y_j + \beta_1 f_j h,\ t_j + \alpha_1 h) \tag{12.40}$$
$$f_j^b = f(y_j + \beta_2 f_j h + \beta_3 f_j^a h,\ t_j + \alpha_2 h) \tag{12.41}$$
$$f_j^c = f(y_j + \beta_4 f_j h + \beta_5 f_j^a h + \beta_6 f_j^b h,\ t_j + \alpha_3 h) \tag{12.42}$$

$$f_j^a = f^a(0) + hf^{a\prime}(0) + \frac{h^2}{2!}f^{a\prime\prime}(0) + \frac{h^3}{3!}f^{a\prime\prime\prime}(0) + O(h^4) \tag{12.43}$$
$$\begin{aligned}
f^a(0) &= f \\
f^{a\prime}(0) &= \alpha_1 f_t + \beta_1 f f_y \\
f^{a\prime\prime}(0) &= \alpha_1^2 f_{tt} + 2\alpha_1\beta_1 f f_{ty} + \beta_1^2 f^2 f_{yy} \\
f^{a\prime\prime\prime}(0) &= \alpha_1^3 f_{ttt} + 3\alpha_1^2\beta_1 f f_{tty} + 3\alpha_1\beta_1^2 f^2 f_{tyy} + \beta_1^3 f^3 f_{yyy}
\end{aligned}$$

$$f_j^b = f^b(0) + hf^{b\prime}(0) + \frac{h^2}{2!}f^{b\prime\prime}(0) + \frac{h^3}{3!}f^{b\prime\prime\prime}(0) + O(h^4) \tag{12.44}$$
$$\begin{aligned}
f^b(0) &= f \\
f^{b\prime}(0) &= \alpha_2 f_t + \beta_{23} f f_y \\
f^{b\prime\prime}(0) &= \alpha_2^2 f_{tt} + 2\alpha_2\beta_{23} f f_{ty} + \beta_{23}^2 f^2 f_{yy} + 2f_y\beta_3(\alpha_1 f_t + \beta_1 f f_y) \\
f^{b\prime\prime\prime}(0) &= \alpha_2^3 f_{ttt} + 3\alpha_2^2\beta_{23} f f_{tty} + 3\alpha_2\beta_{23}^2 f^2 f_{tyy} + \beta_{23}^3 f^3 f_{yyy} \\
&\quad + 6(\alpha_2 f_{ty} + \beta_{23} f f_{yy})\beta_3(\alpha_1 f_t + \beta_1 f f_y) \\
&\quad + 3f_y\beta_3(\alpha_1^2 f_{tt} + 2\alpha_1\beta_1 f f_{ty} + \beta_1^2 f^2 f_{yy})
\end{aligned}$$

$$f_j^c(h) = f^c(0) + hf^{c\prime}(0) + \frac{h^2}{2!}f^{c\prime\prime}(0) + \frac{h^3}{3!}f^{c\prime\prime\prime}(0) + O(h^4) \tag{12.45}$$
$$\begin{aligned}
f^c(0) &= f \\
f^{c\prime}(0) &= \alpha_3 f_t + \beta_{456} f f_y \\
f^{c\prime\prime}(0) &= \alpha_3^2 f_{tt} + 2\alpha_3\beta_{456} f f_{ty} + \beta_{456}^2 f^2 f_{yy} \\
&\quad + 2f_y(\beta_5(\alpha_1 f_t + \beta_1 f f_y) + \beta_6(\alpha_2 f_t + \beta_{23} f f_y)) \\
f^{c\prime\prime\prime}(0) &= \alpha_3^3 f_{ttt} + 3\alpha_3^2\beta_{456} f f_{tty} + 3\alpha_3\beta_{456}^2 f^2 f_{tyy} + \beta_{456}^3 f^3 f_{yyy}
\end{aligned}$$

$$
\begin{aligned}
&+ 6(\alpha_3 f_{ty} + \beta_{456} f f_{yy})(\beta_5(\alpha_1 f_t + \beta_1 f f_y) + \beta_6(\alpha_2 f_t + \beta_{23} f f_y)) \\
&+ 3f_y(\beta_5(\alpha_1^2 f_{tt} + 2\alpha_1\beta_1 f f_{ty} + \beta_1^2 f^2 f_{yy}) \\
&\quad + \beta_6(\alpha_2^2 f_{tt} + 2\alpha_2\beta_{23} f f_{ty} + \beta_{23}^2 f^2 f_{yy}))
\end{aligned}
$$

比較式(12.34)與式(12.38)中各項係數可得下列條件：其中前三個條件可直接看出。

$$
\begin{aligned}
\alpha_1 &= \beta_1 \\
\alpha_2 &= \beta_2 + \beta_3 \\
\alpha_3 &= \beta_4 + \beta_5 + \beta_6 \\
f \quad &: \quad 1 = \gamma_o + \gamma_1 + \gamma_2 + \gamma_3 \\
f_t + f f_y \quad &: \quad \frac{1}{2} = \gamma_1\alpha_1 + \gamma_2\alpha_2 + \gamma_3\alpha_3 \\
f_{tt} + 2f f_{ty} + f^2 f_{yy} \quad &: \quad \frac{1}{3} = \gamma_1\alpha_1^2 + \gamma_2\alpha_2^2 + \gamma_3\alpha_3^2 \\
f_{ttt} + 3f f_{tty} + 3f^2 f_{tyy} + f^3 f_{yyy} \quad &: \quad \frac{1}{4} = \gamma_1\alpha_1^3 + \gamma_2\alpha_2^3 + \gamma_3\alpha_3^3 \\
f_t f_y + f f_y^2 \quad &: \quad \frac{1}{6} = \gamma_2\alpha_1\beta_3 + \gamma_3(\alpha_1\beta_5 + \alpha_2\beta_6) \\
(f_{tt} + 2f f_{ty} + f^2 f_{yy}) f_y \quad &: \quad \frac{1}{12} = \gamma_2\alpha_1^2\beta_3 + \gamma_3(\alpha_1^2\beta_5 + \alpha_2^2\beta_6) \\
f_t f_{ty} + f f_y f_{ty} + f f_t f_{yy} + f^2 f_y f_{yy} \quad &: \quad \frac{1}{8} = \gamma_2\alpha_2\alpha_1\beta_3 + \gamma_3\alpha_3(\alpha_1\beta_5 + \alpha_2\beta_6) \\
f_t f_y^2 + f f_y^3 \quad &: \quad \frac{1}{24} = \gamma_3\alpha_1\beta_3\beta_6
\end{aligned}
$$

利用以上11個條件決定13個未知數會有多種選擇：令$\gamma_1 = \gamma_2 = \frac{1}{3}$可解得郎吉氏之係數$\gamma_0 = \gamma_3 = \frac{1}{6}$，$\alpha_1 = \alpha_2 = \beta_1 = \beta_3 = \frac{1}{2}$，$\alpha_3 = \beta_6 = 1$，$\beta_2 = \beta_4 = \beta_5 = 0$。令$\gamma_1 = \gamma_2 = \frac{3}{8}$可解得庫達氏之係數$\gamma_0 = \gamma_3 = \frac{1}{8}$，$\alpha_1 = \beta_1 = \frac{1}{3}$，$\alpha_2 = \frac{2}{3}$，$\alpha_3 = \beta_3 = \beta_4 = -\beta_5 = \beta_6 = 1$，$\beta_2 = -\frac{1}{3}$。

12.5 穩定條件之分析

前面提過米爾內之改正式 (12.25)具不穩定性，以下即以該式為例說明如何分析其不穩定性。因預測式僅用以獲得初始試算值以減少改正式之試算次數，改正式之收斂值與初始試算值無關，故穩定分析均以改正式為之。現考慮下列一階常微分式，即$f(y,t) = ay$，a為常數，其解析解

為$y(t) = y_o e^{at}$。

$$\frac{dy}{dt} = ay, \qquad y(0) = y_o \tag{12.46}$$

令$f_j = ay_j$代入式(12.25)可得式(12.47)或式(12.48)，令$H = \frac{ah}{3}$以簡化符號可得式(12.49)之常係數差分方程式。該差分方程式之通解為$y_j = \sum_{k=1}^{2} C_k \beta_k^j$，因此令$y_j = C\beta^j$代入該式後，除以共同因數$C\beta^{j-1}$，可得式(12.50)為$\beta$之二次多項式。由此式可解得二根$\beta_k$，$k = 1,\ 2$如式(12.51)。若$H << 1$可得二根之近似值如式(12.52)與式(12.53)所示。故得米爾內改正式之解為式(12.54)。利用該解有幾點值得注意：差分式為二階故β有二根，對應亦有二個解。但原微分式為一階故僅有一個解。差分式之二解中有一個會近似原微分式之解，其餘之解為多餘之解稱“寄生解”。由於數值計算難免有誤差使寄生解之係數不為零，如寄生根之絕對值$|\beta| > 1$，則$C\beta^j$將隨j而增大至無窮大，這種現象稱為不穩定。而避免產生不穩定的條件為所有寄生根之絕對值均小於1。米爾內式之寄生解在$a < 0$時即會不穩定，因此不能用米爾內式求解。

$$y_{j+1} = y_{j-1} + \frac{h}{3}(ay_{j+1} + 4ay_j + ay_{j-1}) \tag{12.47}$$

$$(1 - \frac{ah}{3})y_{j+1} - \frac{4ah}{3}y_j - (1 + \frac{ah}{3})y_{j-1} = 0 \tag{12.48}$$

$$(1 - H)y_{j+1} - 4Hy_j - (1 + H)y_{j-1} = 0 \tag{12.49}$$

$$(1 - H)\beta^2 - 4H\beta - (1 + H) = 0 \tag{12.50}$$

$$\beta_k = \frac{2H \pm \sqrt{3H^2 + 1}}{1 - H} = (1 + H + O(H^2))(2H \pm 1 + O(H^2)) \tag{12.51}$$

$$\beta_1 = 1 + 3H + O(H^2) \doteq 1 + ah \tag{12.52}$$

$$\beta_2 = -1 + H + O(H^2) \doteq -1 + \frac{ah}{3} \tag{12.53}$$

$$\begin{aligned} y_j &= C_1(1 + ah)^j + C_2(-1)^j(1 - \frac{ah}{3})^j \\ &= C_1((1 + ah)^{\frac{1}{ah}})^{ahj} + C_2(-1)^j((1 - \frac{ah}{3})^{-\frac{3}{ah}})^{-\frac{ahj}{3}} \\ &\doteq C_1 e^{at_j} + C_2(-1)^j e^{-\frac{at_j}{3}} \end{aligned} \tag{12.54}$$

12.6 二階常係數微分方程之數值解法

本節以式(12.7)之二階常微分式經轉化為式(12.8)後以四階亞當氏公式求解。如$[M]$, $[C]$, $[K]$矩陣不為定值而是因時而異，則可用預測值代入改正式反覆試算求解。當$[M]$, $[C]$, $[K]$為常數時，改正式為$2n$元聯立線性方程式，可直接求解。若將有關公式化簡，可先消去n個未知數而成為n元聯立線性方程式。以下為詳細化簡過程：將四階亞當氏閉隱式寫成下列通式，並將式(12.8)代入可得式(12.56)。將$\{y_{j+1}\}$與$\{v_{j+1}\}$移至等號左邊，並引用式(12.61)即得式(12.57)。再消去後半數方程式中$\{y_{j+1}\}$之係數如式(12.58)所示，並分列成式(12.59)與式(12.60)。式(12.59)即為n元聯立線性方程式，由此可解得$\{v_{j+1}\}$。再代入式(12.60)即可求得$\{y_{j+1}\}$。

$$y_{j+1} = y_j + h(\gamma_1 f_{j+1} + \gamma_2 f_j + \gamma_3 f_{j-1} + \gamma_4 f_{j-2}) \tag{12.55}$$

$$\begin{Bmatrix} \{y_{j+1}\} \\ [M]\{v_{j+1}\} \end{Bmatrix} = \begin{Bmatrix} \{y_j\} \\ [M]\{v_j\} \end{Bmatrix} + \begin{bmatrix} [0] & [I] \\ -[K] & -[C] \end{bmatrix} \begin{Bmatrix} h(\gamma_1\{y_{j+1}\} + \gamma_2\{y_j\} + \gamma_3\{y_{j-1}\} + \gamma_4\{y_{j-2}\}) \\ h(\gamma_1\{v_{j+1}\} + \gamma_2\{v_j\} + \gamma_3\{v_{j-1}\} + \gamma_4\{v_{j-2}\}) \end{Bmatrix} + \begin{Bmatrix} \{0\} \\ h(\gamma_1\{p_{j+1}\} + \gamma_2\{p_j\} + \gamma_3\{p_{j-1}\} + \gamma_4\{p_{j-2}\}) \end{Bmatrix} \tag{12.56}$$

$$\begin{bmatrix} [I] & -\gamma_1 h[I] \\ \gamma_1 h[K] & [M]+\gamma_1 h[C] \end{bmatrix} \begin{Bmatrix} \{y_{j+1}\} \\ \{v_{j+1}\} \end{Bmatrix} = \begin{Bmatrix} \{y_j\} \\ [M]\{v_j\} \end{Bmatrix} + \begin{bmatrix} [0] & [I] \\ -[K] & -[C] \end{bmatrix} \begin{Bmatrix} h(\gamma_2\{y_j\} + \gamma_3\{y_{j-1}\} + \gamma_4\{y_{j-2}\}) \\ h(\gamma_2\{v_j\} + \gamma_3\{v_{j-1}\} + \gamma_4\{v_{j-2}\}) \end{Bmatrix} + \begin{Bmatrix} \{0\} \\ h\{\overline{p}_j\} \end{Bmatrix} \tag{12.57}$$

$$\begin{bmatrix} [I] & -\gamma_1 h[I] \\ [0] & [M]+\gamma_1 h[C]+\gamma_1^2 h^2[K] \end{bmatrix} \begin{Bmatrix} \{y_{j+1}\} \\ \{v_{j+1}\} \end{Bmatrix} = \begin{Bmatrix} \{y_j\} \\ [M]\{v_j\} - \gamma_1 h[K]\{y_j\} \end{Bmatrix} + \begin{bmatrix} [0] & [I] \\ -[K] & -[C]-\gamma_1 h[K] \end{bmatrix} \begin{Bmatrix} h\{\hat{y}_j\} \\ h\{\hat{v}_j\} \end{Bmatrix} + \begin{Bmatrix} \{0\} \\ h\{\overline{p}_j\} \end{Bmatrix} \tag{12.58}$$

$$[\overline{M}]\{v_{j+1}\} = [M]\{v_j\} - h[C]\{\hat{v}_j\} - h[K](\gamma_1\{y_j\} + \{\hat{y}_j\} + \gamma_1 h\{\hat{v}_j\}) + h\{\overline{p}_j\} \tag{12.59}$$

$$\{y_{j+1}\}=\{y_j\}+h(\gamma_1\{v_{j+1}\}+\gamma_2\{v_j\}+\gamma_3\{v_{j-1}\}+\gamma_4\{v_{j-2}\}) \tag{12.60}$$

$$[\overline{M}]=[M]+\gamma_1 h[C]+\gamma_1^2 h^2[K]$$

$$\{\hat{v}_j\}=\gamma_2\{v_j\}+\gamma_3\{v_{j-1}\}+\gamma_4\{v_{j-2}\}$$

$$\{\hat{y}_j\}=\gamma_2\{y_j\}+\gamma_3\{y_{j-1}\}+\gamma_4\{y_{j-2}\}$$

$$\{\overline{p}_j\}=\gamma_1\{p_{j+1}\}+\gamma_2\{p_j\}+\gamma_3\{p_{j-1}\}+\gamma_4\{p_{j-2}\} \tag{12.61}$$

上式亦可改為下式計算二時間段之差值，該式可少做乘$[M]$之運算。

$$h^{-1}[\overline{M}](\{v_{j+1}\}-\{v_j\})=-[C]\{\tilde{v}_j\}-[K](\{\tilde{y}_j\}+\gamma_1 h\{\tilde{v}_j\})+\{\overline{p}_j\} \tag{12.62}$$

$$h^{-1}(\{y_{j+1}\}-\{y_j\})=\gamma_1(\{v_{j+1}\}-\{v_j\})+\{\tilde{v}_j\} \tag{12.63}$$

$$\{\tilde{v}_j\}=\gamma_1\{v_j\}+\gamma_2\{v_j\}+\gamma_3\{v_{j-1}\}+\gamma_4\{v_{j-2}\}$$

$$\{\tilde{y}_j\}=\gamma_1\{y_j\}+\gamma_2\{y_j\}+\gamma_3\{y_{j-1}\}+\gamma_4\{y_{j-2}\}$$

結構動力分析中習慣以式(12.7)之二階微分式直接計算數值解。現以威爾森法(Wilson θ method)為例說明其運算式。該法假設t_j至$t_{j+\theta}=t_j+\theta h$間之加速度為$t=t_j+\eta h$的一次函數，即如式(12.64)。積分可得$\{y'(t)\}$為式(12.65)之$\{y'_{j+\eta}\}$與$\{y(t)\}$為式(12.66)之$\{y_{j+\eta}\}$。令$\eta=\theta$可得式(12.67)與式(12.68)。代入式(12.7)可得$t_{j+\theta}$時之平衡式(12.69)，由該式可解得$\{y''_{j+\theta}\}$，再代入式(12.67)與式(12.68)可得$\{y'_{j+\theta}\}$與$\{y_{j+\theta}\}$。最後代入式(12.64)至式(12.66)求$\eta=1$時之$\{y''_{j+1}\}$，$\{y'_{j+1}\}$與$\{y_{j+1}\}$而完成一個時間段的計算。若威爾森法之$\theta=1$即相當於線性加速度法。但線性加速度法僅為有條件($2h<T_{min}$=結構最小自然週期)穩定。若$\theta\geq 1.37$則為無條件穩定。式(12.67)至式(12.69)亦可用式(12.70)至式(12.72)或式(12.73)至式(12.75)取代。式(12.70)與式(12.71)係由式(12.67)與式(12.68)求得：將$y'_{j+\theta}$移至等號右邊，亦將$y''_{j+\theta}$與$y_{j+\theta}$移至等號左邊，做為二聯立式之未知數即可解得$y''_{j+\theta}$與$y_{j+\theta}$如式(12.70)與式(12.71)。類似做法可得式(12.73)與式(12.74)。

$$\{y''_{j+\eta}\}=\{y''_j\}+\frac{\eta h}{\theta h}(\{y''_{j+\theta}\}-\{y''_j\}) \tag{12.64}$$

$$\{y'_{j+\eta}\}=\{y'_j\}+\eta h\{y''_j\}+\frac{\eta^2 h^2}{2\theta h}(\{y''_{j+\theta}\}-\{y''_j\}) \tag{12.65}$$

$$\{y_{j+\eta}\}=\{y_j\}+\eta h\{y'_j\}+\frac{\eta^2 h^2}{2}\{y''_j\}+\frac{\eta^3 h^3}{6\theta h}(\{y''_{j+\theta}\}-\{y''_j\}) \tag{12.66}$$

$$\{y'_{j+\theta}\}=\{y'_j\}+\theta h\{y''_j\}+\frac{\theta h}{2}(\{y''_{j+\theta}\}-\{y''_j\}) \tag{12.67}$$

$$\{y_{j+\theta}\}=\{y_j\}+\theta h\{y'_j\}+\frac{\theta^2h^2}{2}\{y''_j\}+\frac{\theta^2h^2}{6}(\{y''_{j+\theta}\}-\{y''_j\}) \tag{12.68}$$

$$([M]+\frac{\theta h}{2}[C]+\frac{\theta^2h^2}{6}[K])\{y''_{j+\theta}\}=\{p_{j+\theta}\}-[C](\{y'_j\}+\frac{\theta h}{2}\{y''_j\})$$
$$-[K](\{y_j\}+\theta h\{y'_j\}+\frac{\theta^2h^2}{3}\{y''_j\}) \tag{12.69}$$

$$\{y''_{j+\theta}\}=\frac{2}{\theta h}\{y'_{j+\theta}\}-\frac{2}{\theta h}\{y'_j\}-\{y''_j\} \tag{12.70}$$

$$\{y_{j+\theta}\}=\frac{\theta h}{3}\{y'_{j+\theta}\}+\{y_j\}+\frac{2\theta h}{3}\{y'_j\}+\frac{\theta^2h^2}{6}\{y''_j\} \tag{12.71}$$

$$(\frac{2}{\theta h}[M]+[C]+\frac{\theta h}{3}[K])\{y'_{j+\theta}\}=\{p_{j+\theta}\}+[M](\frac{2}{\theta h}\{y'_j\}+\{y''_j\})$$
$$-[K](\{y_j\}+\frac{2\theta h}{3}\{y'_j\}+\frac{\theta^2h^2}{6}\{y''_j\}) \tag{12.72}$$

$$\{y''_{j+\theta}\}=\frac{6}{\theta^2h^2}\{y_{j+\theta}\}-\frac{6}{\theta^2h^2}\{y_j\}-\frac{6}{\theta h}\{y'_j\}-2\{y''_j\} \tag{12.73}$$

$$\{y'_{j+\theta}\}=\frac{3}{\theta h}\{y_{j+\theta}\}-\frac{3}{\theta h}\{y_j\}-2\{y'_j\}-\frac{\theta h}{2}\{y''_j\} \tag{12.74}$$

$$(\frac{6}{\theta^2h^2}[M]+\frac{3}{\theta h}[C]+[K])\{y_{j+\theta}\}=\{p_{j+\theta}\}+[M](\frac{6}{\theta^2h^2}\{y_j\}+\frac{6}{\theta h}\{y'_j\}+2\{y''_j\})$$
$$+[C](\frac{3}{\theta h}\{y_j\}+2\{y'_j\}+\frac{\theta h}{2}\{y''_j\}) \tag{12.75}$$

式(12.67)至式(12.75)亦可改用式(12.76)至式(12.84)計算二時間段之差值。

$$\{\Delta y'_{j+\theta}\}\equiv\{y'_{j+\theta}\}-\{y'_j\}=\theta h\{y''_j\}+\frac{\theta h}{2}\{\Delta y''_{j+\theta}\} \tag{12.76}$$

$$\{\Delta y_{j+\theta}\}\equiv\{y_{j+\theta}\}-\{y_j\}=\theta hy'_j+\frac{\theta^2h^2}{2}\{y''_j\}+\frac{\theta^2h^2}{6}\{\Delta y''_{j+\theta}\} \tag{12.77}$$

$$([M]+\frac{\theta h}{2}[C]+\frac{\theta^2h^2}{6}[K])\{\Delta y''_{j+\theta}\}=\{\Delta p_{j+\theta}\}-[C](\theta h\{y''_j\})$$
$$-[K](\theta h\{y'_j\}+\frac{\theta^2h^2}{2}\{y''_j\}) \tag{12.78}$$

$$\{\Delta y''_{j+\theta}\}\equiv\{y''_{j+\theta}\}-\{y''_j\}=\frac{2}{\theta h}\{\Delta y'_{j+\theta}\}-2\{y''_j\} \tag{12.79}$$

$$\{\Delta y_{j+\theta}\}\equiv\{y_{j+\theta}\}-\{y_j\}=\frac{\theta h}{3}\{\Delta y'_{j+\theta}\}+\theta hy'_j+\frac{\theta^2h^2}{6}\{y''_j\} \tag{12.80}$$

$$(\frac{2}{\theta h}[M]+[C]+\frac{\theta h}{3}[K])\{\Delta y'_{j+\theta}\}=\{\Delta p_{j+\theta}\}+[M](2\{y''_j\})$$
$$-[K](\theta h\{y'_j\}+\frac{\theta^2h^2}{6}\{y''_j\}) \tag{12.81}$$

$$\{\Delta y''_{j+\theta}\} \equiv \{y''_{j+\theta}\} - \{y''_j\} = \frac{6}{\theta^2 h^2}\{\Delta y_{j+\theta}\} - \frac{6}{\theta h}\{y'_j\} - 3\{y''_j\} \tag{12.82}$$

$$\{\Delta y'_{j+\theta}\} \equiv \{y'_{j+\theta}\} - \{y'_j\} = \frac{3}{\theta h}\{\Delta y_{j+\theta}\} - 3\{y'_j\} - \frac{\theta h}{2}\{y''_j\} \tag{12.83}$$

$$\begin{aligned}(\frac{6}{\theta^2 h^2}[M] + \frac{3}{\theta h}[C] + [K])\{\Delta y_{j+\theta}\} &= \{\Delta p_{j+\theta}\} + [M](\frac{6}{\theta h}\{y'_j\} + 3\{y''_j\}) \\ &+ [C](3\{y'_j\} + \frac{\theta h}{2}\{y''_j\})\end{aligned} \tag{12.84}$$

12.7 步長之決定與誤差控制

亞當氏類型每個步驟均有開顯式所得之預測值$y_{j+1}^{(o)}$與閉隱式所得之改正值$y_{j+1}^{(\infty)}$，因此可知正確之值$y_{j+1}^{(t)}$應如下二式所示。注意式中之誤差項並不含y_j值之誤差，一般稱為局部誤差，亦即在y_j沒有誤差的情況下，一個步長之誤差。但對於一固定長之時間T而言，步長短則步數多，即步數與h^{-1}成正比，而前一步的誤差會累積到本步誤差中，故累積總誤差與一步之誤差乘步數成正比，因此稱下列之$y_{j+1}^{(o)}$與$y_{j+1}^{(\infty)}$之階數為n 。

$$y_{j+1}^{(t)} = y_{j+1}^{(o)} + \alpha_p h^{n+1} y^{(n+1)}(\xi_p) \tag{12.85}$$

$$y_{j+1}^{(t)} = y_{j+1}^{(\infty)} - \alpha_c h^{n+1} y^{(n+1)}(\xi_c) \tag{12.86}$$

若假設$y^{(n+1)}(\xi_p) = y^{(n+1)}(\xi_c)$，則該二未知值可由上列二式消去而得式(12.87)與式(12.88)。若α_p與α_c已知，即可由式(12.89)估計所得$y_{j+1}^{(\infty)}$之誤差$\delta_{old}^{(\infty)}$，如果總誤差$\Delta_{old}^{(\infty)} = |\delta_{old}^{(\infty)}|\frac{T}{h_{old}}$大於要求之誤差$\Delta_{req}^{(\infty)}$，可用式(12.90)計算新的步長。

$$y_{j+1}^{(t)} = \frac{\alpha_p y_{j+1}^{(\infty)} + \alpha_c y_{j+1}^{(o)}}{\alpha_p + \alpha_c}$$

$$y_{j+1}^{(t)} = y_{j+1}^{(o)} + \frac{\alpha_p}{\alpha_p + \alpha_c}(y_{j+1}^{(\infty)} - y_{j+1}^{(o)}) \tag{12.87}$$

$$y_{j+1}^{(t)} = y_{j+1}^{(\infty)} - \frac{\alpha_c}{\alpha_p + \alpha_c}(y_{j+1}^{(\infty)} - y_{j+1}^{(o)}) \tag{12.88}$$

$$\delta_{old}^{(\infty)} = y_{j+1}^{(\infty)} - y_{j+1}^{(t)} = \frac{\alpha_c}{\alpha_p + \alpha_c}(y_{j+1}^{(\infty)} - y_{j+1}^{(o)}) \tag{12.89}$$

$$h_{new} = h_{old}\left(\frac{\Delta_{req}^{(\infty)}}{\Delta_{old}^{(\infty)}}\right)^{1/n} \tag{12.90}$$

對於亞當氏之閉隱式(12.55)以反覆方式試算，相當於直接代入法以$y = G(y) = \gamma_1 h f(y, t_{j+1}) + c_2$求解，其中$\gamma_1$與$t_{j+1}$及$c_2 = y_j + h\gamma_2 f_j + \cdots$

為與y_{j+1}無關之常數。其會收斂之條件為$|G'(y)| < 1$，即$\gamma_1 h|f_y| < 1$。令$r = \gamma_1 h|f_y|$，則收斂條件為$r < 1$。若考慮運算效率，希望第一次改正值$y_{j+1}^{(1)}$即能符合所需精度而為$y_{j+1}^{(\infty)}$之近似值，其h可由下述方式決定：由第一章式(1.13)可得式(12.91)，經整理可得式(12.92)與式(12.93)。通常$r << 1$，可設$1 - r \doteq 1$，以$r = \gamma_1 h|f_y|$代入，即得式(12.94)可用以估算改正值$y_{j+1}^{(\infty)}$，與式(12.95)可用以估算第一次改正值$y_{j+1}^{(1)}$之誤差$\delta_{old}^{(1)}$，如果總誤差$\Delta_{old}^{(1)} = |\delta_{old}^{(1)}|\frac{T}{h_{old}}$大於要求之誤差$\Delta_{req}^{(1)}$，可參考式(12.90)之關係，得新步長的計算式(12.96)。注意f_y值可由$(f_{j+1}^{(1)} - f_{j+1}^{(o)})/(y_{j+1}^{(1)} - y_{j+1}^{(o)})$估算，或由前一步已算得之值近似之，即$f_y = (f_j - f_j^{(o)})/(y_j - y_j^{(o)})$。

$$y_{j+1}^{(\infty)} - y_{j+1}^{(1)} \doteq r(y_{j+1}^{(\infty)} - y_{j+1}^{(o)}) \tag{12.91}$$

$$y_{j+1}^{(1)} - y_{j+1}^{(o)} \doteq (1-r)(y_{j+1}^{(\infty)} - y_{j+1}^{(o)}) \tag{12.92}$$

$$y_{j+1}^{(\infty)} - y_{j+1}^{(1)} \doteq \frac{r}{1-r}(y_{j+1}^{(1)} - y_{j+1}^{(o)}) \tag{12.93}$$

$$y_{j+1}^{(\infty)} \doteq y_{j+1}^{(1)} + \gamma_1 h|f_y|(y_{j+1}^{(1)} - y_{j+1}^{(o)}) \tag{12.94}$$

$$\delta_{old}^{(1)} = y_{j+1}^{(\infty)} - y_{j+1}^{(1)} \doteq \gamma_1 h|f_y|(y_{j+1}^{(1)} - y_{j+1}^{(o)}) \tag{12.95}$$

$$h_{new} = h_{old}\left(\frac{\Delta_{req}^{(1)}}{\Delta_{old}^{(1)}}\right)^{1/(n+1)} \tag{12.96}$$

參考式(12.89)與式(12.94)，若$\Delta_{req}^{(1)} = \Delta_{req}^{(\infty)}$，則可比較$\gamma_1 h|f_y|$與$\frac{\alpha_c}{\alpha_p+\alpha_c}$，或$h$與$\hat{h} = \frac{\alpha_c}{\alpha_p+\alpha_c} \cdot \frac{1}{\gamma_1|f_y|}$，以決定採用式(12.90)或式(12.96)計算h_{new}：
(1)若$h<\hat{h}$，則僅須用一次改正式：用式(12.89)估算$\delta_{old}^{(\infty)}$，如誤差太大，用式(12.90)算h_{new}；否則用式(12.94)估算$y_{j+1}^{(\infty)}$，再用式(12.88)估算$y_{j+1}^{(t)}$。
(2)若$h>\hat{h}$，但不允許做反覆試算：用式(12.95)估算$\delta_{old}^{(1)}$，如誤差太大，用式(12.96)算h_{new}；否則用式(12.94)估算$y_{j+1}^{(\infty)}$，再用式(12.88)估算$y_{j+1}^{(t)}$。
(3)若$h>\hat{h}$，但允許多次反覆試算：用式(12.89)估算$\delta_{old}^{(\infty)}$，如誤差太大，用式(12.90)算h_{new}；否則用式(12.88)估算$y_{j+1}^{(t)}$。如不計算$\hat{h}$以與h比較，而均由反覆試算求$y_{j+1}^{(\infty)}$，則亦採用情況(3)之方式處理。

對於郎吉庫達類型並無開顯式與閉隱式之二種數值可用以估算誤差。此時可以用二種不同步長之結果估算誤差。一般常取另一步長為原來步長h之二倍。若以步長h做二步所得之結果為y_1；以步長$2h$做一步所得之結果為y_2，則正確之值y應如下二式所示。

$$y = y_1 + C2 \cdot h^{n+1}y^{(n+1)}(\xi_1) \tag{12.97}$$

$$y = y_2 + C(2h)^{n+1}y^{(n+1)}(\xi_2) \tag{12.98}$$

若假設$y^{(n+1)}(\xi_1) = y^{(n+1)}(\xi_2)$，則該二未知值可由上列二式消去而得：

$$y = \frac{2^n y_1 - y_2}{2^n - 1} = y_1 + \frac{1}{2^n - 1}(y_1 - y_2) \tag{12.99}$$

$$\delta_{old} = y_1 - y = -\frac{1}{2^n - 1}(y_1 - y_2) \tag{12.100}$$

由上式估算y_1之誤差δ_{old}後，如果總誤差$\Delta_{old} = |\delta_{old}|\frac{T}{h_{old}}$大於要求之誤差$\Delta_{req}$，可比照式(12.90)之類似算式計算新的步長。

12.8 巴利希史特爾法

本節將簡要介紹一種以低階之解連續利用理查生外插法 (Richardson extrapolation)以求較高階之解之巴利希史特爾法(Bulirsch–Stoer method)。該法選用之低階方法為修改自式(12.101)之 二階郎吉庫達法或稱中點法(Midpoint method)。修改之法如式(12.102)至式(12.105)，稱修訂中點法。注意式(12.103)僅為一階，建議改用二階或四階之郎吉庫達法計算$\bar{y}_1$。

$$y_{j+1} = y_j + hf(y_j + \frac{1}{2}hf_j,\ t_j + \frac{1}{2}h) \tag{12.101}$$

$$\bar{y}_o = y_o \tag{12.102}$$

$$\bar{y}_1 = \bar{y}_o + hf(\bar{y}_o,\ t_o) \tag{12.103}$$

$$\bar{y}_{j+1} = \bar{y}_j + 2hf(\bar{y}_{j-1},\ t_{j-1}), \qquad j = 1, 2, \cdots, n-1 \tag{12.104}$$

$$y_n = \frac{1}{2}(\bar{y}_n + \bar{y}_{n-1} + hf(\bar{y}_n,\ t_n)) \tag{12.105}$$

$$y(t_o + H) = y_n + \sum_{i=1}^{\infty} c_i h^{2i} \tag{12.106}$$

該法基本上以一個較大區間H為一求解步驟。將區間依序分成$n = 2, 4, (6), 8, (12), 16, \cdots, 64, (96)$之$n$個小步長$h = H/n$，利用上述修訂中點法計算各$y_n$值。然後利用式(12.106)之關係以求得$y(t_o + H)$之值而完成一大區間之求解步驟。以式(12.106)求$y(t_o + H)$之法有二種方式：一為採用成倍數遞增之n值，如上述不含()內之n值，即可用理查生外插法由下列式(12.107)求得$y(t_o + H) = y_2^{(M)}$。注意這些計算式與第二章倫伯格積分表所用計算式相同。 另一方式為採用上述不成倍數遞增之n值，目的是使n值不要增加太快，但有關之類似外插公式之係數之計算規則較複雜， 此時可考慮用牛頓插值法(或葉特肯插值公式)，將y_n視為$x = 1/n^2$之函數

$F(x) = y_n$，然後以式(12.108)計算$x = 0$時之$F(0)$即為$y(to + H)$。巴利希史特爾另外採用有理函數(Rational function)，即二個多項式之商，以取代式(12.106)之多項式函數，此種做法對於計算區間有函數值接近奇異點時，能較準確的求得近似值，其詳細做法請參考[1]。

$$\begin{aligned} y_n^{(1)} &= y_n\,, & n = 2, 4, 8, \cdots, 2^M \\ y_n^{(m+1)} &= \frac{4^m y_{2n}^{(m)} - y_n^{(m)}}{4^m - 1}\,, & m = 1, 2, 3, \cdots, M-1 \\ y(t_o + H) &= y_2^{(M)} & (12.107) \end{aligned}$$

$$\begin{aligned} y(t_o + H) = F(0) &= F(\frac{1}{4}) - \frac{1}{4}F[\frac{1}{4}, \frac{1}{16}] + \frac{1}{4 \cdot 16}F[\frac{1}{4}, \frac{1}{16}, \frac{1}{36}] \\ &\quad - \frac{1}{4 \cdot 16 \cdot 36}F[\frac{1}{4}, \frac{1}{16}, \frac{1}{36}, \frac{1}{64}] + \cdots \quad (12.108) \end{aligned}$$

雖然表面上巴利希史特爾法可外插求得很高階數(2^M)的函數值，但因該高階函數含蓋之範圍頗大，其實際準確性並不一定會高於含蓋範圍小之低階函數。含蓋範圍最小(h)的郎吉庫達法，其用以計算新點之其他函數值均在一個步長(h)之內，因此最能準確計算局部劇烈變化之函數。含蓋範圍較大$(4h)$的四階亞當氏法，其用以計算新點之其他函數值則在四個步長$(4h)$之內，因此對局部劇烈變化之函數已略有困難。含蓋範圍最大(nh)的巴利希史特爾法，其用以計算新點之其他函數值則分布在n個步長$(nh = H)$之內，因此最不能應付局部劇烈變化之函數。基於上述考量，本章不考慮提供巴利希史特爾法之相關程式。對於函數變化較平緩者，為求效率，以採用亞當氏法較適當，因其有時還能兼顧局部劇烈變化之函數。對於局部劇烈變化之函數，則以採用郎吉庫達法較適當。由以上之說明亦可略知，郎吉庫達法雖然每步之函數計算量為相等步長之亞當氏法之二倍，但其多出的計算量應該不會完全浪費。因此，如果計算效率不是重要考量因素，例如計算時間不多，則以郎吉庫達法為最佳選擇。

12.9 邊界值問題

僅以下列二階常微分方程為例，簡要說明如何以初始值問題求解。

$$A(t)\frac{d^2y}{dt^2} + B(t)y = F(t) \qquad (12.109)$$

$$y(t_o) = y_o, \qquad y(t_1) = y_1 \qquad (12.110)$$

令$y'(t_o)=v$，其中v為任意假設之值，解式(12.111)與式(12.112)之初始值問題可得$y(t_1)=y_1^*$。因y_1^*之值係隨v值而變，可視其為v之函數。令$f(v)=y_1^*-y_1$，若解得$f(v)=0$之根為$v=v_o$。則原邊界值問題之解為對應初始值為$y(t_o)=y_o$與$y'(t_o)=v_o$之初始值問題之解。解$f(v)=0$之根請直接參考第一章。此法又稱試射法 (shooting method)，因調整$y'(t_o)$有如調整槍把之角度，$y(t_1)$為t_1處之射點高度。

$$A(t)\frac{d^2y}{dt^2}+B(t)y=F(t) \tag{12.111}$$

$$y(t_o)=y_o, \qquad y'(t_o)=v \tag{12.112}$$

$$y(t_1)=y_1^*, \qquad f(v)=y_1^*-y_1=0 \tag{12.113}$$

上述方法可延用於n個條件不在t_o時之情況：若第i個條件為某點(非t_o點)之某一函數值或微分值為g_i，假設n個待定之未知初始值為變數，設為x_i，$i=1,2,\cdots,n$。由任意設定之未知初始值x_i配合其他已知之初始值，即可按初始值問題求解，設對應於第i條件之值為g_i^*，因g_i^*會隨$x_1,x_2,\cdots,x_n$之值而變，可視其為這些變數之函數。令$f_i(x_1,x_2,\cdots,x_n)=g_i^*-g_i$。解$f_i(x_1,x_2,\cdots,x_n)=0$之根，則其所對應之初始值問題之解，即為原邊界值問題之解。

若常微分方程式為線性，則可僅用$n+1$組初始條件之解，然後由這些解之線性組合，以使其符合$n+1$個條件，則此線性組合之解即為所求之解。例如：設第j組未知初始值為$\{x\}_j=\{e_j\}$，即第j個變數為1，其餘變數為0。配合其他已知初始值，可求得該第j組初始值問題之解為$y_j(t)$，同時得第i條件之值為a_{ij}。除上述n組解外，還需解一組稱為第0組之初始值問題，其初始值為所有未知初始值等於0，即$\{x\}_o=\{0\}$，亦配合其他已知初始值，同樣可得解$y_o(t)$與第i條件之值a_{io}。因此初始值為$\{x_1,x_2,\cdots,x_n\}$之解為式(12.114)之線性組合解，若符合n個邊界值條件之式(12.115)與另一條件式(12.116)，則該線性組合解即為原邊界值問題之解。

$$y(t)=\sum_{j=0}^{n}x_jy_j(t) \tag{12.114}$$

$$\sum_{j=0}^{n}a_{ij}x_j-g_i=0, \qquad i=1,2,\cdots,n \tag{12.115}$$

$$\sum_{j=0}^{n}x_j-1=0 \tag{12.116}$$

式(12.116)之條件為使線性組合之初始值符合其他已知初始值。其理由說明如下：先請注意對應未知初始值為$\{e_j\}x_j$之解為：上述第j組解乘以x_j，即$x_jy_j(t)$，所有已知初始值亦應為原初始值乘以x_j，例如某一已知初始值為g，則其解對應之該初始值為x_jg。因此對應式(12.114)之組合解之該初始值為$x_og+x_1g+x_2g+\cdots+x_ng$，令其等於$g$，即得條件式(12.116)。式(12.116)之另一種解釋為：以式(12.109)為例，將組合解$y(t)=x_oy_o(t)+x_1y_1(t)$代入式(12.109)，得$x_o(A(t)y_o''(t)+B(t)y_o(t))+x_1(A(t)y_1''(t)+B(t)y_1(t))=F(t)$，但因$A(t)y_o''(t)+B(t)y_o(t)=F(t)$，$A(t)y_1''(t)+B(t)y_1(t)=F(t)$，故得$x_oF(t)+x_1F(t)=F(t)$，因此得條件式$x_o+x_1=1$。

12.10 以聯立差分式解邊界值問題

將微分方程式之微分以差分代替做近似處理可得差分方程式，邊界上之條件如為微分亦以差分代替可得差分條件式。以四階常微分方程式$f(y'''',y''',y'',y',y,t)=0$為例，設其四個邊界條件寫成$g_i(y''''(a),y''(a),y'(a),y(a),y'''(b),y''(b),y'(b),y(b))=0$，$i=1,2,3,4$。將$t=a$至$t=b$間分為$m$等分，即令$h=\frac{b-a}{m}$，$t_j=a+hj$，$y_j=y(t_j)$。將下列差分(只取等式右邊第一項)代入$j=0,1,2,\cdots,m$共$m+1$個常微分方程式$f(y_j'''',y_j''',y_j'',y_j',y_j,t_j)=0$中，及代入$i=1,\cdots,4$等四個邊界條件$g_i(y_o''',y_o'',y_o',y_o,y_m''',y_m'',y_m',y_m)=0$中，可得僅含$y_{-2},y_{-1},y_o,y_1,\cdots,y_{m-1},y_m,y_{m+1},y_{m+2}$等$m+5$個函數值之$m+5$個差分式($m+1$個差分方程式加4個差分條件式)。注意其中不含函數之微分值，但多了$y_{-2},y_{-1},y_{m+1},y_{m+2}$等四個不在$t=a$至$b$範圍內之函數值。由此$m+5$個差分式即可解得$m+5$個函數值。注意此處所用之差分為對稱型式之差分，一般稱為中心差分，具有二階近似之優點；式(12.19)中所用之反向差分則僅為一階近似。

$$
\begin{aligned}
y_j' &= \frac{y_{j+1}-y_{j-1}}{2h}-\frac{h^2}{3!}y_j'''-\cdots\\
y_j'' &= \frac{y_{j+1}-2y_j+y_{j-1}}{h^2}-\frac{h^2}{12}y_j''''-\cdots\\
y_j''' &= \frac{y_{j+2}-2y_{j+1}+2y_{j-1}-y_{j-2}}{2h^3}-\frac{h^2}{4}y_j^{(v)}-\cdots\\
y_j'''' &= \frac{y_{j+2}-4y_{j+1}+6y_j-4y_{j-1}+y_{j-2}}{h^4}-\frac{h^2}{6}y_j^{(vi)}-\cdots
\end{aligned}
$$

利用上述方式處理奇數階之微分方程式時，差分式會比未知函數少一個。以三階微分方程式為例，未知函數仍為$y_{-2},\cdots,y_{m+2}$等$m+5$個，但差分條件式僅有三個，因此差分式比未知函數少一個，可用下述方式處理：將其中一邊界點處所用之最高階微分用下列不對稱型式之差分取代，則可減少y_{-2}(或y_{m+2})之未知函數。

$$\begin{aligned} y'_j &= \frac{y_{j+1}-y_j}{h} - \frac{h}{2!}y''_j - \frac{h^2}{3!}y'''_j - \cdots \\ y'''_j &= \frac{y_{j+2}-3y_{j+1}+3y_j-y_{j-1}}{h^3} - \frac{h}{2}y''''_j - \frac{h^2}{4}y^{(v)}_j - \cdots \end{aligned}$$

但上列差分僅為一階近似，如欲得到與其他中心差分同為二階近似之差分，可用下列不對稱型式之偶數階差分代入上式中含h之微分項，即得其後所列之二階近似之不對稱型式之奇數階差分。上述二階近似之中心差分，亦可採用類似方法將其後之高階微分項以中心差分代入，即可得四階或更高階近似之中心差分。

$$\begin{aligned} y''_j &= \frac{y_{j+2}-2y_{j+1}+y_j}{h^2} - hy'''_j - \frac{7h^2}{12}y''''_j - \cdots \\ y''''_j &= \frac{y_{j+3}-4y_{j+2}+6y_{j+1}-4y_j+y_{j-1}}{h^4} - hy^{(v)}_j - \frac{2h^2}{3}y^{(vi)}_j - \cdots \end{aligned}$$

$$\begin{aligned} y'_j &= \frac{-y_{j+2}+4y_{j+1}-3y_j}{2h} + \frac{h^2}{3}y'''_j + \cdots \\ y'''_j &= \frac{-y_{j+3}+6y_{j+2}-12y_{j+1}+10y_j-3y_{j-1}}{2h^3} + \frac{h^2}{4}y^{(v)}_j + \cdots \end{aligned}$$

如果常微分方程式為線性，則所得之聯立差分式亦為線性聯立方程式，其係數矩陣常為帶寬不大之帶矩陣，可以用第三章或第四章之方法分解成下三角矩陣與上三角矩陣，再以前進代入及反向代入直接求解。其運算速度應該比用高斯賽德法以反覆試算方式求解快而準確。注意若微分方程式為奇數階時，在邊界點處之差分式，會因採用二階近似之不對稱型式之差分，而使其帶寬較一般大多數差分式之帶寬多一，另外在邊界處之條件式之帶寬亦可能較大，此時可採用變寬帶矩陣之方式處理，即可不影響求解效率。

以下列邊界值問題為例，取$m=5$，$h=0.2$，可列出二個邊界差分條件式(第一式與最後一式)及六個差分方程式。注意由最後一式知$y_o=1$，將其代入第六式以消去該式之y_o變數，則前六個方程式僅含六個未知

數，即可直接求解。此時第七式主要用以計算y_{-1}之值，並不影響前六個變數之結果，而y_{-1}並非需要之值，故第七式亦可被去掉而不用。亦即用八個未知數的八元聯立式，與用六個未知數的六元聯立式所得之結果相同，均為$y_6 = 1.879,\ y_5 = 1.345,\ y_4 = 1.079,\ y_3 = .949,\ y_2 = .901,\ y_1 = .918$ 。

$$\frac{d^2y}{dt^2} - 4t\frac{dy}{dt} + (4t^2 - 1)y = e^{t^2} \tag{12.117}$$

$$y(0) = 1, \qquad y'(1) = 2 \tag{12.118}$$

$$y_6 - y_4 = 2 * 2 * 0.2$$

$$(y_6 - 2y_5 + y_4) - 4 * 1.0 * .1 * (y_6 - y_4) + .04 * (4 * 1.00 - 1)y_5 = .04 * e^{1.00}$$

$$(y_5 - 2y_4 + y_3) - 4 * 0.8 * .1 * (y_5 - y_3) + .04 * (4 * 0.64 - 1)y_4 = .04 * e^{0.64}$$

$$(y_4 - 2y_3 + y_2) - 4 * 0.6 * .1 * (y_4 - y_2) + .04 * (4 * 0.36 - 1)y_3 = .04 * e^{0.36}$$

$$(y_3 - 2y_2 + y_1) - 4 * 0.4 * .1 * (y_3 - y_1) + .04 * (4 * 0.16 - 1)y_2 = .04 * e^{0.16}$$

$$(y_2 - 2y_1 + y_o) - 4 * 0.2 * .1 * (y_2 - y_o) + .04 * (4 * 0.04 - 1)y_1 = .04 * e^{0.04}$$

$$(y_1 - 2y_o + y_{-1}) - 4 * 0.0 * .1 * (y_1 - y_{-1}) + .04 * (4 * 0.00 - 1)y_o = .04 * e^{0.00}$$

$$y_o = 1$$

$$\begin{bmatrix} 1 & 0 & -1 & & & & & \\ 0.60 & -1.880 & 1.40 & & & & & \\ & 0.68 & -1.938 & 1.32 & & & & \\ & & 0.76 & -1.982 & 1.24 & & & \\ & & & 0.84 & -2.014 & 1.16 & & \\ & & & & 0.92 & -2.034 & 1.08 & \\ & & & & & 1.00 & -2.040 & 1.00 \\ & & & & & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_o \\ y_{-1} \end{Bmatrix} = \begin{Bmatrix} .80 \\ .10873 \\ .07586 \\ .05733 \\ .04694 \\ .04163 \\ .04 \\ 1.0 \end{Bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 & & & & \\ 0.60 & -1.880 & 1.40 & & & & \\ & 0.68 & -1.938 & 1.32 & & & \\ & & 0.76 & -1.982 & 1.24 & & \\ & & & 0.84 & -2.014 & 1.16 & \\ & & & & 0.92 & -2.034 & \end{bmatrix} \begin{Bmatrix} y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \end{Bmatrix} = \begin{Bmatrix} .80 \\ .10873 \\ .07586 \\ .05733 \\ .04694 \\ -1.03837 \end{Bmatrix}$$

12.11 副程式 $RUNGEC$

[表一]中副程式為採用四階郎吉氏係數之郎吉庫達法。當$Mmax = 0$時可做固定步長之解，計算時間最少，但不能獲得解之誤差大小，可用於已經先做過誤差分析比較之情形。當$Mmax = 1$時亦可做固定步長之解，但為了獲得解之誤差大小，內部自動分成二個小區間求解，以便與全區間之解比較，而利用式(12.100)與式(12.99)計算誤差及較正確之值。但其計算時間為$Mmax = 0$者之$11/8$倍。該副程式所傳回之誤差值可用以調整下步之步長。如果$Mmax$設定較大之值，該副程式可連續將小區間數加倍，直至達到要求精度，或小區間數已達2^{Mmax}為止。該副程式所傳回之誤差值及$Muse$值(表示所用小區間數為2^{Muse})，亦可用以調整下步之步長或$Mmax$值。該副程式內部之步長調整係以減半方式進行，可能有點保守而效率略差，但可得等間距之函數值，且呼叫容易，應為理想之做法。如果不要求等間距之函數值，則可考慮於每次呼叫前自行決定步長，以增加效率。[表二]為範例主程式。副程式$RUNGEC$之輸入$YI(N)$與輸出$YO(N)$可以相同，範例程式即令二者均為$Y(N)$，因不留存於陣列中須將其寫出。如欲將結果留存於陣列中以備後繼計算之用，可改用以$'CRN'$及$'C*N'$標示之指令。又該範例主程式亦可做為呼叫[表三]副程式$ADAMSC$之用，但須分別改用以$'CA1'$及$'CA*'$標示（不留存$Y(N)$）或以$'CAN'$、$'C*N'$及$'CA*'$標示（要存$Y(N,*)$）之指令。由本範例程式可看出副程式$RUNGEC$與$ADAMSC$均很容易叫用，故可不必另寫複雜的驅動程式。該二副程式之呼叫方式差異不大，可互相試用或取代。

[表一] 郎吉庫達法求解副程式

```
************************************************************************
      SUBROUTINE RUNGEC(N,FF,DT,T,YI,YO,X,Mmax,Muse,EPSreq,EPSold)
C**   ================================================================   **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION YI(N),YO(N),X(N,7)
      EXTERNAL FF
C**   ================================================================   **
C**   Input T,YI(N) at T=TI;          Output T,YO(N) at T=TO=TI+DT    **
C**   Also Output FI(N)=f_i(YI,TI) in X(N,1)                          **
C**   YI(N) & YO(N) can be the same location : Loc.YO(N)=Loc.YI(N)    **
C**   ----------------------------------------------------------------   **
C*I   N      = Number of dependent variables                          **
C*S   FF     = Subroutine to compute F(N) for given T & Y(N)          **
C*I   DT     = Time_step = Final_time - Initial_time                  **
CIO   T      = Initial time TI & Final time TO=TI+DT                  **
C*I   YI(N)  = y_i(TI)                                                **
C*O   YO(N)  = y_i(TO)                                                **
```

```
C*W   X(N,7) = Buffers X(4*N) if Mmax=0; Buffers X(7*N) if Mmax>0     **
C*I   Mmax   = Index for accuracy control                            **
C**          = 0: No control on accuracy (EPSreq,EPSold,Muse:NA)     **
C**          > 0: Control accuracy with allowed Max_steps = 2**Mmax  **
C*O   Muse   = Actual Used_steps = 2**Muse                           **
C*I   EPSreq = Req. epsilon such that EPSold < EPSreq                **
C*O   EPSold = Max. epsilon = Max_{i=1}^n |1-Y2(i)/Y1(i)|/15         **
C**   To adjust DT : DT_new = (EPSreq/EPSold)**(1/4) * DT / 2**Muse  **
C**   ================================================================
      CALL FF(T,YI,X(1,1),N)
C**   ----------------------------------------------------------------
      IF(Mmax.LE.0) THEN
        CALL RUNGE(N,FF,DT,T,YI,YO,X(1,1),X(1,2),X(1,3),X(1,4))
C**     CALL FF(T,YO,X(1,2),N)
        RETURN
      ENDIF
C**   +--------------------------------------------------------------+
C**   | Compute Y2(N) = X(N,5) by Large_step_size = 2*h               |
C**   +--------------------------------------------------------------+
      T2=T
      CALL RUNGE(N,FF,DT,T2,YI,X(1,5),X(1,1),X(1,2),X(1,3),X(1,4))
C**   +--------------------------------------------------------------+
C**   | Compute Y1(N) = X(N,6) by Small_step_size = h = DT/Nstep      |
C**   +--------------------------------------------------------------+
      NSTEP=1
      DO 90 Muse=1,Mmax
      NSTEP=NSTEP*2
      HT=DT/NSTEP
C**   +--------------------------------------------------------------+
C**   | Restart Nstep of HT = h = DT/Nstep                           |
C**   +--------------------------------------------------------------+
      T1=T
      CALL RUNGE(N,FF,HT,T1,YI,X(1,6),X(1,1),X(1,2),X(1,3),X(1,4))
      DO 50 ISTEP=2,NSTEP
      CALL FF(T1,X(1,6),X(1,7),N)
      CALL RUNGE(N,FF,HT,T1,X(1,6),X(1,6),X(1,7),X(1,2),X(1,3),X(1,4))
   50 CONTINUE
C**   +--------------------------------------------------------------+
C**   | Find the Max. reletive error = |1 - Y2(I))/Y1(I)|/15         |
C**   +--------------------------------------------------------------+
      EPSold=0.0
      DO 60 I=1,N
      IF(X(I,6).NE.0.) EPSold=DMAX1(EPSold,DABS((1.0-X(I,5)/X(I,6))))
   60 CONTINUE
      EPSold=EPSold/15.0
C**   +--------------------------------------------------------------+
C**   | If EPSold >  EPSreq : Double the number of steps             |
C**   | If EPSold <= EPSreq : Return YO(I)=Y1(I)+(Y1(I)-Y2(I))/15|
C**   +--------------------------------------------------------------+
      IF(EPSold.GT.EPSreq.AND.Muse.LT.Mmax) THEN
        DO 70 I=1,N
        X(I,5)=X(I,6)
   70   CONTINUE
      ELSE
        T=T+DT
        DO 80 I=1,N
        YO(I)=X(I,6)+(X(I,6)-X(I,5))/15D0
   80   CONTINUE
C**     CALL FF(T,YO,X(1,2),N)
        RETURN
```

```
      ENDIF
   90 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE RUNGE(N,FF,DT,T,YI,YO,FI,FO,FC,YC)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION YI(N),YO(N),FI(N),FO(N),FC(N),YC(N)
      EXTERNAL FF
C**   ------------------------------------------------------------    **
C**   Input  T,YI(N),FI(N) at T=TI                                    **
C**   Output T,YO(N)       at T=TO=TI+DT                              **
C**   YI(N) & YO(N) can be the same location : Loc.YO(N)=Loc.YI(N)    **
C**   ------------------------------------------------------------    **
C*I   N      = Number of dependent variables                          **
C*S   FF     = Subroutine to compute F(N) for given T & Y(N)          **
C*I   DT     = Time_step = Final_time - Initial_time                  **
CIO   T      = Initial time TI & Final time TO=TI+DT                  **
C*I   YI(N)  = y_i(TI)                                                **
C*O   YO(N)  = y_i(TO)                                                **
C*I   FI(N)  = f_i(YI,TI)                                             **
C*W   FO(N)  = Working array for Fa(N),Fa(N)+Fb(N)                    **
C*W   FC(N)  = Working array for       Fb(N),Fc(N)                    **
C*W   YC(N)  = Working array for Ya(N),Yb(N),Yc(N)                    **
C**   ============================================================    **
      HT=DT*0.5
      T=T+HT
      DO 10 I=1,N
      YC(I)=YI(I)+FI(I)*HT
   10 CONTINUE
      CALL FF(T,YC,FO,N)
C**   ------------------------------------------------------------    **
      DO 20 I=1,N
      YC(I)=YI(I)+FO(I)*HT
   20 CONTINUE
      CALL FF(T,YC,FC,N)
C**   ------------------------------------------------------------    **
      T=T+HT
      DO 30 I=1,N
      YC(I)=YI(I)+FC(I)*DT
      FO(I)=FO(I)+FC(I)
   30 CONTINUE
      CALL FF(T,YC,FC,N)
C**   ------------------------------------------------------------    **
      HT=DT/6.0
      DO 40 I=1,N
      YO(I)=YI(I)+(FI(I)+FO(I)+FO(I)+FC(I))*HT
   40 CONTINUE
C**   CALL FF(T,YO,FO,N)
      RETURN
      END
*************************************************************************
```

［表二］ 郎吉庫達法求解之主程式

```
*************************************************************************
      PROGRAM RUNGET
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      DIMENSION Y(2),X(2,7)
CRN   DIMENSION Y(2,100),X(2,7)
CA1   DIMENSION Y(2),X(2,16),ID(3)
CAN   DIMENSION Y(2,100),X(2,16),ID(3)
      EXTERNAL FUN
      DATA N,T,U0,V0/2,0.0D0,1.0D0,2.0D0/,W,XI/3.0D0,0.05D0/
C**   ================================================================    **
C**   y''(T) + 2*XI*W*y'(T) + W*W*y(T) = 0 with y(0)=U0, y'(0)=V0     **
C**   ================================================================    **
      READ (*,'(I4,4F8.0)') Mmax,DT,EPSreq
      IF(EPSreq.EQ.0.0) EPSreq=0.0001D0
      IF(DT.EQ.0.0) DT=0.2D0
      NSTEP=3.01/DT+1
C**   ----------------------------------------------------------------    **
      Y(1)=U0
      Y(2)=V0
C*N   Y(1,1)=U0
C*N   Y(2,1)=V0
CA*   ID(3)=-3
C**   ----------------------------------------------------------------    **
      WRITE(*,'(5X,''T        V            Y        YT(exact)''
     *         ,6X,''Y-YT       EPSold*YT  Muse'')')
C**   +---------------------------------------------------------------+    **
C**   | YT = Exact solution of Y(1)                                   |    **
C**   +---------------------------------------------------------------+    **
      D=DSQRT(1.0-XI**2)
      O=DATAN(XI/D)
      WD=W*D
      EDT=1.0/D
      EWDT=DEXP(-XI*W*DT)
      DO 50 ISTEP=1,NSTEP
      YT=EDT*(U0*DCOS(WD*T-O)+V0*DSIN(WD*T)/W)
      EDT=EDT*EWDT
C**   ----------------------------------------------------------------    **
      WRITE(*,'(F7.3,1P5E13.5,I4)')
     *      T,Y(2),Y(1),YT,Y(1)-YT,EPSold*YT,Muse
C*N  *      T,Y(2,ISTEP),Y(1,ISTEP),YT,Y(1,ISTEP)-YT,EPSold*YT,Muse
C**   ----------------------------------------------------------------    **
      CALL RUNGEC(N,FUN,DT,T
     *     ,Y,Y,X,Mmax,Muse,EPSreq,EPSold)
CRN  *     ,Y(1,ISTEP),Y(1,ISTEP+1),X,Mmax,Muse,EPSreq,EPSold)
CA*   CALL ADAMSC(N,FUN,DT,T
CA1  *     ,Y,Y,X,Mmax,Muse,EPSreq,EPSold,ID)
CAN  *     ,Y(1,ISTEP),Y(1,ISTEP+1),X,Mmax,Muse,EPSreq,EPSold,ID)
   50 CONTINUE
      END
*************************************************************************
      SUBROUTINE FUN(T,Y,F,N)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(N),F(N)
      DATA WW,W,XI/9.0D0,3.0D0,0.05D0/
C**   ================================================================    **
C**   y''(T) + 2*XI*W*y'(T) + W*W*y(T) = 0 with y(0)=U0, y'(0)=V0     **
C**   ================================================================    **
      F(1)=Y(2)
      F(2)=-2.0*XI*W*Y(2)-WW*Y(1)
      RETURN
      END
*************************************************************************
```

12.12 副程式 $ADAMSC$

[表三]中副程式為採用四階亞當氏之預測改正法， 並以式(12.88)(其中$\alpha_p = 251/720$ 、$\alpha_c = 19/720$)利用預測值與改正值內插計算較正確之值。當$Mmax = 0$時可做固定步長之解，計算時間最少，可用於已經先做過誤差分析比較之情形。當$Mmax > 0$時，該副程式可連續將小區間數加倍，直至達到要求精度，或小區間數已達2^{Mmax}或已達4為止。該副程式所傳回之誤差值及$Muse$值(表示所用小區間數為2^{Muse})，亦可做為調整步長之參考。理想之步長應使大部分$Muse$值為0，最好不要或極少有$Muse = 2$者出現。如果在$Muse$大部分為0時，又出現頗多之2，即可能表示函數之變化，有時平緩，有時劇烈，此種函數以採用$RUNGEC$較合適。因為亞當氏法多用到前面三點之函數值，即其記性較強，較難適應變化較大的新狀況，故對此富變化之函數較不適用。因此該副程式僅做到$Mmax = 2$之情形(雖然$Mmax > 2$之運算程式亦可比照寫成，但$Mmax$愈大所須之暫存陣列愈多，程式亦相對增長)。該副程式內部之步長調整係以減半方式進行，可能有點保守而效率略差，但可得等間距之函數值，且呼叫容易，應為理想之做法。另注意該程式僅用一次改正式，如欲用多次改正式試算至收斂，可更改$MAXC$值。但增加$MAXC$亦增加函數計算次數，倒不如減小步長有利。因步長減小，預測值與改正值更接近(與h^5成比例)，改正式之收斂率亦較快(與h成比例)，通常即可達到不用第二次改正式之效果。本副程式之使用範例可參考[表二]，與前一節中對範例主程式之說明。

[表三] 亞當氏法求解副程式

```
************************************************************************
      SUBROUTINE ADAMSC(N,FF,DT,T,YI,YO,X,Mmax,Muse,EPSreq,EPSold,ID)
C**   ================================================================   **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION YI(N),YO(N),X(N,16),ID(3)
      EXTERNAL FF
      DATA MAXC/0/
C**   ================================================================   **
C**   Input T,YI(N) at T=TI;          Output T,YO(N) at T=TO=TI+DT    **
C**   Input ID(3)=-3 for the first step, then DON'T CHANGE ID(1:3)    **
C**   YI(N) & YO(N) can be the same location : Loc.YO(N)=Loc.YI(N)    **
C**   ----------------------------------------------------------------   **
C*I   N      = Number of dependent variables                          **
C*S   FF     = Subroutine to compute F(N) for given T & Y(N)          **
C*I   DT     = Time_step = Final_time - Initial_time                  **
CIO   T      = Initial time TI & Final time TO=TI+DT                  **
C*I   YI(N)  = y_i(TI)                                                **
```

```
C*O   YO(N)  = y_i(TO)                                                  **
C*I   X(N,1:3) = F3(Y3,T3),F2(Y2,T2),F1(Y1,T1) : Tk=TI-k*DT             **
C*O   X(N,1:3) = F2(Y2,T2),F1(Y1,T1),FI(YI,TI) : Yk=y_i(Tk)             **
C*W   X(N,16)  = Buffers X(9*N),X(12*N),X(16*N) for Mmax=(0,1,2)        **
C*W            : X(N,6:8) for SUBROUTINE ADAMS(...,FP,YC,YP)           **
C*W            : X(N,9:9) for YI(N) in case of Loc.YO(N)=Loc.YI(N)     **
C**   ------------------------------------------------------------     **
C**   X1            X2           X3             X4             X5     **
C**     ID() = 1:         X10    X3     X11     X4     X12     X5     **
C**     ID() = 2:                   X13 X11 X14 X4 X15 X12 X16 X5     **
C**              :     ------------ -------------- --------------      **
C**   ID(3)  = 0:2  ===>  ID(1)         ID(2)          ID(3)           **
C**   ------------------------------------------------------------     **
C**   MAXC   = Additional corrector cycles allowed = 0                 **
C**   MUSC   = Actual corrector cycles used                            **
C*I   Mmax   = Allowed Max_steps = 2**Mmax                             **
C*O   Muse   = Actual Used_steps = 2**Muse                             **
C*I   EPSreq = Req. epsilon such that Max.|dYC/YC(i)| < EPSreq         **
C*O   EPSold = Max. epsilon = Max_{i=1}^n |1-YP(i)/YC(i)|*19/270       **
C**   To adjust DT : DT_new = (EPSreq/EPSold)**(1/4) * DT / 2**Muse    **
C**   ============================================================     **
C**   +----------------------------------------------------------+     **
C**   | For the initial 3-steps by RUNGEC, also set ID(1:3)=0    |     **
C**   +----------------------------------------------------------+     **
      IF(ID(3).GE.0) GO TO 100
      ID(3)=ID(3)+1
      ISTEP=ID(3)+3
      IF(ISTEP.LE.0) STOP 'ADAMSC: ID(3) < -3'
      ID(ISTEP)=0
      CALL RUNGEC(N,FF,DT,T,YI,YO,X(1,ISTEP),Mmax,Muse,EPSreq,EPSold)
      RETURN
C**   +----------------------------------------------------------+     **
C**   | Find FI(N) = X(N,4)                                      |     **
C**   | Save YI(N) in X(N,9) in case of Loc.YO(N)=Loc.YI(N)      |     **
C**   +----------------------------------------------------------+     **
  100 CALL FF(T,YI,X(1,4),N)
      DO 110 I=1,N
      X(I,9)=YI(I)
  110 CONTINUE
C**   +----------------------------------------------------------+     **
C**   | Perform 1_step at DT_step_size                           |     **
C**   +----------------------------------------------------------+     **
      IF(ID(3).GT.0) GO TO 200
      T1=T
      CALL ADAMS(N,FF,DT,T1,YI,YO,X(1,1),X(1,2),X(1,3),X(1,4),X(1,5)
     *          ,X(1,6),X(1,7),X(1,8),MAXC,MUSC,EPSreq,EPSold)
      IF(EPSold.LE.EPSreq.OR.Mmax.LE.0) GO TO 500
      ID(3)=1
      DO 120 I=1,N
      YI(I)=X(I,9)
  120 CONTINUE
C**   +----------------------------------------------------------+     **
C**   | Perform 2_steps at DT/2_step_size                        |     **
C**   +----------------------------------------------------------+     **
  200 IF(ID(3).GT.1) GO TO 400
      IF(ID(1).LT.1) THEN
        ID(1)=1
        DO 210 I=1,N
        X(I,10)=(-X(I,1)+9D0*(X(I,2)+X(I,3))-X(I,4))/16D0
  210   CONTINUE
```

```
      ENDIF
      IF(ID(2).LT.1) THEN
        ID(2)=1
        DO 220 I=1,N
        X(I,11)=(X(I,2)-4D0*X(I,10)+6D0*X(I,3)+X(I,4))/4D0
  220   CONTINUE
      ENDIF
      T2=T
      HT=DT*0.5D0
      CALL ADAMS(N,FF,HT,T2,YI,YO,X(1,10),X(1,3),X(1,11),X(1,4),X(1,12)
     *           ,X(1,6),X(1,7),X(1,8),MAXC,MUSC,EPSreq,EPSold)
      CALL FF(T2,YO,X(1,12),N)
      CALL ADAMS(N,FF,HT,T2,YO,YO,X(1,3),X(1,11),X(1,4),X(1,12),X(1,5)
     *           ,X(1,6),X(1,7),X(1,8),MAXC,MUSC,EPSreq,EPSold)
      IF(EPSold.LE.EPSreq.OR.Mmax.LE.1) GO TO 500
      ID(3)=2
      DO 240 I=1,N
      YI(I)=X(I,9)
  240 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Perform 4_steps at DT/4_step_size                      |    **
C**   +--------------------------------------------------------+    **
  400 IF(ID(3).GT.2) GO TO 500
      IF(ID(2).LT.1) STOP 'ADAMSC: INTERNAL ERROR !'
      IF(ID(2).LT.2) THEN
        ID(2)=2
        DO 430 I=1,N
        X(I,13)=(-X(I,10)+9D0*(X(I,3)+X(I,11))-X(I,4))/16D0
        X(I,14)=(X(I,10)-5D0*(X(I,3)-X(I,4))+15D0*X(I,11))/16D0
  430   CONTINUE
      ENDIF
      T4=T
      HT=DT*0.25D0
      CALL ADAMS(N,FF,HT,T4,YI,YO,X(1,13),X(1,11),X(1,14),X(1,4),X(1,15)
     *           ,X(1,6),X(1,7),X(1,8),MAXC,MUSC,EPSreq,EPSold)
      CALL FF(T4,YO,X(1,15),N)
      CALL ADAMS(N,FF,HT,T4,YO,YO,X(1,11),X(1,14),X(1,4),X(1,15),X(1,12)
     *           ,X(1,6),X(1,7),X(1,8),MAXC,MUSC,EPSreq,EPSold)
      CALL FF(T4,YO,X(1,12),N)
      CALL ADAMS(N,FF,HT,T4,YO,YO,X(1,14),X(1,4),X(1,15),X(1,12),X(1,16)
     *           ,X(1,6),X(1,7),X(1,8),MAXC,MUSC,EPSreq,EPSold)
      CALL FF(T4,YO,X(1,16),N)
      CALL ADAMS(N,FF,HT,T4,YO,YO,X(1,4),X(1,15),X(1,12),X(1,16),X(1,5)
     *           ,X(1,6),X(1,7),X(1,8),MAXC,MUSC,EPSreq,EPSold)
      IF(EPSold.LE.EPSreq) GO TO 500
      ID(3)=3
C**   +--------------------------------------------------------+    **
C**   | Shift one step                                         |    **
C**   +--------------------------------------------------------+    **
  500 Muse=MIN0(ID(3),2)
      ID(1)=ID(2)
      ID(2)=ID(3)
      ID(3)=MAX0(ID(3)-1,0)
      T=T+DT
C**   +--------------------------------------------------------+    **
C**   | Shift F_values for DT_step_size                        |    **
C**   +--------------------------------------------------------+    **
      DO 510 I=1,N
        X(I,1)=X(I,2)
        X(I,2)=X(I,3)
```

```
        X(I,3)=X(I,4)
  510 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Shift F_values for DT/2_step_size & DT/4_step_size     |    **
C**   +--------------------------------------------------------+    **
      IF(ID(1).GE.1) THEN
        DO 520 I=1,N
        X(I,10)=X(I,11)
  520   CONTINUE
      ENDIF
      IF(ID(2).EQ.1) THEN
        DO 530 I=1,N
        X(I,11)=X(I,12)
  530   CONTINUE
      ELSEIF(ID(2).GE.2) THEN
        DO 540 I=1,N
        X(I,11)=X(I,12)
        X(I,13)=X(I,15)
        X(I,14)=X(I,16)
  540   CONTINUE
      ENDIF
      RETURN
      END
************************************************************************
      SUBROUTINE ADAMS(N,FF,DT,T,YI,YO,F3,F2,F1,FI,FC,FP,YC,YP
     *                          ,MAXC,MUSC,EPSreq,EPSold)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION YI(N),YO(N),F3(N),F2(N),F1(N)
      DIMENSION FI(N),FC(N),FP(N),YC(N),YP(N)
      EXTERNAL FF
C**   ------------------------------------------------------------    **
C**   Input  T,YI(N),FI(N) at T=TI & F3(N),F2(N),F1(N) at T-3DT,...   **
C**   Output T,YO(N)       at T=TO=TI+DT                              **
C**   YI(N) & YO(N) can be the same location : Loc.YO(N)=Loc.YI(N)    **
C**   ------------------------------------------------------------    **
C*I   N      = Number of dependent variables                          **
C*S   FF     = Subroutine to compute F(N) for given T & Y(N)          **
C*I   DT     = Time_step = Final_time - Initial_time                  **
CIO   T      = Initial time TI & Final time TO=TI+DT                  **
C*I   YI(N)  = y_i(TI)                                                **
C*O   YO(N)  = y_i(TO)                                                **
C*I   F3(N),F2(N),F1(N),FI(N) = f_i(Y3,T3),..,f_i(Y1,T1),f_i(YI,TI)   **
C*W   FC(N),YC(N) = f_i(YC,TO),y_i(TO) Corrector                      **
C*W   FP(N),YP(N) = f_i(YP,TO),y_i(TO) Predictor                      **
C*I   MAXC   = Additional corrector cycles allowed                    **
C*O   MUSC   = Actual corrector cycles used                           **
C*I   EPSreq = Req. epsilon such that Max.|dYC/YC(i)| < EPSreq        **
C*O   EPSold = Max. epsilon = Max_{i=1}^n |1-YP(i)/YC(i)|*19/270      **
C**   ============================================================    **
      DT24=DT/24D0
      DT924=9D0*DT24
      F19270=19D0/270D0
      T=T+DT
C**   +--------------------------------------------------------+    **
C**   | Find the Predictor YP(N) & FP(N)                       |    **
C**   +--------------------------------------------------------+    **
      DO 10 I=1,N
      YP(I)=YI(I)+(55D0*FI(I)-59D0*F1(I)+37D0*F2(I)-9D0*F3(I))*DT24
   10 CONTINUE
```

```
      CALL FF(T,YP,FP,N)
C**   +-------------------------------------------------------+    **
C**   | Find the Corrector YC(N)                              |    **
C**   +-------------------------------------------------------+    **
      DO 20 I=1,N
      YC(I)=YI(I)+(9D0*FP(I)+19D0*FI(I)-5D0*F1(I)+F2(I))*DT24
   20 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Find the congergent Corrector                         |    **
C**   +-------------------------------------------------------+    **
      DO 50 MUSC=1,MAXC
      CALL FF(T,YC,FC,N)
      EPSold=0.0
      DO 30 I=1,N
      DY=(FC(I)-FP(I))*DT924
      FP(I)=FC(I)
      YC(I)=YC(I)+DY
      IF(YC(I).NE.0.0) EPSold=DMAX1(EPSold,DABS(DY/YC(I)))
   30 CONTINUE
      IF(EPSold.LE.EPSreq.OR.MUSC.GE.MAXC) GO TO 60
   50 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Find accepted YO from YC and YP                       |    **
C**   +-------------------------------------------------------+    **
   60 EPSold=0.0
      DO 70 I=1,N
      DY=(YC(I)-YP(I))*F19270
      YO(I)=YC(I)-DY
      IF(YO(I).NE.0.0) EPSold=DMAX1(EPSold,DABS(DY/YO(I)))
   70 CONTINUE
C**   CALL FF(T,YO,FC,N)
      RETURN
      END
*************************************************************************
```

習題

1. 求下列常微分方程式之數值解。設$\xi = 0.7$，$\omega = 125.6637\,\text{rad/sec}$，$g(t) = \sin(2t)$。

$$\frac{d^2y}{dt^2} + 2\xi\omega\frac{dy}{dt} + \omega^2 y = \omega^2 g(t)$$

$$y(0) = 0.0, \qquad y'(0) = 0.0$$

2. 求上題之常微分方程式之數值解。但$g(t)$改為以直線聯接下表所列各點$(t_i,\ g_i)$之函數。

i	0	1	2	3	4	5	6	7
t_i	0.0	0.1	0.16	0.20	0.24	0.30	0.36	0.40
g_i	0.0	0.3	-0.25	0.40	-0.10	-0.10	0.00	0.00

3. 試以初始值問題求下列常微分方程式之數值解並與其後所列之解析解比較。設$y_o = 1$，$v_o = 2$。

$$\frac{d^2y}{dt^2} - 4t\frac{dy}{dt} + (4t^2 - 1)y = e^{t^2}$$
$$y(0) = y_o, \qquad y'(0) = v_o$$
$$y(t) = e^{t^2}(1 + (y_o - 1)\cos t + v_o \sin t)$$

4. 試解下列常微分方程式之邊界值問題。

$$t^2\frac{d^2y}{dt^2} + t\frac{dy}{dt} + t^2y = 0$$
$$y(1) = 1, \qquad y(9) = 0$$

5. 試以威爾森法（設$\theta = 1.4$）寫一程式直接分析二階常係數微分方程式。

6. 試將威爾森法之程式修改為以四階亞當氏法直接分析二階常係數微分方程式。

參考文獻

1. Stoer, J. and Bulirsch, R. *Introduction to Numerical Analysis,* New York, Springer-Veriag, 1980.

第十三章

偏微分方程之數值解法

13.1 前言

微分方程式以自變數之多寡區分二大類：只有一個自變數的為常微分方程；二個或二個以上自變數的為偏微分方程；二者均為工程上常會遇上之問題。本章僅介紹偏微分方程之數值解法；常微分方程之數值解法則另以一章介紹。工程應用上偏微分方程式之常用自變數可歸納為二類：一為表示空間位置之變數如一維之x至三維之x, y, z，或極座標之r, θ, ϕ；另一為時間變數t。僅含空間變數而不含時間變數之偏微分方程式通常處理穩態或靜態之問題；兼含時間變數者通常處理暫態或動態之問題。對應於空間變數通常會有一定的考慮範圍或稱區域(Region)，及其邊界(Boundary)。但是區域不一定為有限，其對應邊界則會在無窮遠處。在邊界上通常會給定函數值或其微分值，稱為邊界條件。對應於時間變數通常會給定某一時間之函數值或其微分值，稱為初始條件。偏微分方程之階次(Order)為微分方程式中之最高微分階數。偏微分方程式中，所有因變數及其各階微分均為線性者稱為線性(偏微分方程式)；只有最高階微分為線性者為準線性；否則即為非線性。本章之數值解法仍以差分法為主，並以二階之偏微分方程為例說明有關之做法。對於二階偏微分方程式常依其特性分為(1)橢圓型，(2)拋物線型，與(3)雙曲線型。其中橢圓型之偏微分方程式常僅含空間位置之變數，以差分法求解一般不會有數值不穩定之現象；而拋物線型與雙曲線型之偏微分方程式常兼含時間變數，以差分法求解則可能會有數值不穩定之現象。因此這三種類型將予分別討論。

13.2 二階偏微分方程式之分類

二階偏微分方程式一般可分為三種類型：(1)橢圓型；(2)拋物線型；(3)雙曲線型。本節將略述這三種類型之分法。考慮下列之準線性二階偏微分方程式：

$$a\,\frac{\partial^2 u}{\partial x^2} + b\,\frac{\partial^2 u}{\partial x \partial y} + c\,\frac{\partial^2 u}{\partial y^2} = d \tag{13.1}$$

式中a, b, c, d為u, $\partial u/\partial x$, $\partial u/\partial y$, x, y之函數。現考慮在(x, y)平面之某一已知曲線C上各點之u, $\partial u/\partial x$, $\partial u/\partial y$為已知。並設曲線上之該等值滿足下列關係(否則$\partial u/\partial x$, $\partial u/\partial y$即非u之偏微分值)：

$$du = \frac{\partial u}{\partial x}\,dx + \frac{\partial u}{\partial y}\,dy \tag{13.2}$$

式(13.1)與下列二式，組成了以$\partial^2 u/\partial x^2$, $\partial^2 u/\partial x \partial y$, $\partial^2 u/\partial y^2$為未知數之線性聯立方程式。注意若聯立式之係數矩陣之行列式值等於零，即式(13.5)成立時，則聯立式無解。

$$d\left(\frac{\partial u}{\partial x}\right) = \frac{\partial^2 u}{\partial x^2}\,dx + \frac{\partial^2 u}{\partial x \partial y}\,dy \tag{13.3}$$

$$d\left(\frac{\partial u}{\partial y}\right) = \frac{\partial^2 u}{\partial x \partial y}\,dx + \frac{\partial^2 u}{\partial y^2}\,dy \tag{13.4}$$

$$\begin{vmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{vmatrix} = a\,(dy)^2 - b\,dx\,dy + c\,(dx)^2 = 0 \tag{13.5}$$

式(13.5)正好為dy/dx之二次多項式。因此若$b^2 - 4ac > 0$，則有二個方向之dy/dx可使式(13.5)成立；若$b^2 - 4ac = 0$，則有一個方向之dy/dx可使式(13.5)成立；若$b^2 - 4ac < 0$，則無一方向可使式(13.5)成立。若一曲線在各點之切線方向均與式(13.5)所得之方向一致時，該曲線稱特性曲線。因此$b^2 - 4ac > 0$時有二組特性曲線；$b^2 - 4ac = 0$時有一組特性曲線；$b^2 - 4ac < 0$時沒有特性曲線。

根據上述分析，在(x, y)平面之某一區域內，若各點之$b^2 - 4ac > 0$，則偏微分方程式屬於雙曲線型；若$b^2 - 4ac = 0$，則方程式屬於拋物線型；若$b^2 - 4ac < 0$，則方程式屬於橢圓型。

若偏微分方程式為非線性，例如$a=1,\ b=0,\ c=u$，則方程式之屬性與其解u有關：即若某點之解$u>0$，則該點屬於橢圓型；若其解$u<0$，則該點屬於雙曲線型。若偏微分方程式為線性，即$a,\ b,\ c$僅為$x,\ y$之函數，則方程式在各點之屬性與方程式之解無關，可於解題之前預知。若偏微分方程式之最高階係數，即$a,\ b,\ c$，均為常數，則方程式之屬性與位置無關，即整個區域為同一屬性。

(1)以下之波森(Poisson)方程式 與拉卜拉斯(Laplace)方程式(當$\rho=0$時) 為橢圓型偏微分方程式。其邊界條件可以為式(13.7)之Dirichlet型，或為式(13.8)之Neumann型，亦可為其他較複雜之型式如式(13.9)。惟S_u與S_n應包含全部邊界且不重複。式中$u_n=\partial u/\partial n$為$u$在邊界垂直方向之微分。

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=\rho(x,y) \tag{13.6}$$

$$u\ (x,y)=f(x,y)\ ,\qquad \text{於邊界}\ S_u \tag{13.7}$$

$$u_n(x,y)=g(x,y)\ ,\qquad \text{於邊界}\ S_n \tag{13.8}$$

$$a\,u(x,y)+b\,u_n(x,y)=h(x,y) \tag{13.9}$$

(2)以下之線性偏微分方程式(13.10)為拋物線型之一般式。其初始條件為式(13.11)，邊界條件為式(13.12)。式中$a,\ b,\ c,\ d$為(x,t)之函數，考慮之範圍為$R\ [0\le x\le L;\ 0\le t\le T]$。

$$\frac{\partial u}{\partial t}-a^2\frac{\partial^2 u}{\partial x^2}+b\frac{\partial u}{\partial x}+c\,u=d \tag{13.10}$$

$$u(x,0)=f(x)\ ,\qquad 0<x<L \tag{13.11}$$

$$u(0,t)=g_o(t)\ ,\quad u(L,t)=g_1(t)\ ,\quad t>0 \tag{13.12}$$

(3)以下之波方程式(13.13)為雙曲線型偏微分方程式。其初始條件為式(13.14)，邊界條件為式(13.15)與式(13.16)。式中$a_o,\ b_o,\ a_1,\ b_1,\ c$為常數，考慮之範圍為$R\ [0\le x\le L;\ 0\le t\le T]$。

$$\frac{\partial^2 u}{\partial t^2}=c^2\frac{\partial^2 u}{\partial x^2} \tag{13.13}$$

$$u=f(x)\,,\quad \frac{\partial u}{\partial t}=g(x)\,,\quad 0<x<L\,,\ \ t=0 \tag{13.14}$$

$$a_o\,u+b_o\frac{\partial u}{\partial x}=g_o(t)\,,\qquad x=0\,,\ \ t>0 \tag{13.15}$$

$$a_1\,u+b_1\frac{\partial u}{\partial x}=g_1(t)\,,\qquad x=L\,,\ \ t>0 \tag{13.16}$$

13.3 拋物線型方程式之數值解法

考慮下列之線性偏微分方程式 (13.17)，及初始條件式(13.18)，與邊界條件式(13.19)。

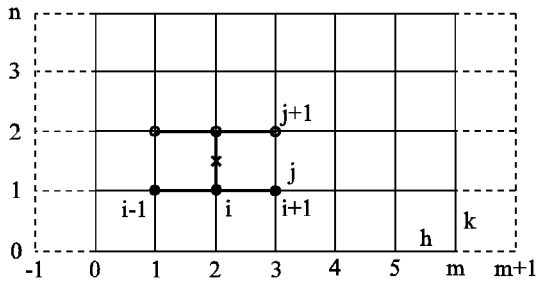
$$u_t - a^2 u_{xx} = d \tag{13.17}$$

$$u(x,0) = f(x)\,, \quad 0 < x < L \tag{13.18}$$

$$u(0,t) = g_o(t)\,, \quad u(L,t) = g_1(t)\,, \quad t > 0 \tag{13.19}$$

其中a, d為(x,t)之函數，$u_t = \partial u/\partial t$，$u_{xx} = \partial^2 u/\partial x^2$。現對考慮之範圍$R\,[0 \le x \le L; 0 \le t \le T]$劃分為$m \times n$之網格，並令$h = L/m,\ k = T/n$，$x_i = ih,\ t_j = jk$，$f_i^j = f(x_i, t_j)$。$f_i^j$表示網點上各種函數值，非網點上之函數值假設以網點之值用線性內插計算如下式$(0 \le \theta \le 1)$：

$$f(x_i, t_j + \theta k) \equiv f_i^{j+\theta} = (1-\theta) f_i^j + \theta f_i^{j+1} \tag{13.20}$$



圖一 差分式之網點

以下將函數之各偏微分值以泰勒級數展開：

$$u_t(x_i, t_j + \theta k) = \frac{u_i^{j+1} - u_i^j}{k} + \delta_t \tag{13.21}$$

$$u_{xx}(x_i, t_j + \theta k) = \frac{u_{i+1}^{j+\theta} - 2u_i^{j+\theta} + u_{i-1}^{j+\theta}}{h^2} + \delta_{xx} \tag{13.22}$$

式中

$$\delta_t = \frac{1-2\theta}{2} k u_{tt} + \frac{1 - 3\theta + 3\theta^2}{6} k^2 \bar{u}_{ttt} \tag{13.23}$$

$$\delta_{xx} = \frac{h^2}{12} \bar{u}_{xxxx} \tag{13.24}$$

式(13.21)之關係可由$u(x_i, t_{j+1})$與$u(x_i, t_j)$分別對$u(x_i, t_j + \theta k)$做泰勒式展開如式(13.25)與式(13.26)，二式相減後除以k即得。式(13.23)至式(13.28)中之$\bar{u}, \tilde{u}, \hat{u}$等，為在考慮區間內某點之值，按均值定理可表示省略項之

值。因此$\tilde{u}_{ttt}(1-\theta)^3k^3/3!+\hat{u}_{ttt}\theta^3k^3/3!$介於$\tilde{u}_{ttt}(1-3\theta+3\theta^2)k^3/3!$與$\hat{u}_{ttt}(1-3\theta+3\theta^2)k^3/3!$之間。

$$\begin{aligned} u(x_i,t_{j+1}) &= u(x_i,t_j+\theta k)+u_t(x_i,t_j+\theta k)(1-\theta)k \\ &\quad +u_{tt}(x_i,t_j+\theta k)(1-\theta)^2k^2/2!+\tilde{u}_{ttt}(1-\theta)^3k^3/3! \qquad (13.25)\\ u(x_i,t_j) &= u(x_i,t_j+\theta k)-u_t(x_i,t_j+\theta k)\theta k \\ &\quad +u_{tt}(x_i,t_j+\theta k)\theta^2k^2/2!-\hat{u}_{ttt}\theta^3k^3/3! \qquad (13.26)\end{aligned}$$

式(13.22)之關係可由$u(x_{i+1},t_j+\theta k)$與$u(x_{i-1},t_j+\theta k)$分別對$u(x_i,t_j+\theta k)$做泰勒式展開如式(13.27)與式(13.28)，二式相加後除以h^2即得。

$$\begin{aligned} u(x_{i+1},t_j+\theta k) &= u(x_i,t_j+\theta k)+u_x(x_i,t_j+\theta k)h+u_{xx}(x_i,t_j+\theta k)h^2/2! \\ &\quad +u_{xxx}(x_i,t_j+\theta k)h^3/3!+\tilde{u}_{xxxx}h^4/4! \qquad (13.27)\\ u(x_{i-1},t_j+\theta k) &= u(x_i,t_j+\theta k)-u_x(x_i,t_j+\theta k)h+u_{xx}(x_i,t_j+\theta k)h^2/2! \\ &\quad -u_{xxx}(x_i,t_j+\theta k)h^3/3!+\hat{u}_{xxxx}h^4/4! \qquad (13.28)\end{aligned}$$

現考慮將式(13.21)與式(13.22)之差分式代入式(13.17)，省略$\delta_t-a^2\delta_{xx}$之高階微分項，可得：

$$\frac{u_i^{j+1}-u_i^j}{k}-(a_i^{j+\theta})^2\,\frac{u_{i+1}^{j+\theta}-2u_i^{j+\theta}+u_{i-1}^{j+\theta}}{h^2}=d_i^{j+\theta} \qquad (13.29)$$

或
$$u_i^{j+1}-u_i^j-\alpha\,(u_{i+1}^{j+\theta}-2u_i^{j+\theta}+u_{i-1}^{j+\theta})=k\,d_i^{j+\theta} \qquad (13.30)$$

式中
$$\alpha=\frac{k}{h^2}\,(a_i^{j+\theta})^2 \qquad (13.31)$$

上式中之$u^{j+\theta}$引用式(13.20)可得：

$$\begin{aligned} &-\theta\alpha u_{i+1}^{j+1}+(1+2\theta\alpha)u_i^{j+1}-\theta\alpha u_{i-1}^{j+1} \\ &=(1-\theta)\alpha u_{i+1}^j+(1-2(1-\theta)\alpha)u_i^j+(1-\theta)\alpha u_{i-1}^j+k\,d_i^{j+\theta} \qquad (13.32)\end{aligned}$$

當$\theta=0$時可得式(13.33)。該式屬於顯式差分式，因利用該式可由$t=t_j$時之各已知值，即等式右邊之值，直接計算$t=t_{j+1}$時之u_i^{j+1}(即與$t=t_{j+1}$時之其他函數值無關)。

$$u_i^{j+1}=\alpha u_{i+1}^j+(1-2\alpha)u_i^j+\alpha u_{i-1}^j+k\,d_i^j \qquad (13.33)$$

當$\theta=\frac{1}{2}$時可得式(13.34)，該式稱為Crank-Nicolson差分式。該式屬於隱式差分式，因由$t=t_j$時之各已知值，計算$t=t_{j+1}$時之u_i^{j+1}時，需列出

$i=1,2,\ldots,m-1$之聯立式，方可由此聯立式解得。但該聯立式之係數矩陣為三對角矩陣，故極容易求解，詳[表一]程式。利用$\theta=\frac{1}{2}$之好處為此差分式省略之高階誤差項$\delta_t-a^2\delta_{xx}=\frac{1}{24}k^2\bar{u}_{ttt}-\frac{1}{12}h^2a^2\bar{u}_{xxxx}=O(k^2)+O(h^2)$較$\theta\neq\frac{1}{2}$時之誤差項$\delta_t-a^2\delta_{xx}=\frac{1-2\theta}{2}ku_{tt}+\cdots=O(k)+O(h^2)$為小(當$k\to 0,\ h\to 0$時)。

$$-\alpha u_{i+1}^{j+1}+(2+2\alpha)u_i^{j+1}-\alpha u_{i-1}^{j+1}=\alpha u_{i+1}^{j}+(2-2\alpha)u_i^{j}+\alpha u_{i-1}^{j}+2kd_i^{j+\frac{1}{2}} \quad (13.34)$$

如果式(13.19)之邊界條件改為$u_x(0,t)=g_o(t)$時，可將x之範圍增加一個網點x_{-1}。亦即將$i=0$視為內部網點，可用式(13.33)或式(13.34)列出計算式，而邊界條件採用二階精度之中心差分式：$u_1^j-u_{-1}^j=2h\,g_o^j$。

採用式(13.34)之差分式除捨去之高階誤差項較小外，不管α採用何值，均不會有數值不穩定之現象，或稱該差分式為無條件穩定。而採用式(13.33)之顯式差分式，則當$\alpha>\frac{1}{2}$時，會有數值不穩定之現象，因此該差分式為有條件穩定，其穩定條件為$\alpha\le\frac{1}{2}$。有關之穩定分析請詳次節。

13.4 穩定條件之分析

數值穩定條件之分析，為考慮計算過程之捨入誤差是否有逐步放大之現象，因此與d_i^j及邊界條件無關，故以下討論時假設$d_i^j=0$，$u_o^j=g_o^j=0$，$u_m^j=g_1^j=0$。並為簡化分析，假設a為常數，即α為常數。則式(13.33)與式(13.34)可分別寫成(以$m=5$為例)：

$$\begin{Bmatrix} u_1^{j+1}\\ u_2^{j+1}\\ u_3^{j+1}\\ u_4^{j+1}\end{Bmatrix}=\begin{bmatrix} 1-2\alpha & \alpha & & \\ \alpha & 1-2\alpha & \alpha & \\ & \alpha & 1-2\alpha & \alpha\\ & & \alpha & 1-2\alpha\end{bmatrix}\begin{Bmatrix} u_1^{j}\\ u_2^{j}\\ u_3^{j}\\ u_4^{j}\end{Bmatrix}$$

或
$$u^{j+1}=A\,u^j \quad (13.35)$$

$$\begin{bmatrix} 2+2\alpha & -\alpha & & \\ -\alpha & 2+2\alpha & -\alpha & \\ & -\alpha & 2+2\alpha & -\alpha\\ & & -\alpha & 2+2\alpha\end{bmatrix}\begin{Bmatrix} u_1^{j+1}\\ u_2^{j+1}\\ u_3^{j+1}\\ u_4^{j+1}\end{Bmatrix}=\begin{bmatrix} 2-2\alpha & \alpha & & \\ \alpha & 2-2\alpha & \alpha & \\ & \alpha & 2-2\alpha & \alpha\\ & & \alpha & 2-2\alpha\end{bmatrix}\begin{Bmatrix} u_1^{j}\\ u_2^{j}\\ u_3^{j}\\ u_4^{j}\end{Bmatrix}$$

或
$$B\,u^{j+1}=C\,u^j \quad (13.36)$$

故
$$u^{j+1}=B^{-1}C\,u^j=A\,u^j \quad (13.37)$$

由式(13.35)或式(13.37)可証知$u^j = A^j u^o$。設若u^o因捨入誤差而具有e^o之誤差，則此誤差傳播至$t = t_j$時，u^j之誤差變成$e^j = A^j e^o$。若將e^o寫成矩陣A之特徵向量v_i之線性組合，即$e^o = \sum_{i=1}^{m-1} c_i v_i$，則$e^j = A^j e^o = \sum_{i=1}^{m-1} c_i A^j v_i = \sum_{i=1}^{m-1} c_i \lambda_i^j v_i$。當$j$很大時，如欲使$e^j$不趨近於無窮大，則須所有之特徵值$\lambda_i$之絕對值均不大於1。令$T$為對稱之三對角矩陣，其對角元素全為2，上下次對角元素全為-1，則其特徵值為$4\sin^2 \frac{i\pi}{2m}$。注意式(13.35)中之A矩陣可寫成$[I - \alpha T]$，而式(13.37)中之A矩陣可寫成$[2I + \alpha T]^{-1}[2I - \alpha T]$，因此該二式中之$A$矩陣之特徵值分別為：

$$\lambda_i = 1 - 4\alpha \sin^2 \frac{i\pi}{2m} \tag{13.38}$$

$$\lambda_i = \frac{2 - 4\alpha \sin^2 \frac{i\pi}{2m}}{2 + 4\alpha \sin^2 \frac{i\pi}{2m}} \tag{13.39}$$

因α為正值，由式(13.39)可知其所有之$|\lambda_i|$均小於1。故式(13.34)之隱式差分為無條件穩定。而由式(13.38)之$|\lambda_i| \le 1$之條件可得式(13.40)。因α為正值，該式之上限一定滿足，因此由下限之條件，可得式(13.41)之條件。由此可知$\alpha \le \frac{1}{2}$時，式(13.33)之顯式差分式為穩定。

$$-1 \le 1 - 4\alpha \sin^2 \frac{i\pi}{2m} \le 1 \tag{13.40}$$

$$\alpha \le \tfrac{1}{2} \sin^{-2} \frac{i\pi}{2m} \tag{13.41}$$

對於空間為二維或三維之情形，設$h_x = h_y = h_z = h$，對應於式(13.33)之顯式差分式分別如下式，其穩定條件為$\alpha \le \frac{1}{4}$與$\alpha \le \frac{1}{6}$。

$$u_{p,q}^{j+1} = (1 - 4\alpha)\, u_{p,q}^j + \alpha(u_{p+1,q}^j + u_{p-1,q}^j + u_{p,q+1}^j + u_{p,q-1}^j) + k\, d_{p,q}^j \tag{13.42}$$

$$\begin{aligned} u_{p,q,r}^{j+1} = (1 - 6\alpha) u_{p,q,r}^j &+ \alpha(u_{p+1,q,r}^j + u_{p-1,q,r}^j \\ &+ u_{p,q+1,r}^j + u_{p,q-1,r}^j + u_{p,q,r+1}^j + u_{p,q,r-1}^j) + k\, d_{p,q,r}^j \end{aligned} \tag{13.43}$$

13.5 橢圓型方程式之數值解法

考慮下列之波森(Poisson)方程式(13.44)， 設其邊界條件為式(13.45)之Dirichlet型。考慮之區域為$R\,[0 \le x \le L_x;\, 0 \le y \le L_y]$。

$$u_{xx} + u_{yy} = \rho(x, y) \tag{13.44}$$

$$u(0, y) = u(L_x, y) = u(x, 0) = u(x, L_y) = 0 \tag{13.45}$$

現對考慮之區域$R\ [0 \le x \le L_x;\ 0 \le y \le L_y]$劃分為$m \times n$之網格，並令$h = L_x/m,\ k = L_y/n$，$x_i = ih,\ y_j = jk$，$f_{i,j} = f(x_i, y_j)$。$f_{i,j}$表示網點上各種函數值。利用式(13.22)之關係，令$\theta = 0$可得：

$$u_{xx}(x_i, y_j) = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{h^2}{12}\,\bar{u}_{xxxx} \tag{13.46}$$

$$u_{yy}(x_i, y_j) = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} + \frac{k^2}{12}\,\bar{u}_{yyyy} \tag{13.47}$$

將式(13.46)與式(13.47)之差分式代入式(13.44)，省略$\frac{h^2}{12}\,\bar{u}_{xxxx} + \frac{k^2}{12}\,\bar{u}_{yyyy}$之高階微分項，可得：

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = \rho_{i,j} \tag{13.48}$$

通常取$h = k$，可得較單純的差分式如下：

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = h^2\rho_{i,j} \tag{13.49}$$

上式應用於區域內部之點，即$i = 1, 2, \ldots, m-1$，$j = 1, 2, \ldots, n-1$，總共有$(m-1)(n-1)$個方程式，正好與未知函數$u_{i,j}$之個數相等，解此線性聯立方程式即可求得各$u_{i,j}$值。以下為$m = n = 4$之聯立方程式$(A\,u = b)$：

$$\begin{bmatrix}
4 & -1 & & -1 & & & & & \\
-1 & 4 & -1 & & -1 & & & & \\
 & -1 & 4 & & & -1 & & & \\
-1 & & & 4 & -1 & & -1 & & \\
 & -1 & & -1 & 4 & -1 & & -1 & \\
 & & -1 & & -1 & 4 & & & -1 \\
 & & & -1 & & & 4 & -1 & \\
 & & & & -1 & & -1 & 4 & -1 \\
 & & & & & -1 & & -1 & 4
\end{bmatrix}
\begin{Bmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{Bmatrix}
=
\begin{Bmatrix} -h^2\rho_{11} \\ -h^2\rho_{21} \\ -h^2\rho_{31} \\ -h^2\rho_{12} \\ -h^2\rho_{22} \\ -h^2\rho_{32} \\ -h^2\rho_{13} \\ -h^2\rho_{23} \\ -h^2\rho_{33} \end{Bmatrix} \tag{13.50}$$

該線性聯立方程式除可用第三章之方法直接求解外，亦可用如下之高斯賽德反覆試算法 (Gauss–Seidel iteration) 求解。其中$u_{i,j}^{(k)}$為第k次試算後之結果，初始值$u_{i,j}^{(o)}$可設為零或其他接近答案之值。注意式(13.51)之上標有一部分為(k)，有一部分為$(k+1)$，以使等式右邊永遠採用最新之值(設按$i = 1, 2, \ldots, m-1,\ j = 1, 2, \ldots, n-1$之順序計算)。注意在計算過程中各$u_{i,j}$值可僅保留最新值，而於$u_{i,j}^{(k+1)}$算出後立即取代$u_{i,j}^{(k)}$，如此

則於計算式(13.51)時，即可不管其上標。其後為對應之矩陣式，可寫成 $[L+D]\,u^{(k+1)} = -[U]\,u^{(k)} + b$ ，其中 D 為取自上式 A 矩陣之對角元素之對角矩陣， L 為取自 A 矩陣之對角以下元素之下三角矩陣， U 為取自 A 矩陣之對角以上元素之上三角矩陣。注意 $L+D+U=A$ 。

$$u_{i,j}^{(k+1)} = \tfrac{1}{4}(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)}) - \tfrac{1}{4}h^2\rho_{i,j} \qquad (13.51)$$

$$\begin{bmatrix} 4 & & & & & & & & \\ -1 & 4 & & & & & & & \\ & -1 & 4 & & & & & & \\ -1 & & & 4 & & & & & \\ & -1 & & -1 & 4 & & & & \\ & & -1 & & -1 & 4 & & & \\ & & & -1 & & & 4 & & \\ & & & & -1 & & -1 & 4 & \\ & & & & & -1 & & -1 & 4 \end{bmatrix} \begin{Bmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{Bmatrix} = \begin{Bmatrix} u_{21}+u_{12}-h^2\rho_{11} \\ u_{31}+u_{22}-h^2\rho_{21} \\ u_{32}-h^2\rho_{31} \\ u_{22}+u_{13}-h^2\rho_{12} \\ u_{32}+u_{23}-h^2\rho_{22} \\ u_{33}-h^2\rho_{32} \\ u_{23}-h^2\rho_{13} \\ u_{33}-h^2\rho_{23} \\ -h^2\rho_{33} \end{Bmatrix}$$

另外亦可用如下之 ADI 法(Alternating–Direction–Implicit)求解。

$$-u_{i+1,j}^{(k+1)} + 4u_{i,j}^{(k+1)} - u_{i-1,j}^{(k+1)} = u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - h^2\rho_{i,j} \qquad (13.52)$$

$$-u_{i,j+1}^{(k+1)} + 4u_{i,j}^{(k+1)} - u_{i,j-1}^{(k+1)} = u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} - h^2\rho_{i,j} \qquad (13.53)$$

以式(13.52)求某一 j 值之各 $u_{i,j},\ i=1,2,\ldots,m-1$ 時，須解一組三對角矩陣之聯立方程式。式(13.52)之 j 由1起解至 $n-1$ 後，再改用式(13.53)之 i 由1起解至 $m-1$ ，如此即完成一回合之試算。以下為式(13.52)之矩陣式：

$$\begin{bmatrix} 4 & -1 & & & & & & & \\ -1 & 4 & -1 & & & & & & \\ & -1 & 4 & & & & & & \\ -1 & & & 4 & -1 & & & & \\ & -1 & & -1 & 4 & -1 & & & \\ & & -1 & & -1 & 4 & & & \\ & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{bmatrix} \begin{Bmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{Bmatrix} = \begin{Bmatrix} u_{12}-h^2\rho_{11} \\ u_{22}-h^2\rho_{21} \\ u_{32}-h^2\rho_{21} \\ u_{13}-h^2\rho_{12} \\ u_{23}-h^2\rho_{22} \\ u_{33}-h^2\rho_{22} \\ -h^2\rho_{13} \\ -h^2\rho_{23} \\ -h^2\rho_{23} \end{Bmatrix}$$

比較前二個矩陣式，可知 ADI 法較高斯賽德法容易收斂：因為在每列4個非對角元素中，前者包含3個，而後者僅包含2個。類似方法用於三維

之問題時，每列有6個非對角元素，前者包含4個，而後者僅包含3個，ADI法之優點就沒有像二維之問題那麼顯著。

高斯賽德法之收斂速度經常很慢，效率上常比直接解法為低，因此該法常須配合一些加速之手段，如將新值之增量放大ω倍之SOR法(Simultaneous Over–Relaxation)，即採用之新值為$u_{i,j}=u_{i,j}^{(k)}+\omega\,(u_{i,j}^{(k+1)}-u_{i,j}^{(k)})$。式中$\omega$通常取大於1小於2之值，其大小對收斂速度之影響很大，可針對處理之問題及網格數量經由試算決定。下列之值可供參考(該值亦用於[表二]之程式)：

$$\omega=\frac{2}{1+\sqrt{1-\lambda_J^2}}\,,\qquad \lambda_J=\frac{k^2\cos\frac{\pi}{m}+h^2\cos\frac{\pi}{n}}{h^2+k^2} \tag{13.54}$$

上式中之λ_J為矩陣$[D]^{-1}[L+U]$之最大特徵值之絕對值，此矩陣得自另一反覆試算法稱賈柯比(Jacobi)法：寫成矩陣式為$[D]\,u^{(k+1)}=-[L+U]\,u^{(k)}+b$。令試算$k$次後之近似解$u^{(k)}$之誤差為$e^{(k)}=u^{(k)}-u$，將$A\,u=b$改為$[L+D+U]\,u=b$，再與反覆試算式相減，可得$e^{(k+1)}=-[D]^{-1}[L+U]\,e^{(k)}$。可知誤差$e^{(k)}$約以$\lambda_J$之比率遞減，即$||e^{(k+s)}||\approx\lambda_J^s\,||e^{(k)}||$。如果要$e^{(k+s)}$之準確位數比$e^{(k)}$者多$p$位十進位數值，所需之反覆試算次數$s$可由$\lambda_J^s\approx10^{-p}$估計，因此得$s\approx p\ln10/(-\ln\lambda_J)$。高斯賽德法之$\lambda_{GS}=\lambda_J^2$為$[L+D]^{-1}[U]$之最大特徵值。$SOR$法採用$\omega$加速後之$\lambda_{SOR}=(\frac{\lambda_J}{1+\sqrt{1-\lambda_J^2}})^2$。當$h=k$，且$m=n$均很大時，由式(13.54)可得$\lambda_J=\cos\frac{\pi}{m}\approx1-\frac{\pi^2}{2m^2}$；$\lambda_{GS}=\lambda_J^2\approx1-\frac{\pi^2}{m^2}$；$\lambda_{SOR}\approx1-\frac{2\pi}{m}$。反覆試算次數則約為：$s_J\approx\frac{p\ln10}{-\ln\lambda_J}\approx\frac{2pm^2\ln10}{\pi^2}\approx\frac{pm^2}{2}$；$s_{GS}\approx\frac{pm^2\ln10}{\pi^2}\approx\frac{pm^2}{4}$；$s_{SOR}\approx\frac{pm\ln10}{2\pi}\approx\frac{pm}{3}$。

13.6 不規則邊界之處理

當邊界點不與正規網點一致時，可考慮以下列方式之一處理：

(1)將邊界移至正規網點，當網點間距h很小時，此種近似方式之誤差不大，但因可採用正規網點之差分式，精度為較高之$O(h^2)$，也許可以彌補移點之誤差，此方式可以不改變程式。

(2)將網點移至邊界(圖二(a))，此時靠近邊界內點之差分式應改如下式：

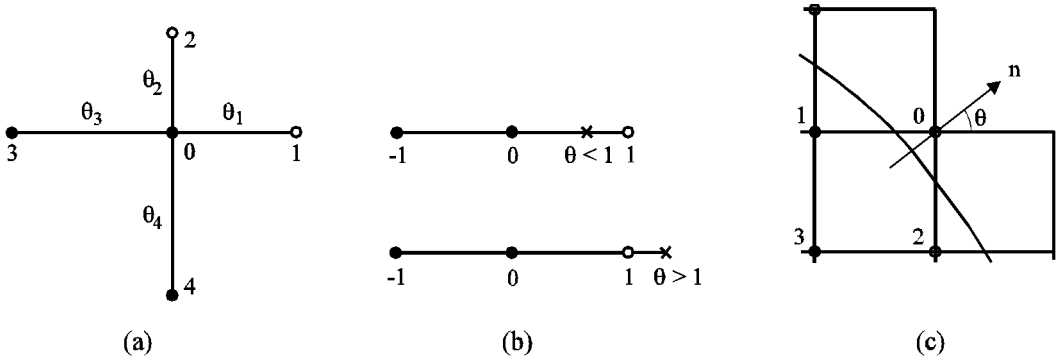
$$u_x=\frac{u_1-u_o}{\theta_1 h}+O(h)\,,\qquad u_x=\frac{u_o-u_3}{\theta_3 h}+O(h) \tag{13.55}$$

$$u_x = \frac{\theta_3}{\theta_1+\theta_3}\frac{u_1-u_o}{\theta_1 h} + \frac{\theta_1}{\theta_1+\theta_3}\frac{u_o-u_3}{\theta_3 h} + O(h^2) \tag{13.56}$$

$$u_{xx} = \frac{2}{h^2}\left[\frac{u_1-u_o}{\theta_1(\theta_1+\theta_3)} + \frac{u_3-u_o}{\theta_3(\theta_1+\theta_3)}\right] + O(h) \tag{13.57}$$

$$u_{yy} = \frac{2}{h^2}\left[\frac{u_2-u_o}{\theta_2(\theta_2+\theta_4)} + \frac{u_4-u_o}{\theta_4(\theta_2+\theta_4)}\right] + O(h) \tag{13.58}$$

$$u_{xx}+u_{yy} = \frac{2}{h^2}\Big[\frac{u_1}{\theta_1(\theta_1+\theta_3)} + \frac{u_2}{\theta_2(\theta_2+\theta_4)} + \frac{u_3}{\theta_3(\theta_1+\theta_3)} + \frac{u_4}{\theta_4(\theta_2+\theta_4)} - (\frac{1}{\theta_1\theta_3}+\frac{1}{\theta_2\theta_4})u_o\Big] + O(h) \tag{13.59}$$



圖二 邊界網點之處理

(3)以邊界附近之正規網點做為邊界網點(圖二(b))，邊界網點之u_1視為未知數。然後用正規網點$(u_{-1},\, u_o,\, u_1)$插值計算邊界點之值，令其等於邊界點已知值u_θ以做為邊界條件式，如式(13.60)；或由正規網點$(u_{-1},\, u_o)$與邊界點已知值u_θ插值計算邊界網點u_1之值，如式(13.61)。注意如邊界網點在邊界點之外時$\theta < 1$；如邊界網點在邊界點之內時$\theta > 1$。最好取較靠近邊界點之正規網點做邊界網點，則$\frac{1}{2} \le \theta \le \frac{3}{2}$。

$$u_\theta = \frac{\theta(\theta+1)}{2}\,u_1 + (1-\theta^2)\,u_o + \frac{\theta(\theta-1)}{2}\,u_{-1} + O(h^3) \tag{13.60}$$

$$u_1 = \frac{2}{\theta(\theta+1)}\,u_\theta - \frac{2(1-\theta)}{\theta}\,u_o + \frac{1-\theta}{1+\theta}\,u_{-1} + O(h^3) \tag{13.61}$$

注意方式(3)與方式(2)之式(13.57)之結果相同。因為由式(13.61)代入$u_{xx} = (u_1 - 2u_o + u_{-1})/h^2$可得式(13.62)，而該式與式(13.57)中令$\theta_1 = \theta,\, \theta_3 = 1,\, u_1 = u_\theta,\, u_3 = u_{-1}$所得之計算式相同。但程式之處理方式不同：方式(2)

為改變內部網點之計算式，可參考[表二]；方式(3)為增加邊界網點之計算式。

$$u_{xx} = \frac{u_1 - 2u_o + u_{-1}}{h^2} = \frac{2}{h^2}\left[\frac{1}{\theta(\theta+1)}\,u_\theta - \frac{1}{\theta}\,u_o + \frac{1}{1+\theta}\,u_{-1}\right] + O(h) \quad (13.62)$$

若邊界條件為在邊界之垂直方向之微分值時，情形將更為複雜，需用到較多網點之值以列出誤差小之計算式，以下僅列出一階精度之差分式供參考。設邊界之垂直方向與x軸之夾角為θ(圖二(c))。

$$\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial n} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial n} = \frac{u_o - u_1}{h}\cos\theta + \frac{u_o - u_2}{h}\sin\theta \quad (13.63)$$

$$u_o = \frac{h}{\cos\theta + \sin\theta}\frac{\partial u}{\partial n} + \frac{\cos\theta}{\cos\theta + \sin\theta}\,u_1 + \frac{\sin\theta}{\cos\theta + \sin\theta}\,u_2 \quad (13.64)$$

13.7 雙曲線型方程式之數值解法

考慮下列之線性偏微分方程式(13.65)，及初始條件式(13.66)，與邊界條件式(13.67)。

$$u_{tt} = c^2\,u_{xx} \quad (13.65)$$

$$u(x,0) = f(x)\,, \quad u_t(x,0) = g(x)\,, \quad 0 < x < L \quad (13.66)$$

$$u(0,t) = g_o(t)\,, \quad u(L,t) = g_1(t)\,, \quad t > 0 \quad (13.67)$$

其中c為常數，$u_{tt} = \partial^2 u/\partial t^2$，$u_{xx} = \partial^2 u/\partial x^2$。現對考慮之範圍$R$ $[0 \le x \le L; 0 \le t \le T]$劃分為$m \times n$之網格，並令$h = L/m$, $k = T/n$，$x_i = ih$, $t_j = jk$，$f_i^j = f(x_i, t_j)$。f_i^j表示網點上各種函數值。利用式(13.46)之類似關係，式(13.65)可寫成式(13.68)，若取h, k使$ck/h = 1$，可使差分式大為簡化，如式(13.69)。

$$u_i^{j+1} - 2u_i^j + u_i^{j-1} = (\frac{ck}{h})^2\,(u_{i+1}^j - 2u_i^j + u_{i-1}^j) \quad (13.68)$$

$$u_i^{j+1} = u_{i+1}^j + u_{i-1}^j - u_i^{j-1} \quad (13.69)$$

注意式(13.69)需用到j與$j-1$二時間段之函數值以求得$j+1$時之函數值。因此為了求得u_i^1之值，除了由初始條件可得$u_i^o = f(x_i)$外，還需有u_i^{-1}之值，該值可用另一初始條件$u_t(x_i, 0) = g(x_i)$求得，其二階中心差分式為：

$$\frac{u_i^1 - u_i^{-1}}{2k} = g(x_i)\,, \quad \text{或} \quad u_i^{-1} = u_i^1 - 2k\,g(x_i) \quad (13.70)$$

將上式代入式(13.69)，取$j=0$，即可解得：

$$u_i^1 = \tfrac{1}{2}\,(u_{i+1}^o + u_{i-1}^o) + k\,g(x_i) \tag{13.71}$$

由式(13.71)求得j=1之函數值u_i^1後，$j>0$以後之u_i^{j+1}即可依次由式(13.69)求得。注意$u_o^j = g_o(t_j)$與$u_m^j = g_1(t_j)$。如邊界條件含微分項，可仿拋物線型之方式，將x之範圍增加一個網點x_{-1}或x_{m+1}，將i=0與i=m視為內部網點，再以中心差分式$(u_1^j - u_{-1}^j)/h$近似邊界條件之$u_x(x_o,t_j)$，於i=m之邊界點為$u_x(x_m,t_j) = (u_{m+1}^j - u_{m-1}^j)/h$。

13.8 雙曲線型方程式之解析解

式(13.65)之波方程式之解析解型式相當簡單，且有明顯的物理意義，值得在此略作說明。式(13.72)為其解析解之一般式，$F(x+ct)$與$G(x-ct)$為任一函數。將式(13.72)之$u(x,t)$之偏微分直接代入式(13.65)可証明其確實滿足波方程式。

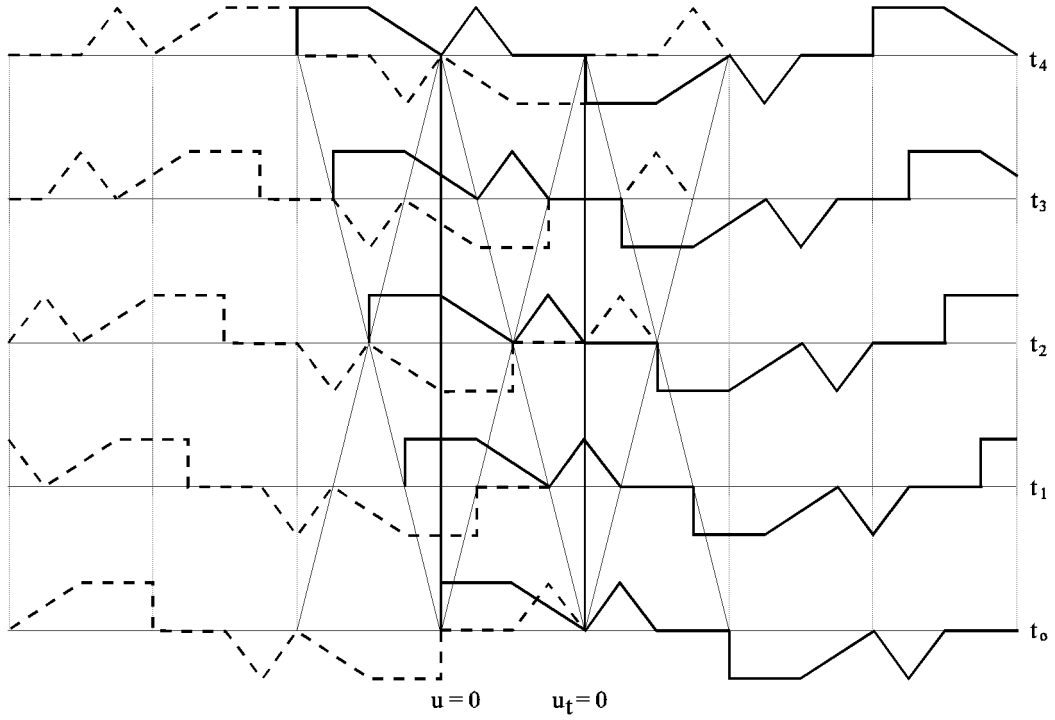
$$\begin{aligned}
u(x,t) &= F(x+ct) + G(x-ct) && (13.72)\\
\frac{\partial u}{\partial t} &= F'\frac{\partial(x+ct)}{\partial t} + G'\frac{\partial(x-ct)}{\partial t} = c\,F' - c\,G' && (13.73)\\
\frac{\partial^2 u}{\partial t^2} &= c^2\,F'' + c^2\,G'' && (13.74)\\
\frac{\partial u}{\partial x} &= F'\frac{\partial(x+ct)}{\partial x} + G'\frac{\partial(x-ct)}{\partial x} = F' + G' && (13.75)\\
\frac{\partial^2 u}{\partial x^2} &= F'' + G'' && (13.76)
\end{aligned}$$

注意$F(x+ct)$表示一個以速度c向左移動的波，當$t=t_o$之波形為$F(x+ct_o)$時，則經過Δt時間後之波形$F(x+c\Delta t+ct_o)$，會等於其右方距離$c\Delta t$之處於$t=t_o$時之波形。同理$G(x-ct)$為一個以速度c向右移動的波。任何型式的初始擾動均會形成二種波形，一個向左移動，另一個則向右移動。以下為根據式(13.66)之初始條件所得之波形：

$$u(x,t) = \frac{1}{2}\,[f(x+ct) + f(x-ct)] + \frac{1}{2c}\int_{x-ct}^{x+ct} g(\xi)\,d\xi \tag{13.77}$$

令$g(x)$之積分為$\bar{g}(x)$，亦即$\bar{g}'(x) = g(x)$，則上式對應於式(13.72)之$F(x+ct) = \frac{1}{2}\,[f(x+ct)+\frac{1}{c}\,\bar{g}(x+ct)]$，$G(x-ct) = \frac{1}{2}\,[f(x-ct)-\frac{1}{c}\,\bar{g}(x-ct)]$。可知其滿

足式(13.65)。又式(13.77)令$t=0$可得$u(x,0)=f(x)$，式(13.77)對t偏微分可得$u_t(x,t)=\frac{1}{2}[\,[c\,f'(x+ct)-c\,f'(x-ct)]+\frac{1}{2c}\,[c\,g(x+ct)+c\,g(x-ct)]$，令$t=0$可得$u_t(x,0)=g(x)$。可知其滿足初始條件。另外向左移動的波$F(x+ct)$碰到左邊界時，會有部分或全部的波反射回來，變成向右移動的波$\tilde{F}(x-ct)$。此反射波可想像來自左邊界之左邊區域，與左移的波在左邊界相遇，如二波之和滿足左邊界條件，則此反射波即為正確之反射波。若邊界條件為$u(0,t)=g_o(0,t)=0$，即$u(0,t)=F(0+ct)+\tilde{F}(0-ct)=0$，可得$\tilde{F}(-ct)=-F(ct)$。若邊界條件為$u_x(0,t)=0$，即$u_x(0,t)=F'(0+ct)+\tilde{F}'(0-ct)=0$，可得$\tilde{F}'(-ct)=-F'(ct)$，此式相當於$\tilde{F}(-ct)=F(ct)$。同理向右移動的波碰到右邊界時，會反射回來，變成向左移動的波$\tilde{G}(x+ct)$。若邊界條件為$u(L,t)=G(L-ct)+\tilde{G}(L+ct)=0$，可得$\tilde{G}(L+ct)=-G(L-ct)$。若邊界條件為$u_x(L,t)=G'(L-ct)+\tilde{G}'(L+ct)=0$，可得$\tilde{G}'(L+ct)=-G'(L-ct)$，此式相當於$\tilde{G}(L+ct)=G(L-ct)$。



圖三 波的左右平移與反射

利用上述之解析解，可以証明：當有連續二個時間段之u_i^j與u_i^{j-1}為正確值時，則由式(13.69)所求得之u_i^{j+1}亦為正確值(除捨入誤差外)。因$ck/h=1$，即$ck=h$，故得$x_i+ct_j=ih+jck=(i+j)h$，$x_i-ct_j=ih-jck=$

$(i-j)h$ 。若 u_i^j 為正確值，亦即 $u_i^j = F(x_i+ct_j) + G(x_i-ct_j) = F((i+j)h) + G((i-j)h)$ ，則由式(13.69)所得之 $u_i^{j+1} = u_{i+1}^j + u_{i-1}^j - u_i^{j-1} = [F((i+1+j)h) + G((i+1-j)h)] + [F((i-1+j)h) + G((i-1-j)h)] - [F((i+j-1)h) + G((i-j+1)h)] = F((i+j+1)h) + G((i-j-1)h) = F(x_i+ct_{j+1}) + G(x_i-ct_{j+1})$ ，亦為正確值。另外利用式(13.77)之積分式可求得較由式(13.71)為精確之 u_i^1 如下式：

$$\begin{aligned} u_i^1 = u(x_i, t_1) &= \frac{1}{2}[f(x_i+ct_1) + f(x_i-ct_1)] + \frac{1}{2c}\int_{x_i-ct_1}^{x_i+ct_1} g(\xi)\,d\xi \\ &= \frac{1}{2}[u_{i+1}^o + u_{i-1}^o] + \frac{1}{2c}\int_{x_{i-1}}^{x_{i+1}} g(\xi)\,d\xi \end{aligned} \tag{13.78}$$

13.9 雙曲線型方程式之特性曲線法

雙曲線型方程式有二組特性曲線，本節將說明一種沿著該特性曲線求解之方法。考慮式(13.1)之準線性二階偏微分方程式，但將式中自變數 y 改為 t 。首先由式(13.3)與式(13.4)解 $\frac{\partial^2 u}{\partial x^2}$ 與 $\frac{\partial^2 u}{\partial t^2}$ 可得下式：

$$\frac{\partial^2 u}{\partial x^2} = \frac{d}{dx}\left(\frac{\partial u}{\partial x}\right) - \frac{\partial^2 u}{\partial x \partial t}\frac{dt}{dx} \tag{13.79}$$

$$\frac{\partial^2 u}{\partial t^2} = \frac{d}{dt}\left(\frac{\partial u}{\partial t}\right) - \frac{\partial^2 u}{\partial x \partial t}\frac{dx}{dt} \tag{13.80}$$

上式代入式(13.1)可得式(13.81)，該式乘以 $-dt/dx$ 即得式(13.82)。

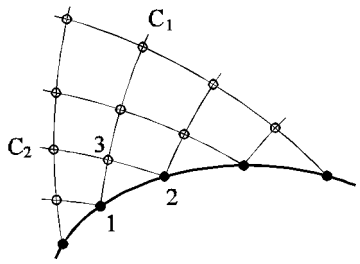
$$-a\frac{\partial^2 u}{\partial x \partial t}\frac{dt}{dx} + b\frac{\partial^2 u}{\partial x \partial t} - c\frac{\partial^2 u}{\partial x \partial t}\frac{dx}{dt} + a\frac{d}{dx}\left(\frac{\partial u}{\partial x}\right) + c\frac{d}{dt}\left(\frac{\partial u}{\partial t}\right) = d \tag{13.81}$$

$$\frac{\partial^2 u}{\partial x \partial t}\left[a\left(\frac{dt}{dx}\right)^2 - b\left(\frac{dt}{dx}\right) + c\right] - \frac{1}{dx}\left[a\frac{dt}{dx}\,d\left(\frac{\partial u}{\partial x}\right) + c\,d\left(\frac{\partial u}{\partial t}\right) - d\,dt\right] = 0 \tag{13.82}$$

上式中第一括號項若為零時，第二括號項亦必須為零，方可使式(13.1)成立。因雙曲線型方程式之 $b^2 - 4ac > 0$ ，故有二個不相等之 $m = dt/dx$ 使第一括號項為零，設該二 dt/dx 為 m_1 與 m_2 ，因此在二特性曲線上(即曲線 C_1 之切線方向為 m_1 ；曲線 C_2 之切線方向為 m_2)下式必須成立：

$$a\,m_1\,d\left(\frac{\partial u}{\partial x}\right) + c\,d\left(\frac{\partial u}{\partial t}\right) - d\,dt = 0\,, \quad m_1\,dx = dt\,, \quad \text{在曲線} C_1 \text{上} \tag{13.83}$$

$$a\,m_2\,d\left(\frac{\partial u}{\partial x}\right) + c\,d\left(\frac{\partial u}{\partial t}\right) - d\,dt = 0\,, \quad m_2\,dx = dt\,, \quad \text{在曲線} C_2 \text{上} \tag{13.84}$$



圖四 特性曲線

故若欲求點3之u_x^3與u_t^3，可由曲線C_1上之已知函數點1，與曲線C_2上之另一已知點2，分別由式(13.83)與式(13.84)列出下列一階近似之關係式。

$$a\,m_1\,(u_x^3-u_x^1)+c\,(u_t^3-u_t^1)-d\,(t^3-t^1)=0\,,\quad m_1\,(x^3-x^1)=(t^3-t^1)$$
$$a\,m_2\,(u_x^3-u_x^2)+c\,(u_t^3-u_t^2)-d\,(t^3-t^2)=0\,,\quad m_2\,(x^3-x^2)=(t^3-t^2)$$

由上二式求出u_x^3與u_t^3後即可由$du=u_x\,dx+u_t\,dt$求點3之u^3，如下式：

$$u^3=u^1+\frac{1}{2}(u_x^3+u_x^1)(x^3-x^1)+\frac{1}{2}(u_t^3+u_t^1)(t^3-t^1)$$
$$u^3=u^2+\frac{1}{2}(u_x^3+u_x^2)(x^3-x^2)+\frac{1}{2}(u_t^3+u_t^2)(t^3-t^2)$$

注意如函數在點1不連續，則在點3亦不連續，可沿特性曲線二側之不同函數值分別計算。如函數在點2也不連續，則在點3之上下左右會有不同數值，亦均可由特性曲線二側之不同函數值分別計算。由此可知函數之不連續點會沿著特性曲線移動。亦注意式(13.83)與式(13.84)相當於自變數為t，因變數為u_x與u_t之聯立一階常微分方程式，但屬於邊界值問題，即點3處由式(13.83)所得之u_x與u_t須與由式(13.84)所得者相等。

［表一］ 解拋物線型方程式之程式

```
************************************************************************
      PROGRAM PARTST
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION U(101,100),BL(100),BR(100),D(101,100)
      DIMENSION C(201),B(101) ! C(2*NXMAX-1),B(NXMAX)
      DATA NXMAX,NTMAX/101,100/
C**   ============================================================    **
      READ(*,'(2I4,4F8.0,I4,3F8.0)') NX,NT,DX,DT,AXT,DXT,IBC,G0,G1,U0
      IF(NX.GT.NXMAX.OR.NT.GT.NTMAX) STOP 'NX.GT.NXMAX.OR.NT.GT.NTMAX'
      DO 10 I=1,NX
   10 U(I,1)=U0
C     READ(*,'(10F8.0)') (U(I,1),I=1,NX)
      A=AXT**2*DT/DX**2
```

```
      IF(IBC.EQ.1.OR.IBC.EQ.3) G0=G0*DX
      IF(IBC.EQ.2.OR.IBC.EQ.3) G1=G1*DX
      DO 30 J=1,NT
      DO 20 I=1,NX
      D(I,J)=DXT*DT
   20 CONTINUE
      BL(J)=G0
      BR(J)=G1
   30 CONTINUE
      CALL PARABE(U,BL,BR,D,A,IBC,C,B,NX,NT,NXMAX)
      DO 50 J=1,NT
      WRITE(*,'(1X,10F8.3)') (U(I,J),I=1,NX)
   50 CONTINUE
      STOP
      END
*********************************************************************
      SUBROUTINE PARABE(U,BL,BR,D,A,IBC,C,B,NX,NT,NXMAX)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION U(NXMAX,NT),BL(NT),BR(NT),D(NXMAX,NT)
      DIMENSION C(NX),B(NX)
C**   ============================================================    **
C**   Use Crank-Nicolson method to solve parabolic equation :         **
C**    du/dt - a**2 * d2u/dt2 = d(x,t) ; a=constant                   **
C**    u(x,0) = given (initial condition)                             **
C**    u(0,t) = g0(t)  OR  du/dx(0,t) = g0'(t)                        **
C**    u(L,t) = g1(t)  OR  du/dt(L,t) = g1'(t)                        **
C**   +------------------+---------------------------------------+    **
C**     1---2---3---4--NX   h=DX=L/(NX-1); k=DT; A=a**2*k/h**2       **
C**                         ---------------------------------     **
C**       2+2A -A           A(u1+u3)+(2-2A)u2 + 2k*d2 + A*u1          **
C**        -A 2+2A -A       A(u2+u4)+(2-2A)u3 + 2k*d3                 **
C**            -A 2+2A      A(u3+u5)+(2-2A)u4 + 2k*d4 + A*u5          **
C**   +----< C[N,N] >-----+----------< B{N} >--------------------+    **
C**    1+A -A              A(   u2)+(1- A)u1 +  k*d1 - A*h*du/dx     **
C**    -A 2+2A -A          A(u1+u3)+(2-2A)u2 + 2k*d2                 **
C**        -A 2+2A -A      A(u2+u4)+(2-2A)u3 + 2k*d3                 **
C**            -A 2+2A -A  A(u3+u5)+(2-2A)u4 + 2k*d4                 **
C**                -A 1+A  A(u4   )+(1- A)u5 +  k*d5 + A*h*du/dx     **
C**                        ---------------------------------     **
C**     1---2---3---4--NX    u0=u2-2*h*du/dx  ;  u6=u4+2*h*du/dx      **
C**   +------------------+---------------------------------------+    **
C*I   U(I,1)  = u(x,0)       ;   t=(J-1)*DT                         **
C*O   U(I,J)  = u(x,t)                                              **
C*I   BL(J)   = g0(t) or g0'(t)*DX                                  **
C*I   BR(J)   = g1(t) or g1'(t)*DX                                  **
C*I   D(I,J)  = d(x,t)*DT                                           **
C*I   A       = a(x,t)*DT/DX**2 (Assume A=constant)                 **
C*I   IBC     = 0+1+2 : boundary conditions on Left/Right ends      **
C*I           = 1 : given du/dx = g0'(t) else given u=g0(t)         **
C*I           = 2 : given du/dx = g1'(t) else given u=g1(t)         **
C*W   C(2*NX-1) = Working array for coefficient matrix              **
C*W   B(NX)   = Working array for RHS constant vector               **
C*I   NX      = Number of grid points in X (h=DX=grid spacing)      **
C*I   NT      = Number of grid points in T (k=DT=grid spacing)      **
C*I   NXMAX   = DIMENSION of U(NXMAX,NT),D(NXMAX,NT)                **
C**   ============================================================    **
      N=NX-2
      IF(IBC.EQ.1.OR.IBC.EQ.3) N=N+1
      IF(IBC.EQ.2.OR.IBC.EQ.3) N=N+1
```

```
C**   +--------------------------------------------------------+    **
C**   | Setup coefficient matrix (upper triangular of symmetric) |    **
C**   +--------------------------------------------------------+    **
      C(1)=2+A+A
      DO 20 I=2,N
      C(2*I-2)=-A
      C(2*I-1)=2+A+A
   20 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Modify 1st and/or N-th equation to make matrix symmetric |    **
C**   +--------------------------------------------------------+    **
      IF(IBC.EQ.1.OR.IBC.EQ.3) C(1)=C(1)*0.5
      IF(IBC.EQ.2.OR.IBC.EQ.3) C(2*N-1)=C(2*N-1)*0.5
C**   +--------------------------------------------------------+    **
C**   | Decompose the coefficient matrix C(N,N)                |    **
C**   +--------------------------------------------------------+    **
      CALL CBDECP(C,1,N,1)
C**   +--------------------------------------------------------+    **
C**   | Setup RHS constant vector B(N)                         |    **
C**   +--------------------------------------------------------+    **
C     WRITE(*,'(1X,10F8.3)') (U(I,1),I=1,NX)
      DO 80 J=1,NT-1
      DO 40 I=2,NX-1
      B(I)=A*(U(I-1,J)+U(I+1,J))+(2-A-A)*U(I,J)+(D(I,J)+D(I,J+1))
   40 CONTINUE                        ! D(I,J+1) = d(x,J*DT)*DT
C**   +--------------------------------------------------------+    **
C**   | Modify last equation                                   |    **
C**   +--------------------------------------------------------+    **
      IF(IBC.EQ.2.OR.IBC.EQ.3) THEN
        B(NX)=A*U(NX-1,J)+(1-A)*U(NX,J)+0.5*(D(NX,J)+D(NX,J+1))
     *       +A*(BR(J)+BR(J+1))      ! BR(J+1) = g1'(J*DT)*DX
      ELSE
        B(NX)=BR(J+1)                ! BR(J+1) = g1(J*DT)
        B(NX-1)=B(NX-1)+A*B(NX)
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Modify 1st equation and solve to get U(*,J+1) in B(1:NX) |    **
C**   +--------------------------------------------------------+    **
      IF(IBC.EQ.1.OR.IBC.EQ.3) THEN
        B(1)=A*U(2,J)+(1-A)*U(1,J)+0.5*(D(1,J)+D(1,J+1))
     *      -A*(BL(J)+BL(J+1))       ! BL(J+1) = g0'(J*DT)*DX
        CALL CBSOLX(C,B(1),1,N,1)
      ELSE
        B(1)=BL(J+1)                 ! BL(J+1) = g0(J*DT)
        B(2)=B(2)+A*B(1)
        CALL CBSOLX(C,B(2),1,N,1)
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Move B(1:NX) to U(1:NX,J+1) (include boundary)          |    **
C**   +--------------------------------------------------------+    **
      DO 60 I=1,NX
   60 U(I,J+1)=B(I)
C     WRITE(*,'(1X,10F8.3)') (U(I,J+1),I=1,NX)
   80 CONTINUE
      RETURN
      END
*************************************************************************
```

［表二］ 解橢圓型方程式之程式

```
************************************************************************
      PROGRAM ELPTST
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION U(51,51),P(51,51),IP(51,51),BL(51),BD(51),BR(51),BU(51)
      DATA NDIM/51/
C**   ============================================================    **
C**   This program setup the data for calling ELLIP* to solve         **
C**   Poisson equation : d^2u/dx^2 + d^2u/dy^2 = p, x^2+y^2 < r^2     **
C**   with boundary condition :     u(x,y) = b  on  x^2+y^2 = r^2     **
C**   ============================================================    **
      READ(*,'(I4,5F8.0,I4)') N,R,P0,B,W,EPS,NITER
      IF(N/2*2.EQ.N.OR.N.GT.NDIM) STOP 'N IS EVEN OR N.GT.NDIM'
      H=2*R/(N-1)
      PHH=P0*H*H
C**   +----------------------------------------------------------+    **
C**   | Initialize                                               |    **
C**   +----------------------------------------------------------+    **
      DO 20 J=1,N
      DO 10 I=1,N
      U(I,J)=0
      IP(I,J)=-3
   10 CONTINUE
   20 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Boundary pt on horizontal line, set theta, U()=B, IP()=-2|    **
C**   +----------------------------------------------------------+    **
      DO 40 J=1,N
      Y=(J-N/2-1)*H
      X=DSQRT(R**2-Y**2)
      MI=X/H
      THETA=DABS(X-MI*H)/H
      IF(THETA.LT.0.333333) THEN
        THETA=1+THETA
        MI=MI-1
      ENDIF
      BL(J)=THETA
      BR(J)=THETA
      U(-MI-1+N/2+1,J)=B   ! U(-MI-1+N/2+1,J)=b(-X,Y)
      U( MI+1+N/2+1,J)=B   ! U( MI+1+N/2+1,J)=b(+X,Y)
      IP(-MI-1+N/2+1,J)=-2
      IP( MI+1+N/2+1,J)=-2
C**   +----------------------------------------------------------+    **
C**   | For interior points, set  P()=p(x,y)*h*h and IP()=0       |    **
C**   +----------------------------------------------------------+    **
      DO 30 I=-MI,MI
      P(I+N/2+1,J)=PHH     ! P(I+N/2+1,J)=p(X,Y)*H*H ; X=I*H
      IP(I+N/2+1,J)=0
   30 CONTINUE
   40 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Boundary pt on vertical line, set theta, U()=B, IP()=-2  |    **
C**   +----------------------------------------------------------+    **
      DO 50 I=1,N
      X=(I-N/2-1)*H
      Y=DSQRT(R**2-X**2)
      MJ=Y/H
      THETA=DABS(Y-MJ*H)/H
```

```
      IF(THETA.LT.0.333333) THEN
        THETA=1+THETA
        MJ=MJ-1
      ENDIF
      BD(I)=THETA
      BU(I)=THETA
      U(I,-MJ-1+N/2+1)=B   ! U(I,-MJ-1+N/2+1)=b(X,-Y)
      U(I, MJ+1+N/2+1)=B   ! U(I, MJ+1+N/2+1)=b(X,+Y)
      IP(I,-MJ-1+N/2+1)=-2
      IP(I, MJ+1+N/2+1)=-2
   50 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Calculate acceleration factor W for subroutine ELLIPT |    **
C**   +-------------------------------------------------------+    **
      IF(W.EQ.0.0) W=2/(1+DSQRT(1-DCOS(3.14159D0/(N-1))**2))
C**   +-------------------------------------------------------+    **
C**   | Compute U(I,J) and Output results                     |    **
C**   +-------------------------------------------------------+    **
      IF(NITER.NE.0) THEN
        CALL ELLIPT(U,P,IP,BL,BD,BR,BU,W,EPS,NITER,N,N,NDIM)
      ELSE
        CALL ELLIPS(U,P,IP,BL,BD,BR,BU,N,N,NDIM)
      ENDIF
      DO 80 J=N,1,-1
   80 WRITE(*,'(1X,25I3)') (IP(I,J),I=1,N)
      DO 90 J=N,1,-1
   90 WRITE(*,'(1X,10F8.4)') (U(I,J),I=1,N)
      STOP
      END
*********************************************************************
      SUBROUTINE ELLIPT(U,P,IP,BL,BD,BR,BU,W,EPS,NITER,N,M,NDIM)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION U(NDIM,M),P(NDIM,M),IP(NDIM,M),BL(M),BD(M),BR(N),BU(N)
C**   ========================================================    **
C**   Use Gauss-Seidel iterations to solve Poisson or Laplace eq.     **
C**   --------------------------------------------------------    **
C**            Output : IP(i,j)        -3 : Outside point           **
C**   M  -3 -3 -2 -2 -2 -2 -2 -3 -3    -2 : Boundary point          **
C**  (U) -3 -2  9  8  8  8 10 -2 -3     0 : Inside point            **
C**      -2  9  0  0  0  0  0 10 -2     1 : Boundary on Left  side  **
C**      -2  1  0  0  0  0  0  2 -2     2 : Boundary on Right side  **
C**   j  -2  1  0  0  0  0  0  2 -2     4 : Boundary on Down  side  **
C**      -2  1  0  0  0  0  0  2 -2     8 : Boundary on Up    side  **
C**      -2  5  0  0  0  0  0  6 -2     5 = 1+4 : Bou. on L+D side  **
C**  (D) -3 -2  5  4  4  4  6 -2 -3     6 = 2+4 : Bou. on R+D side  **
C**   1  -3 -3 -2 -2 -2 -2 -2 -3 -3     9 = 1+8 : Bou. on L+U side  **
C**       1(L)        i        (R)N    10 = 2+8 : Bou. on R+U side  **
C**   --------------------------------------------------------    **
C*I   U(I,J) = Assumed initial function values for IP=0,-2          **
C*O          = Computed function values for IP=0                    **
C*I   P(I,J) = p(x,y)*h*h (h=grid spacing) for IP=0                  **
C*I   BL(J)  = Distance_to_boundary_point / h for IP=-2              **
C*I   W      = Acceleration factor (1 <= W <= 2)                    **
C*I   EPS    = Convergence if |Unew-Uold| < |Uold|*EPS              **
C*I   NITER  = Maximum number of iterations allowed                 **
C*I   N,M    = Grid points in X/Y-dirctions                         **
C*I   NDIM   = DIMENSION of U(NDIM,M),P(NDIM,M),IP(NDIM,M)          **
C**   ========================================================    **
C**   +------------------------------------------------------+    **
```

```
C**   | For points near boundary : set IP()=(1+2+4+8)            |    **
C**   +----------------------------------------------------------+    **
      DO 20 J=1,M
      DO 10 I=1,N
      IF(IP(I,J).NE.0) GO TO 10
      IF(IP(I-1,J).EQ.-2) IP(I,J)=IP(I,J)+1
      IF(IP(I,J-1).EQ.-2) IP(I,J)=IP(I,J)+4
      IF(IP(I+1,J).EQ.-2) IP(I,J)=IP(I,J)+2
      IF(IP(I,J+1).EQ.-2) IP(I,J)=IP(I,J)+8
   10 CONTINUE
   20 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Use Gauss-Seidel iterations to solve U(I,J)              |    **
C**   +----------------------------------------------------------+    **
      DO 60 ITER=1,NITER
      NERR=0
      DO 50 J=1,M
      DO 40 I=1,N
      UOLD=U(I,J)
      K=IP(I,J)
      IF(K.LE.-2) GO TO 40
      IF(K.EQ.0) THEN
        UNEW=(U(I-1,J)+U(I,J-1)+U(I+1,J)+U(I,J+1)-P(I,J))*0.25
      ELSE IF(K.GT.0) THEN
        FL=1
        FD=1
        FR=1
        FU=1
        IF(MOD(K  ,2).EQ.1) FL=BL(J)
        IF(MOD(K/4,2).EQ.1) FD=BD(I)
        IF(MOD(K/2,2).EQ.1) FR=BR(J)
        IF(MOD(K/8,2).EQ.1) FU=BU(I)
        UNEW=(U(I-1,J)/(FL*(FL+FR))+U(I,J-1)/(FD*(FD+FU))
     *       +U(I+1,J)/(FR*(FL+FR))+U(I,J+1)/(FU*(FD+FU))
     *       -P(I,J)*0.5)*FL*FR*FD*FU/(FL*FR+FD*FU)
      ENDIF
C**   +----------------------------------------------------------+    **
C**   | Convergence control                                     |    **
C**   +----------------------------------------------------------+    **
      U(I,J)=(1-W)*UOLD+W*UNEW
      IF(DABS(U(I,J)-UOLD).GT.DABS(UOLD)*EPS) NERR=NERR+1
   40 CONTINUE
   50 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Debug output and convergence test                       |    **
C**   +----------------------------------------------------------+    **
      IF(MOD(ITER,10).EQ.1.OR.NITER-ITER.LE.10.OR.NERR.EQ.0) THEN
        WRITE(*,'('' ITER :'',I4)') ITER
        DO 55 J=N,1,-1
   55   WRITE(*,'(1X,10F8.4)') (U(I,J),I=1,N)
      ENDIF
      IF(NERR.EQ.0) RETURN
   60 CONTINUE
      RETURN
      END
*********************************************************************
      SUBROUTINE ELLIPS(U,P,IP,BL,BD,BR,BU,N,M,NDIM)
C**   ==========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION U(NDIM,M),P(NDIM,M),IP(NDIM,M),BL(M),BD(M),BR(N),BU(N)
```

```
      DIMENSION S(180000),T(3000),L(2,3000)
      DATA MAXA,MAXN/180000,3000/
C**   ============================================================    **
C**   Use direct method to solve Poisson or Laplace equation          **
C**   ------------------------------------------------------------    **
C**              Input : IP(i,j)              Output : IP(i,j)        **
C**   M  -3 -3 -2 -2 -2 -2 -2 -3 -3     -3 -3 -2 -2 -2 -2 -2 -3 -3    **
C**  (U) -3 -2  0  0  0  0  0 -2 -3     -3 -2 41 42 43 44 45 -2 -3    **
C**      -2  0  0  0  0  0  0  0 -2     -2 34 35 36 37 38 39 40 -2    **
C**      -2  0  0  0  0  0  0  0 -2     -2 27 28 29 30 31 32 33 -2    **
C**   j  -2  0  0  0  0  0  0  0 -2     -2 20 21 22 23 24 25 26 -2    **
C**      -2  0  0  0  0  0  0  0 -2     -2 13 14 15 16 17 18 19 -2    **
C**      -2  0  0  0  0  0  0  0 -2     -2  6  7  8  9 10 11 12 -2    **
C**  (D) -3 -2  0  0  0  0  0 -2 -3     -3 -2  1  2  3  4  5 -2 -3    **
C**   1  -3 -3 -2 -2 -2 -2 -2 -3 -3     -3 -3 -2 -2 -2 -2 -2 -3 -3    **
C**       1(L)        i        (R)N      1(L)        i        (R)N    **
C**   ------------------------------------------------------------    **
C*I   U(I,J) = Assumed initial function values for IP=0,-2            **
C*O          = Computed function values for IP=0                      **
C*I   P(I,J) = p(x,y)*h*h (h=grid spacing) for IP=0                    **
C*I   BL(J)  = Distance_to_boundary_point / h for IP=-2                **
C*I   N,M    = Grid points in X/Y-dirctions                           **
C*I   NDIM   = DIMENSION of U(NDIM,M),P(NDIM,M),IP(NDIM,M)            **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | For interior points : set IP() = NEQ = Equation Number   |    **
C**   +----------------------------------------------------------+    **
      NEQ=0
      DO 20 J=1,M
      DO 10 I=1,N
      IF(IP(I,J).LE.-2) GO TO 10
      NEQ=NEQ+1
      IP(I,J)=NEQ
   10 CONTINUE
   20 CONTINUE
      IF(NEQ.GT.MAXN) WRITE(*,'(I8,'' > MAXN'')') NEQ
      IF(NEQ.GT.MAXN) STOP 'DIMENSION TOO SMALL'
C**   +----------------------------------------------------------+    **
C**   | Find L(1,K) = Top  non-zero row_number                   |    **
C**   |  and L(2,K) = Left non-zero col_number                   |    **
C**   +----------------------------------------------------------+    **
      DO 30 K=1,NEQ
   30 L(1,K)=K
      DO 40 J=1,M
      DO 35 I=1,N
      K=IP(I,J)
      IF(K.LT.0) GO TO 35
      KMIN=K
      IF(IP(I-1,J).GT.0) KMIN=MIN0(KMIN,IP(I-1,J))
      IF(IP(I,J-1).GT.0) KMIN=MIN0(KMIN,IP(I,J-1))
      IF(IP(I+1,J).GT.K) L(1,IP(I+1,J))=MIN0(K,L(1,IP(I+1,J)))
      IF(IP(I,J+1).GT.K) L(1,IP(I,J+1))=MIN0(K,L(1,IP(I,J+1)))
      L(2,K)=KMIN
   35 CONTINUE
   40 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find L(1,K)/L(2,K) = K_col/K_row location index          |    **
C**   +----------------------------------------------------------+    **
      L(1,1)=0
      L(2,1)=0
```

```
      DO 50 K=2,NEQ
      L(1,K)=L(2,K-1)+(K-L(1,K))
   50 L(2,K)=L(1,K)+(K-L(2,K))
      IF(L(2,NEQ)+NEQ.GT.MAXA) WRITE(*,'(I8,'' > MAXA'')') L(2,NEQ)+NEQ
      IF(L(2,NEQ)+NEQ.GT.MAXA) STOP 'DIMENSION TOO SMALL'
C**   +--------------------------------------------------------+    **
C**   | Initialize and Form S(NEQ,NEQ) & T(NEQ)                |    **
C**   +--------------------------------------------------------+    **
      DO 60 K=1,L(2,NEQ)+NEQ
   60 S(K)=0.0
      DO 70 J=1,M
      DO 65 I=1,N
      K=IP(I,J)
      IF(K.LT.0) GO TO 65
      IF(IP(I-1,J).NE.-2.AND.IP(I,J-1).NE.-2.AND.
     *   IP(I+1,J).NE.-2.AND.IP(I,J+1).NE.-2) THEN
C**     +------------------------------------------------------+    **
C**     | All 5-points inside the boundary                     |    **
C**     +------------------------------------------------------+    **
        T(K)=-P(I,J)
        S(L(2,K)+K)=4.0
        S(L(2,K)+IP(I-1,J))=-1
        S(L(2,K)+IP(I,J-1))=-1
        S(L(1,IP(I+1,J))+K)=-1
        S(L(1,IP(I,J+1))+K)=-1
      ELSE
C**     +------------------------------------------------------+    **
C**     | Some given points on the boundary                    |    **
C**     +------------------------------------------------------+    **
        FL=1
        FD=1
        FR=1
        FU=1
        IF(IP(I-1,J).EQ.-2) FL=BL(J)
        IF(IP(I,J-1).EQ.-2) FD=BD(I)
        IF(IP(I+1,J).EQ.-2) FR=BR(J)
        IF(IP(I,J+1).EQ.-2) FU=BU(I)
        T(K)=-P(I,J)
        S(L(2,K)+K)=2/(FL*FR)+2/(FD*FU)
        IF(IP(I-1,J).EQ.-2) T(K)=T(K)+U(I-1,J)*2.0/(FL*(FL+FR))
        IF(IP(I-1,J).NE.-2) S(L(2,K)+IP(I-1,J))=-2/(FL*(FL+FR))
        IF(IP(I,J-1).EQ.-2) T(K)=T(K)+U(I,J-1)*2.0/(FD*(FD+FU))
        IF(IP(I,J-1).NE.-2) S(L(2,K)+IP(I,J-1))=-2/(FD*(FD+FU))
        IF(IP(I+1,J).EQ.-2) T(K)=T(K)+U(I+1,J)*2.0/(FR*(FL+FR))
        IF(IP(I+1,J).NE.-2) S(L(1,IP(I+1,J))+K)=-2/(FR*(FL+FR))
        IF(IP(I,J+1).EQ.-2) T(K)=T(K)+U(I,J+1)*2.0/(FU*(FD+FU))
        IF(IP(I,J+1).NE.-2) S(L(1,IP(I,J+1))+K)=-2/(FU*(FD+FU))
      ENDIF
   65 CONTINUE
   70 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Solve linear equation : S(NEQ,NEQ) * U(NEQ) = T(NEQ)   |    **
C**   +--------------------------------------------------------+    **
      CALL USDECP(S,L,NEQ,NNT,NSD,MAXA,MAXN)
      CALL USSOLX(S,T,L,NEQ,NNT,T)
C**   +--------------------------------------------------------+    **
C**   | Get function values U(I,J) from T(IP(I,J))             |    **
C**   +--------------------------------------------------------+    **
      DO 80 J=M,1,-1
      DO 75 I=1,N
```

```
      IF(IP(I,J).GT.0) U(I,J)=T(IP(I,J))
   75 CONTINUE
      WRITE(*,'(1X,10F8.4)') (U(I,J),I=1,N)
   80 CONTINUE
      RETURN
      END
*************************************************************************
```

習題

1. 求下列偏微分方程式之解析解，並繪如圖三之波之平移與反射圖。

$$
\begin{aligned}
u_{tt} &= c^2\, u_{xx} \\
u(x,0) = \sin\frac{2x\pi}{L}\,, &\quad u_t(x,0) = 0\,, \quad 0 < x < L \\
u(0,t) &= 0\,, \quad u(L,t) = 0\,, \quad t > 0
\end{aligned}
$$

   其中c為常數。注意二個移動波會合併成一個駐波，亦即有$u(x_n,t)=0$之節點$x=x_n$。

2. 試寫一程式求上題之偏微分方程式之數值解。

3. 試寫一程式求下列橢圓型方程式之數值解。

$$
\begin{aligned}
u_{xx} + u_{yy} &= 2\,, \quad x^2 + y^2 < 1 \\
u(x,y) &= 0\,, \qquad x^2 + y^2 = 1
\end{aligned}
$$

參考文獻

1. Gerald Curtis F., *Applied Numerical Analysis,* Massachusetts, Addison Wesley Publishing Co., 1972.

附錄 T 矩陣之特徵值

以下主要在証明 $(m-1)\times(m-1)$ 三對角 T 矩陣之特徵值為 $4\sin^2\frac{k\pi}{2m}=2-2\cos\frac{k\pi}{m}$，其中 $k=1,2,\ldots,m-1$。或証明下列之 $[2I-T]$ 之對稱雙次對角矩陣之特徵值為 $2-4\sin^2\frac{k\pi}{2m}=2\cos\frac{k\pi}{m}$。

$$\begin{vmatrix} -\lambda & 1 & & \\ 1 & -\lambda & 1 & \\ & 1 & -\lambda & 1 \\ & & 1 & -\lambda \end{vmatrix} = - \begin{vmatrix} \lambda & -1 & & \\ -1 & \lambda & -1 & \\ & -1 & \lambda & -1 \\ & & -1 & \lambda \end{vmatrix} = -P_4(\lambda)$$

$$\begin{aligned} P_o(\lambda) &= 1 \\ P_1(\lambda) &= \lambda P_o(\lambda) \\ P_2(\lambda) &= \lambda P_1(\lambda) - P_o(\lambda) \\ P_3(\lambda) &= \lambda P_2(\lambda) - P_1(\lambda) \\ P_4(\lambda) &= \lambda P_3(\lambda) - P_2(\lambda) \end{aligned}$$

$z^m+1=0$ 的根為 $z=\cos\frac{(2k-1)\pi}{m}+i\sin\frac{(2k-1)\pi}{m}$，$k=1,\ldots,m$；$z^m-1=0$ 的根為 $z=\cos\frac{(2k)\pi}{m}+i\sin\frac{(2k)\pi}{m}$，$k=1,\ldots,m$。二個多項式共有 $2m$ 個根，其中除有二個實根分別為 ±1 外，其餘為成對之共軛複數根，共有 $m-1$ 對。而多項式 $z^m\pm1$ 必能被其共軛複數根之二次因式 $(z^2-2\cos\frac{k\pi}{m}z+1)=(z^2-pz+1)$ 整除，即含此二次因式。現以 $m=5$ 為例：由 (z^2-pz+1) 除 $z^5\pm1$ 可得：$z^5\pm1=(z^2-pz+1)(P_o(p)z^3+P_1(p)z^2+P_2(p)z+P_3(p))+P_4(p)z-P_3(p)\pm1$。其中之 $P_o(p)$ 至 $P_4(p)$ 為 p 之多項式，將 p 改為 λ，即與前面所列之特徵多項式相同。注意若 $p=2\cos\frac{k\pi}{m}$，則因 (z^2-pz+1) 整除 $z^m\pm1$ 之一，故 $P_4(p)=0$。因此得証 $\lambda=2\cos\frac{k\pi}{m}$ 為 $P_{m-1}(\lambda)=0$ 之根。

$$\begin{array}{cccccccc} & 1 & 0 & 0 & 0 & 0 & \pm1 \\ & & pP_o & pP_1 & pP_2 & pP_3 & \\ + & & & -P_o & -P_1 & -P_2 & -P_3 & \\ \hline & P_o & P_1 & P_2 & P_3 & P_4 & -P_3\pm1 \end{array}$$

第十四章

線性規劃問題

14.1 前言

線性規劃問題 (Linear programming problem)在作業研究 (Operation research)及經濟分析上為一項重要而好用的工具，而在工程應用上，如結構最佳化問題，亦為一項基本的數學工具。 本章將介紹最常用的單一法 (Simplex method)並討論變數有上限限制時之處理方法及對應之副程式。

14.2 線性規劃問題之標準型

每一個線性規劃問題均可寫成下列之標準型：

$$\text{極小化：} \quad z = c_1\, x_1 + c_2\, x_2 + ... + c_n\, x_n \tag{14.1}$$

$$\begin{aligned} \text{受制於：} \quad & a_{11}\, x_1 + a_{12}\, x_2 + ... + a_{1n}\, x_n \;\le\; b_1 \\ & a_{21}\, x_1 + a_{22}\, x_2 + ... + a_{2n}\, x_n \;\le\; b_2 \qquad (14.2) \\ & \cdots \\ & a_{m1} x_1 + a_{m2} x_2 + ... + a_{mn} x_n \;\le\; b_m \end{aligned}$$

$$x_i \;\ge\; 0 \tag{14.3}$$

或以矩陣式表示：

$$\text{極小化：} \quad z = \langle C \rangle \{X\} \tag{14.4}$$

$$\text{受制於：} \quad [A]\{X\} \;\le\; \{B\} \tag{14.5}$$

$$\{X\} \;\ge\; \{0\} \tag{14.6}$$

式(14.1)函數稱為目標函數 (Objective function)，式(14.2)稱為限制式 (Constraint equations)，式(14.3)稱為正值限制 (Non–negativity restriction)。

所有線性規劃問題均可經簡單之改變而成上列之標準型。如極大化之目標函數可將係數$\langle C\rangle$變號而改為極小化之目標函數。如限制式為大於等於可將該式之係數$[A]$及常數$\{B\}$變號而改為小於等於。如限制式為等於可改為二個限制式，一個為小於等於，一個為大於等於(再將係數及常數變號改為小於等於)。如變數須限為負值可將其變號而成為正值之新變數，其係數為原變數之係數之異號值。如變數不限於正值或負值可將其改為二個正值變數之差，被減之新變數之係數為原來變數之係數，減數之新變數之係數為原來變數之係數之異號值。

14.3 線性規劃問題的幾何圖解

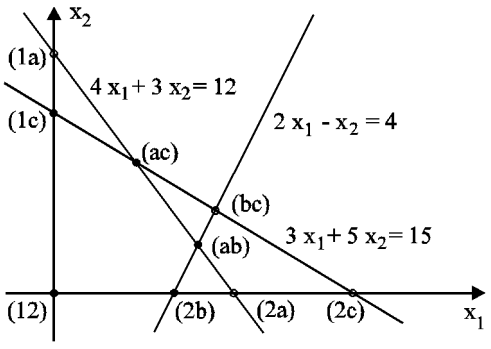
當線性規劃問題之變數之數目n超過二個時就不易用二度平面來表示，因此只用二個變數的線性規劃問題為例做圖解，並藉以說明一些有關名詞。

考慮下列線性規劃問題：

$$\text{極小化：}\quad z = -8x_1 - 4x_2 \tag{14.7}$$

$$\begin{aligned}\text{受制於：}\quad & 4x_1 + 3x_2 \le 12 \\ & 2x_1 - x_2 \le 4 \\ & 3x_1 + 5x_2 \le 15\end{aligned} \tag{14.8}$$

$$x_1 \ge 0\ ,\quad x_2 \ge 0 \tag{14.9}$$



圖一 圖解線性規劃問題

每個限制式及正值限制均將平面分成滿足限制與不滿足限制二部分。同時滿足五個限制之交集部分，見圖一中以(1c)(ac)(ab)(2b)(12)為頂點之五邊形，為一個凸形的範圍。此範圍內之點均滿足限制條件稱為合理解(Feasible solution)。對應五個頂點之解稱為基本合理解(Basic feasible solution)。其中一點(ab)之目標函數值為最小即為最佳解(Optimal solution)。

一般求解線性規劃問題時，均將不等式左邊加上一個正值變數而化為如下所列之等式限制式。該正值變數稱為鬆緊變數 (Slack variable)，除其在目標函數之係數為零外，可照一般變數處理。式(14.7)亦改寫成式(14.10)之型式。

$$-z - 8x_1 - 4x_2 = 0 \tag{14.10}$$

$$\begin{aligned} 4x_1 + 3x_2 + x_3 &= 12 \\ 2x_1 - x_2 + x_4 &= 4 \\ 3x_1 + 5x_2 + x_5 &= 15 \end{aligned} \tag{14.11}$$

式(14.11)為五個變數三個等式之五元三式聯立方程式，如不限變數為正值則有無限組解。如任選其中二個變數為零，此二變數稱為非基本變數 (Non–basic variable)，即可由聯立式求得另外三個變數之值，此三變數稱為基本變數 (Basic variable)，這些解稱為基本解 (Basic solution)。本例共有10組(五選三之組合)基本解，對應於圖一中之五條直線之十個交點。因這些解並非所有變數值均為正值，故不一定為合理解。其中只有五組解之變數值均無負值而滿足所有限制條件方為基本合理解。而最佳解則必包含在這些基本合理解中。注意合理解有無限組而基本合理解僅有有限組。即使如此，當變數及限制式增多時，基本合理解之數目仍然太多而不大可能全部求出以便從中找出最佳解。下節所要介紹的單一法為從某一個基本合理解出發，依次找出另一個會使目標函數值減少之基本合理解，一直到不能再使目標函數值減少為止，此基本合理解即為最佳解。

14.4 單一法

由式(14.11)可以明顯看出：當令$x_1 = x_2 = 0$時可得$x_3 = 12,\ x_4 = 4,\ x_5 = 15$為一組基本合理解。但由此時之目標函數，$z = -8x_1 - 4x_2$，可以看出：

1. 當x_1每增加1時目標函數減少8；
2. 當x_2每增加1時目標函數減少4。
3. 故選擇增加x_1，仍然讓$x_2=0$。即選係數為最大負值之變數為新基本變數。
4. 以後可能遇到的情況為：若目標函數之係數均非負值，則不能再使目標函數減少，即可確定已求得最佳解。

雖然增加x_1或x_2均可使目標函數減少。但通常一次只允許一個非基本變數改為基本變數而增加其值，為了方便一般均選其係數為最大負值者。當然x_1改為基本變數後必須有一個基本變數改為非基本變數。以下就是如何找到這個基本變數及決定x_1可增加到多大的步驟：

1. 由第一式：當x_1增至$\frac{12}{4}$時x_3減至零。即$\frac{12}{4}$為x_1之最大限值。
2. 由第二式：當x_1增至$\frac{4}{2}$時x_4減至零。即$\frac{4}{2}$為x_1之另一最大限值。
3. 由第三式：當x_1增至$\frac{15}{3}$時x_5減至零。即$\frac{15}{3}$為x_1之又一最大限值。
4. 以上三個最大限值以式二之$\frac{4}{2}$為最小，故得$x_1=\frac{4}{2}$成為基本變數，而第二式之$x_4=0$改為非基本變數。x_3及x_5則不一定減至零而仍為基本變數。
5. 以後可能遇到的情況為：若某一限制式之係數不為正值，則該式對新基本變數無限制作用(因新基本變數增加時，該式之基本變數亦跟著增加)。若所有限制式之係數均非正值，則新基本變數可無限增加，而目標函數亦可無限減少，所得之解稱無限解(Unbounded solution)。

為了找出其他基本變數及目標函數之值，可將第二式除以x_1之係數2使x_1之係數為1後，將$x_1=0.5x_2-0.5x_4+2$代入其餘限制式及目標函數以消去x_1之係數而得如下之關係(此項運算相當於以第二式為樞紐列做高斯消去運算)：

$$\begin{aligned} -z \quad & -8.0x_2 \quad +4.0x_4 \quad =16 \qquad (14.12)\\ & 5.0x_2+x_3-2.0x_4 \quad =4 \\ x_1 \; & -0.5x_2 \quad +0.5x_4 \quad =2 \qquad (14.13)\\ & 6.5x_2 \quad -1.5x_4+x_5=9 \end{aligned}$$

至此即完成單一法的一個運算回合。亦即已由一個基本合理解找到另一個更好的基本合理解。且由新的關係式知其解為：

$$z=-16,\ x_3=4,\ x_1=2,\ x_5=9,\ x_2=x_4=0$$

以後之運算均大致相同，僅部分情況略有不同已如前述。下一回合運算簡述如下：

由目標函數知其最大負值係數為x_2之係數-8.0，因此選x_2增加其值以改為基本變數。

由第一式知x_2之最大限值為$\frac{4}{5}$。由第二式知x_2之係數為負值故無最大限值。由第三式知x_2之最大限值為$\frac{9}{6.5}$。以第一式之$\frac{4}{5}$為最小，因此$x_2 = \frac{4}{5}$，並以第一式為樞紐列做高斯消去可得：

$$-z \qquad + 1.6x_3 + 0.8x_4 \qquad = 22.4 \tag{14.14}$$

$$x_2 + 0.2x_3 - 0.4x_4 \qquad = 0.8$$

$$x_1 \qquad + 0.1x_3 + 0.3x_4 \qquad = 2.4 \tag{14.15}$$

$$-1.3x_3 + 1.1x_4 + x_5 = 3.8 \tag{14.16}$$

至此又完成一個運算回合。由式(14.14)知其係數均無負值，故增加非基本變數之值均不會使目標函數z之值減少。因此可確定已求得最佳解為：

$$z = -22.4,\ x_2 = 0.8,\ x_1 = 2.4,\ x_5 = 3.8,\ x_3 = x_4 = 0$$

注意式(14.10)至式(14.15)均具下列特徵及用途：

(1)每個基本變數均只出現在一個限制式中，且其係數為1；其餘限制式及目標函數之係數為零。該行係數只有一個為1，其餘全為零，特稱為基元向量。

(2)限制式右邊之常數項均無負值，而等於其對應之基本變數之值。

(3)只有非基本變數可能出現在目標函數中。因此目標函數右邊之常數項變號後即為目標函數值。並可以直接由其係數大小決定改變那個非基本變數為基本變數，或判斷已求得最佳解。

(4)亦只有非基本變數可能出現在所有限制式中。利用這些係數及右邊之正值常數項即可直接決定新的基本變數之限值及新的非基本變數，或判斷解為無限解。

如將式(14.10)至式(14.15)之變數省略即可改為如下之表格，則較為簡潔。注意表中已省略z的係數行，目標函數式亦改置於最後一列。以後範例均將改以類似之表格說明。

［表一］ 例一 單一法之運算表格

4.0	3.0	1	0	0	12.0		12/4
2.0	-1.0	0	1	0	4.0	<	4/2
3.0	5.0	0	0	1	15.0		15/3
-8.0	-4.0	0	0	0	.0		

0	5.0	1	-2.0	0	4.0	<	4/5.0
1	-.5	0	.5	0	2.0		
0	6.5	0	-1.5	1	9.0		9/6.5
0	-8.0	0	4.0	0	16.0		

0	1	.2	-.4	0	.8
1	0	.1	.3	0	2.4
0	0	-1.3	1.1	1	3.8
0	0	1.6	.8	0	22.4

14.5 無初始合理解之處理－虛設變數

前面所討論的問題因每一個限制式均加入一個鬆緊變數，且等號右邊之常數項均為正值，故可以直接設定這些鬆緊變數等於其右邊之常數而為滿足正值限制之基本變數，其他變數則設定為零而為非基本變數。因此即有一組基本合理解可做為單一法的初始合理解。如果限制式右邊為負值，如下列之線性規劃問題：

極小化： $z = -8x_1 - 4x_2$ (14.17)

受制於：

$$-4x_1 - 3x_2 \leq -12$$

$$-2x_1 + x_2 \leq -4 \quad (14.18)$$

$$3x_1 + 5x_2 \leq 15$$

$$x_1 \geq 0\,,\ x_2 \geq 0 \quad (14.19)$$

[表二(a)]　例二　虛設變數

4.0	3.0	-1	0	0	1	0	12.0
2.0	-1.0	0	-1	0	0	1	4.0
3.0	5.0	0	0	1	0	0	15.0
-8.0	-4.0	0	0	0	M	M	0.0

通常須將該等式(加入鬆緊變數後)之所有係數變號。此時鬆緊變數之係數變成-1，而不能直接設定其為基本變數。亦即該式無法直接看出那一個變數可做為基本變數。此時可再加入一個正值變數x_6或x_7，如[表二(a)]，暫時令其等於等式右邊之正值常數而為基本變數。但該變數如為正值，則很明顯地該等式即不滿足原來的限制條件。為了不使此情況出現在最佳解，可以將目標函數改為如式(14.20)或[表二(a)]所示(其中M為任意非常大的定值)：

$$z = -8x_1 - 4x_2 + Mx_6 + Mx_7 \tag{14.20}$$

表面上雖然允許x_6及x_7大於零，但實際上為了使上列目標函數為最小，必然須使x_6及x_7等於零才會得到最佳解。這些變數只是臨時客串，最後不能真的有非零值，故稱為虛設變數 (Artificial variable)。為了使該表成為具有前節所述之特徵，必須將目標函數中之基本變數x_6及x_7之係數M消去。用含x_6及x_7之二列去減目標函數列即可消除x_6及x_7之係數如下：

$$z = (-8 - 6M)x_1 + (-4 - 2M)x_2 + Mx_3 + Mx_4 + 16M \tag{14.21}$$

目標函數加入含M的係數後須將含M與不含M之係數分為二列，以避免該二種係數相加後由於電腦有效位數的限制而使不含M之係數不準或被去除。含M的目標函數一般列於原來目標函數之後，且可將M提出而只用其係數運算。因此而成[表二(b)]。

[表二(b)]　例二　目標函數分為二列

4.0	3.0	-1	0	0	1	0	12.0	12/4
2.0	-1.0	0	-1	0	0	1	4.0	< 4/2
3.0	5.0	0	0	1	0	0	15.0	15/3
-8.0	-4.0	0	0	0	0	0	0.0	
-6.0	-2.0	1	1	0	0	0	-16.0	* M

以後之做法即與前節所述者相同，不過因增列目標函數須乘以非常大之 M 值，故目標函數之係數之大小當然須以增列目標函數之係數為準。一直等到增列目標函數之係數(虛設變數者除外)全為零後，才能將該列去掉而回到原來目標函數列繼續做運算。以增列目標函數為準之運算稱階段一 (Phase I)；以原來目標函數為準之運算稱階段二 (Phase II)。階段一其實僅是在找一組真正的合理解，故可稱為找合理解階段。階段二則由合理解繼續找最佳解，故可稱為找最佳解階段。

[表二(c)] 例二 找合理解階段之運算

0	5.0	-1	2.0	0	1	-2.0	4.0	< 4/5.0
1	-.5	0	-.5	0	0	.5	2.0	
0	6.5	0	1.5	1	0	-1.5	9.0	9/6.5
0	-8.0	0	-4.0	0	0	4.0	16.0	
0	-5.0	1	-2.0	0	0	3.0	-4.0	* M

0	1	-.2	.4	0	.2	-.4	.8	
1	0	-.1	-.3	0	.1	.3	2.4	
0	0	1.3	-1.1	1	-1.3	1.1	3.8	< 3.8/1.3
0	0	-1.6	-.8	0	1.6	.8	22.4	
0	0	.0	.0	0	(1	1)	.0	* M

[表二(d)] 例二 找最佳解階段之運算

0	1	0	.23	.15	1.38	< 1.38/0.23
1	0	0	-.38	.08	2.69	
0	0	1	-.85	.77	2.92	
0	0	0	-2.15	1.23	27.08	

0	4.33	1	0	.67	6.00	
1	1.67	0	0	.33	5.00	
0	3.67	0	1	1.33	8.00	
0	9.33	0	0	2.67	40.00	

找合理解階段並不一定都可找到真正的合理解。因原限制式可能為不合理(或矛盾)的條件。各種情況均可於增列目標函數之係數(虛設變數者除外)運算到均無負值時(也就是目標函數值不能再減少時)看出：

(1)如果所有係數(虛設變數者除外)均為零，則表示已找到真正的合理解

(如本例之[表二(c)])。
(2)如果至少有一個係數(虛設變數者除外)大於零，且增列目標函數之常數項小於零，則表示限制式之線性組合會得到一個如增列目標函數之限制式(不計入z及虛設變數之項)。該式為不合理之條件，因左邊之正值係數乘正值變數不可能等於右邊之負值常數。因此可確定無合理解。
(3)同2，但增列目標函數之常數項為零，則可以令增列目標函數之係數不為零之對應非基本變數永遠為零，並開始做找最佳解階段。即此階段須特別注意不能選這些永遠為零之非基本變數變為基本變數。

虛設變數既然為臨時客串，總希望其儘早變為非基本變數。它們一旦變為非基本變數，就沒有必要讓其再變為基本變數。所以在運算表中可以將虛設變數所對應的整行係數去除。一方面可以節省這些係數的儲存空間及運算時間，一方面可以避免這些變數在變成非基本變數後，又再成為基本變數(其係數可能為最大負值)。此外再注意階段一結束時以增列目標函數之係數判斷是否有合理解時，必須將虛設變數者除外(例如[表二(c)]中最後一列係數除虛設變數者為1外，其餘均為零)。故將虛設變數之整行係數去除後，可免掉除外的比較運算，反而簡潔。因此在以後的範例，即不再列出虛設變數的各行係數。

在找合理解階段，於決定新基本變數之最大限值時，亦可不理會不等式限制之虛設變數是否變為負值，因該負值變號後正好可改為鬆緊變數之值，並令虛設變數為零(如最大限值均屬不等式限制則取其中之最大值)。因此在例二中，本來最大限值有三個為$(12/4, 4/2, 15/3)$，如不考慮其中二個不使不等式限制之虛設變數變為負值之限值$(12/4, 4/2)$，則最大限值僅剩一個為$(15/3)$，由此得新基本變數之值為5，並以第三列為樞紐列做高斯消去運算可得：

	0	-3.67	-1	0	-1.33	1	0	-8.0	< 0
	0	-4.33	0	-1	-0.67	0	1	-6.0	< 0
	1	1.67	0	0	0.33	0	0	5.0	
	0	9.33	0	0	2.67	0	0	40.0	
	0	8.00	1	1	2.00	0	0	14.0	* M

注意上表中有些限制式等號右邊之常數為負值，將這些限制式變號後常數項變為正值而可改為鬆緊變數之值，同時亦須以該限制式去減增列目標函數以消去該鬆緊變數在增列目標函數之係數，即抵消原來對增加之虛設變數之處理，因此可得如下之結果：

```
+-------------------------------------------------+---------+
|   0     3.67    1       0      1.33   -1       0   |   8.0   |
|   0     4.33    0       1      0.67    0      -1   |   6.0   |
|   1     1.67    0       0      0.33    0       0   |   5.0   |
+-------------------------------------------------+---------+
|   0     9.33    0       0      2.67    0       0   |  40.0   |
|   0      .0     0       0       .0    (1       1)  |    .0   *  M
+-------------------------------------------------+---------+
```

其餘之運算過程則與前述者相同。本例則正好得到最佳解而結束運算。

14.6 基元向量之省略

注意在以上二個範例之運算表中(除增列目標函數外)，所有加入鬆緊變數之各行係數，除該鬆緊變數所在列之係數為 $+1$ 或 -1(此時增列目標函數之係數為 $+1$)外，其餘均為零。因此應可將這幾行去掉以節省空間與時間。這些行所對應的變數號碼可用 $LI(I)$ 記錄之。方法是：如果第 J 個變數之係數只有在第 I 列有個 $+1$ 或 -1，則分別令 $LI(I)$ 等於 $+J$ 或 $-J$。

當限制式為等式時，雖然前面說過可改為二個小於等於之不等式而化為標準型。但照單一法之運算方法，可直接用一個等式加上一個虛設變數即可正常操作。此時可令 $LI(I) = 0$。

注意對某一列 I 而言，其 $LI(I)$ 只能記錄一個值 $(+J, -J$ 或 $0)$，亦即只用以記錄非虛設變數。因該列之右邊常數項為正值，故當 $LI(I)$ 為正值 J 時，表示第 J 個變數為基本變數，其值等於第 I 列右邊之常數項；但當 $LI(I)$ 為零或負值時，表示第 I 列仍無真正的基本變數，必須暫時以虛設變數當做基本變數，因此照前節之說明，須用此列去減增列目標函數列以消去虛設變數之係數。亦即運算之前可用 $LI(I)$ 設定增列目標函數之係數。

沒有去掉的那些行則將不固定存放某一固定變數之係數，因此亦須用一指標 $LJ(J)$ 記錄之。方法是：將第 J 行係數所屬之變數號碼 I 記錄於 $LJ(J)$。

前節提到階段一結束時，可能須設定某些非基本變數永遠為零，這些非基本變數可用負的 $LJ(J)$ 標示之。方法是：若第 J 行係數所屬之非基本變數須永遠為零(設其變數號碼為 I)，則令 $LJ(J) = -I$。

注意在本文所附程式中，會自動將 $LI(I) = 0$ 之值改為絕對值大於 NX 之負值，NX 為真實變數及鬆緊變數之總數。因此大於 NX 之變數為等式限制之虛設變數並永遠以負值存於 $LI(*)$ 或 $LJ(*)$。

當每一回合開始找目標函數之最大負值時，應只找$LJ(J)$大於零所對應之係數做比較。

利用這些$LI(I)$及$LJ(J)$，以[表二(b)]至[表二(d)]為例可改如[表二(e)]所示。注意每一回合之運算均有一個$LI(I)$值與一個$LJ(J)$值互相交換，及正負號之改變。即原來為非基本變數之$LJ(J)$變成基本變數而做為新的$LI(I)$值；原來為基本變數之$LI(I)$(取正值)則變成非基本變數而做為新的$LJ(J)$值。請仔細比較[表二(b)]至[表二(d)]與[表二(e)]並特別注意$LI(I)$與$LJ(J)$之正負號之用法及改變，在此不予贅述。

[表二(e)]　例二　不存基元向量

```
+-------------------+-----------+-----
|   4.00     3.00   |   12.00   |  -3       12/4
|   2.00    -1.00   |    4.00   <  -4        4/2
|   3.00     5.00   |   15.00   |   5       15/3
+-------------------+-----------+-  ^
|  -8.00    -4.00   |     .00   |  LI
|  -6.00    -2.00   |  -16.00   *  M
+-----^-------------+-----------+-----
|     1        2    <--   LJ    |
```

```
+-------------------+-----------+-----
|   2.00     5.00   |    4.00   <  -3       4/5.0
|   -.50     -.50   |    2.00   |   1
|   1.50     6.50   |    9.00   |   5       9/6.5
+-------------------+-----------+-  ^
|  -4.00    -8.00   |   16.00   |  LI
|  -2.00    -5.00   |   -4.00   *  M
+-------------^-----+-----------+-----
|     4        2    <--   LJ    |
```

```
+-------------------+-----------+-----
|    .40     -.20   |     .80   |   2
|   -.30     -.10   |    2.40   |   1
|  -1.10     1.30   |    3.80   <   5       3.8/1.3
+-------------------+-----------+-  ^
|   -.80    -1.60   |   22.40   |  LI
|    .00      .00   |     .00   *  M
+-------------^-----+-----------+-----
|     4        3    <--   LJ    |
```

```
+-------------------+-----------+-----
|    .23      .15   |    1.38   <   2       1.38/0.23
|   -.38      .08   |    2.69   |   1
|   -.85      .77   |    2.92   |   3
+-------------------+-----------+-  ^
|  -2.15     1.23   |   27.08   |  LI
+-----^-------------+-----------+-----
|     4        5    <--   LJ    |
```

```
+------------------+----------+-----
|   4.33      .67  |    6.00  |   4
|   1.67      .33  |    5.00  |   1
|   3.67     1.33  |    8.00  |   3
+------------------+----------+-  ^
|   9.33     2.67  |   40.00  |  LI
+------------------+----------+-----
|     2        5   <--   LJ    |
```

為了便於參考，另外列出一個無限解及一個無合理解之運算例， 分別如[表三]及[表四]所示。

[表三] 例三 無限解

```
+------------------+----------+-----
|   4.00     3.00  |   12.00  |  -3       12/3
|   2.00    -1.00  |    4.00  |   4
|   3.00     5.00  |   15.00  <  -5       15/5
+------------------+----------+-  ^
|  -8.00    -4.00  |     .00  |  LI
|  -7.00    -8.00  |  -27.00  *  M
+-------------^----+----------+-----
|     1        2   <--   LJ    |
```

```
+------------------+----------+-----
|   2.20      .60  |    3.00  <  -3       3/2.2
|   2.60     -.20  |    7.00  |   4       7/2.6
|    .60     -.20  |    3.00  |   2       3/0.6
+------------------+----------+-  ^
|  -5.60     -.80  |   12.00  |  LI
|  -2.20     -.60  |   -3.00  *  M
+-----^------------+----------+-----
|     1        5   <--   LJ    |
```

```
+------------------+----------+-----
|   -.45      .27  |    1.36  |   1
|   1.18     -.91  |    3.45  <   4       3.45/1.18
|    .27     -.36  |    2.18  |   2       2.18/0.27
+------------------+----------+-  ^
|  -2.55      .73  |   19.64  |  LI
|    .00      .00  |     .00  *  M
+-----^------------+----------+-----
|     3        5   <--   LJ    |
```

```
+------------------+----------+-----
|    .38     -.08  |    2.69  |   1
|    .85     -.77  |    2.92  |   3
|   -.23     -.15  |    1.38  |   2
+------------------+----------+-  ^
|   2.15    -1.23  |   27.08  |  LI
+-------------^----+----------+-----
|     4        5   <--   LJ    |
```

［表四］ 例四 無合理解

```
+-------------------+-----------+-----
|   4.00     3.00   |   12.00   |   3        12/4
|   2.00    -1.00   |    4.00   <  -4         4/2
|   3.00     5.00   |   15.00   |  -5        15/3
+-------------------+-----------+-  ^
|  -8.00    -4.00   |     .00   |  LI
|  -5.00    -4.00   |  -19.00   *  M
+-----^-------------+-----------+-----
|     1        2    <--   LJ    |
```

```
+-------------------+-----------+-----
|   2.00     5.00   |    4.00   <   3        4/5.0
|   -.50     -.50   |    2.00   |   1
|   1.50     6.50   |    9.00   |  -5        9/6.5
+-------------------+-----------+-  ^
|  -4.00    -8.00   |   16.00   |  LI
|  -1.50    -6.50   |   -9.00   *  M
+--------------^----+-----------+-----
|     4        2    <--   LJ    |
```

```
+-------------------+-----------+-----
|    .40      .20   |     .80   |   2
|   -.30      .10   |    2.40   |   1
|  -1.10    -1.30   |    3.80   |  -5
+-------------------+-----------+-  ^
|   -.80     1.60   |   22.40   |  LI
|   1.10     1.30   |   -3.80   *  M
+-------------------+-----------+-----
|     4        3    <--   LJ    |
```

14.7 線性規劃問題的對偶性

對於每一個線性規劃問題，稱基本問題 (Primal problem)，均有一個對應的線性規劃問題，稱對偶問題 (Dual problem)，存在於另一組不同的變數空間。這兩個問題之最佳值在有有限解的情況會等值異號；否則為一個問題有無限解另一個問題無合理解。這兩個問題如寫成標準型時，其對稱性最明顯：

基本問題	對偶問題
極小化：$z = \{C\}^T\{X\}$	極小化：$w = \{B\}^T\{Y\}$
受制於：$[A]\{X\} \le \{B\}$	受制於：$-[A]^T\{Y\} \le \{C\}$
$\{X\} \ge \{0\}$	$\{Y\} \ge \{0\}$

(14.22)

對偶問題的對偶問題為基本問題，故稱那一個為基本問題那一個為對偶問題並不重要，只是為了說明之方便而已。

前面已說明過如何將線性規劃問題化為上列標準型。注意一個問題之限制式如為等式則其對偶問題之對應變數必為不限制正負值之變數。因等式之限制可改為二個小於等於之限制，而二者之係數異號，故其對偶問題之對應二個變數之係數亦異號，而可將二個變數相減合併為一個不限正負值之變數。

為了了解基本問題與對偶問題間之許多重要而有趣的關係，首先將變數按最佳解(當有解時)之結果分為二組：一組為基本變數(X_1, S_1)或(Y_1, K_1)，另一組為非基本變數(X_2, S_2)或(Y_2, K_2)。係數矩陣及向量亦予分組。則基本問題及對偶問題之原始關係將如[表五(a)]所示。該表經高斯消去運算後可得最佳解如[表五(b)]所示。再經去除單元向量並對調前後二行後如[表五(c)]所示。比較該表，可明顯看出其對稱性及下列之關係：

(1)當一個問題(以基本問題為例)為有限解時：
(1a)基本問題之最佳解之條件：$A_{11}^{-1}B_1 \geq 0,\ \ B_2 - A_{21}A_{11}^{-1}B_1 \geq 0$ 及 $-C_1^T A_{11}^{-1} \geq 0,\ \ C_2^T - C_1^T A_{11}^{-1}A_{12} \geq 0$ 也就是對偶問題之最佳解之條件：$-A_{11}^{-T}C_1 \geq 0,\ \ C_2 - A_{12}^T A_{11}^{-T}C_1 \geq 0$ 及 $B_1^T A_{11}^{-T} \geq 0,\ \ B_2^T - B_1^T A_{11}^{-T}A_{21}^T \geq 0$。故二個問題均為有限解。且由一個問題的解可以得到另一個問題的解。
(1b)二個問題之最佳解之目標函數值等值異號：$C_1^T A_{11}^{-1}B_1$或$-B_1^T A_{11}^{-T}C_1$。
(1c)二個問題之對應變數$(X_1, K_1), (X_2, K_2), (S_1, Y_1), (S_2, Y_2)$中，一個變數若為基本變數，則另一個變數必為非基本變數。反之亦然。亦即：
基本問題之限制式不產生作用時，即$S_2 > 0$，則對偶問題之對應變數值為零，即$Y_2 = 0$。理當如此，不是嗎？(同樣地，若$K_2 > 0$，則$X_2 = 0$)。
對偶問題之變數值大於零時，即$Y_1 > 0$，則基本問題之對應限制式為等式，即$S_1 = 0$。(同樣地，若$X_1 > 0$，則$K_1 = 0$)。

(2)當一個問題(以基本問題為例)為無限解時，即目標函數有某一個係數為負值，而其在限制式之對應係數均無正值，則其對偶問題之對應限制式為等式右邊之常數為負值，而左邊之係數均無負值(因二個問題之限制式係數矩陣互為轉置異號)，即為不可能滿足之條件，故對偶問題為無解。

[表二(f)]為[表二(e)]之對偶問題之運算過程及最佳解。比較二表將不難看出前面所提到的對偶關係。

[表五(a)] 基本問題與對偶問題之原始關係

$$\begin{array}{cccc|c} A_{11} & A_{12} & I & 0 & B_1 \\ A_{21} & A_{22} & 0 & I & B_2 \\ \hline C_1^T & C_2^T & 0 & 0 & 0 \end{array}$$

$$\begin{array}{cccc|c} -A_{11}^T & -A_{21}^T & I & 0 & C_1 \\ -A_{12}^T & -A_{22}^T & 0 & I & C_2 \\ \hline B_1^T & B_2^T & 0 & 0 & 0 \end{array}$$

[表五(b)] 基本問題與對偶問題之最佳解關係

$$\begin{array}{cccc|ccc} I & A_{11}^{-1}A_{12} & A_{11}^{-1} & 0 & A_{11}^{-1}B_1 & = & X_1 \\ 0 & A_{22}-A_{21}A_{11}^{-1}A_{12} & -A_{21}A_{11}^{-1} & I & B_2-A_{21}A_{11}^{-1}B_1 & = & S_2 \\ \hline 0 & C_2^T-C_1^TA_{11}^{-1}A_{12} & -C_1^TA_{11}^{-1} & 0 & -C_1^TA_{11}^{-1}B_1 & = & -z \\ & X_2=0 & S_1=0 & & & & \end{array}$$

$$\begin{array}{cccc|ccc} I & A_{11}^{-T}A_{21}^T & -A_{11}^{-T} & 0 & -A_{11}^{-T}C_1 & = & Y_1 \\ 0 & -A_{22}^T+A_{12}^TA_{11}^{-T}A_{21}^T & -A_{12}^TA_{11}^{-T} & I & C_2-A_{12}^TA_{11}^{-T}C_1 & = & K_2 \\ \hline 0 & B_2^T-B_1^TA_{11}^{-T}A_{21}^T & B_1^TA_{11}^{-T} & 0 & B_1^TA_{11}^{-T}C_1 & = & -w \\ & Y_2=0 & K_1=0 & & & & \end{array}$$

[表五(c)] 基本問題與對偶問題之最佳解關係–不存基元向量

$$\begin{array}{cc|ccc} A_{11}^{-1} & A_{11}^{-1}A_{12} & A_{11}^{-1}B_1 & = & X_1 \quad (K_1=0) \\ -A_{21}A_{11}^{-1} & A_{22}-A_{21}A_{11}^{-1}A_{12} & B_2-A_{21}A_{11}^{-1}B_1 & = & S_2 \quad (Y_2=0) \\ \hline -C_1^TA_{11}^{-1} & C_2^T-C_1^TA_{11}^{-1}A_{12} & -C_1^TA_{11}^{-1}B_1 & = & -z \quad (-z=w) \\ S_1=0 & X_2=0 & & & \end{array}$$

$$\begin{array}{cc|ccc} -A_{11}^{-T} & A_{11}^{-T}A_{21}^T & -A_{11}^{-T}C_1 & = & Y_1 \quad (S_1=0) \\ -A_{12}^TA_{11}^{-T} & -A_{22}^T+A_{12}^TA_{11}^{-T}A_{21}^T & C_2-A_{12}^TA_{11}^{-T}C_1 & = & K_2 \quad (X_2=0) \\ \hline B_1^TA_{11}^{-T} & B_2^T-B_1^TA_{11}^{-T}A_{21}^T & B_1^TA_{11}^{-T}C_1 & = & -w \quad (-w=z) \\ K_1=0 & Y_2=0 & & & \end{array}$$

[表二(f)] 例二之對偶問題

```
+-------------------------+----------+-----
|  -4.00   -2.00    3.00  |   8.00   | -4       8/3
|  -3.00    1.00    5.00  |   4.00   <  -5       4/5
+-------------------------+----------+-  ^
| -12.00   -4.00   15.00  |    .00   |  LI
|   7.00    1.00   -8.00  | -12.00   *  M
+-----------------------^-+----------+-----
|     1        2       3    <--   LJ    |
```

```
+-------------------------+----------+-----
|  -2.20   -2.60     .60  |   5.60   <  -4      5.6/0.6
|   -.60     .20    -.20  |    .80   |   3
+-------------------------+----------+-  ^
|  -3.00   -7.00    3.00  | -12.00   |  LI
|   2.20    2.60    -.60  |  -5.60   *  M
+-----------------------^-+----------+-----
|     1        2       5    <--   LJ    |
```

```
+-------------------------+----------+-----
|  -3.67   -4.33   -1.67  |   9.33   |   5    (k2=9.33) {x2=0}
|  -1.33    -.67    -.33  |   2.67   |   3    (y3=2.67) {s3=0}
+-------------------------+----------+-  ^
|   8.00    6.00    5.00  | -40.00   |  LI
|    .00     .00     .00  |    .00   *  M
+-------------------------+----------+-----
|     1        2       4    <--   LJ    |
   (y1=0)   (y2=0)   (k1=0)                  {-} 為基本問題之解
   {s1=8}   {s2=6}   {x1=5}                  (-) 為對偶問題之解
```

14.8 上限限制

所謂上限限制為變數除須大於等於零外，亦須受小於等於某一已知之上限值之限制。該限制如按一般限制式處理，須增加限制式之數目。本節將討論單一法如何處理這些上限限制而不必增加限制式之數目。為了有效處理上限限制，對變數做如下之規定：

(1) 當變數X_i增至其上限值D_i時，則將該變數改為新變數$\bar{X}_i$，而定義為

$$\bar{X}_i = D_i - X_i$$

因此新的變數$\bar{X}_i$等於零而變成非基本變數，並受$0 \le \bar{X}_i \le D_i$之正值限制及上限限制，將$X_i = D_i - \bar{X}_i$代入限制式及目標函數中可得新變數$\bar{X}_i$之係數為原變數X_i之係數之異號值，等號右邊之常數須減去原變數X_i之係數乘以D_i。

(2) 當新變數$\bar{X}_i$變成大於零時，則須再將該新變數$\bar{X}_i$改回原變數X_i。當

然係數及常數須做如前項所述之類似處理。

(3) 當變數為新變數時以 $P_i = -1$ 標示；若為原變數時以 $P_i = +1$ 標示。

注意按第 (2) 項規定，新變數 $\bar{X}_i$ 僅能做為非基本變數，因此 $P_i = -1$ 永遠表示新變數 $\bar{X}_i = 0$(或原變數 $X_i = D_i$)。

以下將單一法考慮上限限制之過程簡述如下：

(1) 由目標函數列找最大負值之係數以決定新基本變數。

新基本變數之決定與未考慮上限限制之情況相同，即仍以目標函數之係數為最大負值之變數為新基本變數。不過須按新基本變數是否為新變數而分為下列二種情況：(設最大負值係數在第 j 行，變數號碼為 $l = LJ(j)$)

情況 1：新基本變數為原變數 X_l。

情況 2：新基本變數為新變數 $\bar{X}_l$。但因新變數不能為基本變數，故須改為原變數後再做為基本變數。

(2) 由各限制式對新基本變數之最大限值及新基本變數之上限值 D_l 中找最小者做為新基本變數之值。按最小值之來源分為下列三種情況：(設 $k = LI(i)$)

情況 A：為限制式之係數 A_{ij} 為正值時，新基本變數之最大限值為使該限制式所屬之基本變數 X_k 減至零。即最大限值為 B_i/A_{ij}。

情況 B：為限制式之係數 A_{ij} 為負值時，新基本變數之最大限值為使該限制式所屬之基本變數 X_k 增至其上限值 D_k。即最大限值為 $(B_i - D_k)/A_{ij}$。

情況 C：為新基本變數之本身之上限值。即最大限值為 D_l。

其中情況 A 為以前未考慮變數有上限限制時之僅有一種情況。

(3) 按情況 $1, 2$ 及 A, B, C 之六種組合做下列之運算：

情況 $1A$：做高斯消去。令 $LI(i) = l$，$LJ(j) = k$。

情況 $2A$：將第 j 行係數先乘以 D_l 去減右邊常數再變號。令 $P(l)$=+1。做高斯消去。令 $LI(i) = l$，$LJ(j) = k$。

情況 $1B$：將第 i 列右邊常數減去 D_k。將變數 k 之單元向量變號。令 $P(k)$=−1。做高斯消去。令 $LI(i) = l$，$LJ(j) = k$。

注意變數 k 之單元向量未存於運算表中但高斯消去後移存於第 j 行。

情況$2B$：將第i列右邊常數減去D_k。將變數k之單元向量變號。
令$P(k)=-1$。將第j行係數先乘以D_l去減右邊常數再變號。
令$P(l)=+1$。做高斯消去。令$LI(i)=l$，$LJ(j)=k$。

情況$1C$：將第j行係數先乘以D_l去減右邊常數再變號。令$P(l)=-1$。
非基本變數為新變數X_l。

情況$2C$：將第j行係數先乘以D_l去減右邊常數再變號。令$P(l)=+1$。
非基本變數為原變數X_l。

14.9 線性規劃副程式與試用程式

在副程式$LINEAR$中，i=KI, j=KJ, k=KKI, l=KKJ。上節所述步驟(1)之運算自$C11$，步驟(2)自$C12$開始，步驟(3)自$C14$開始，其中情況$1B, 2B$之第一項運算自$C15$開始，情況$2A, 2B, 1C, 2C$對第j行係數之運算自$C16$開始，情況$1A, 2A, 1B, 2B$之高斯消去自$C17$開始。

為了減少運算之捨去誤差，於完成階段一之運算後，均重新由原始關係式開始按已求得之基本變數做高斯消去，此項運算由副程式$RECOMP$執行。為了做此項運算必須將原始關係式保留在檔案IS或陣列B中。程式會自動根據IS之值做此保留運算：IS>0時存於檔案；IS<0時存於陣列；亦可令IS=0而不做此項運算。

程式之有關輸入及輸出資料均詳列於程式之註標中，在此不另贅述。

［表一］ 線性規劃法副程式與試用程式

```
*************************************************************************
      PROGRAM LPTST
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(51,61),LI(51),LJ(61),D(110),X(110)
      DIMENSION T(51,61),MI(51),MJ(61)
      DATA NIA,NJA/51,61/,IS,IC/0,6/
C**   ============================================================    **
C**   Input data -----+------+--------------------+----------         **
C**               SEQ | SETS | VARIABLES          |  FORMAT          **
C**              -----+------+--------------------+----------         **
C**                1  |  1   | IS,IC              |  2I5             **
C**                2  |  1   | II,JJ,NY           |  3I5             **
C**                3  |  II  | (A(I,J),J=1,JJ)    |  8F10.0          **
C**                4  |  1   | (D(J),J=1,NY)      |  8F10.0          **
C**                5  |  1   | (LI(I),I=1,II-1)   |  16I5            **
C**                6  |  1   | (LJ(J),J=1,JJ-1)   |  16I5            **
C**              -----+------+--------------------+----------         **
C**   IS    = File# to save origi. tableau (no re-perform if IS=0)    **
C**   IC    = File# to save debug output (no debug output if IC=0)    **
```

```
C**   II,JJ = Numbers of rows and columns of simplex tableau          **
C**   NY    = Maximum variable # which have upper bound to input      **
C**           (may include slack var but not artificial var)          **
C**   ------------------------------------------------------------    **
C**   A(I,J)            = Simplex tableau                             **
C**   I=1,II-1;J=1,JJ-1 : For coeff. of variables : A(I,J)           **
C**   I=1,II-1;J=JJ     : For RHS constants       : B(I) = A(I,JJ)    **
C**   I=II    ;J=1,JJ-1 : For objective function  : C(J) = A(II,J)    **
C**   ------------------------------------------------------------    **
C**   D(J)  = Upper bound value for variable I                        **
C**   ------------------------------------------------------------    **
C**   LI(I) = Constaint equation type                                 **
C**         = +1  for lesser  or equal constraint                     **
C**         = -1  for greater or equal constraint                     **
C**         =  0  for equal constarint                                **
C**   LJ(J) = Variable # for column J of the table                    **
C**         = 0  will be reset to J by this program                   **
C**   ============================================================    **
   10 READ (*,'(11I5)') IS,IC
      IF(IS.GT.0) OPEN (IS,FILE='LP.TMP',STATUS='NEW'
     *                 ,FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
      READ (*,'(11I5)') II,JJ,NY
      IF(II.EQ.0.OR.II.GE.NIA.OR.JJ.GT.NJA) STOP
      DO 15 I=1,II
   15 READ (*,'(11F10.0)') (A(I,J),J=1,JJ)
      READ (*,'(11F10.0)') (D(J),J=1,NY)
      READ (*,'(11I5)')    (LI(I),I=1,II-1)
      READ (*,'(11I5)')    (LJ(J),J=1,JJ-1)
C**   +------------------------------------------------------------+    **
C**   | Comput NX and fill D(NY+1:NX)=0                            |    **
C**   | Setup LI(II-1),LJ(JJ-1),D(NX) for simplex tableau          |    **
C**   +------------------------------------------------------------+    **
      DO 20 J=1,JJ-1
      IF(LJ(J).EQ.0) LJ(J)=J
   20 CONTINUE
      NX=JJ-1
      DO 25 I=1,II-1
      IF(LI(I).EQ.0) GO TO 25
      NX=NX+1
      IF(IABS(LI(I)).EQ.1) LI(I)=ISIGN(NX,LI(I))
   25 CONTINUE
      DO 30 J=1,NX
      IF(J.GT.NY) D(J)=0.0
      IF(D(J).LE.0.0) D(J)=1.0D30
   30 CONTINUE
C**   +------------------------------------------------------------+    **
C**   | Write data before calling LP                               |    **
C**   +------------------------------------------------------------+    **
      WRITE(*,'('' ********** DATA BEFORE CALLING LINEAR **********'')')
      DO 45 I=1,II
   45 WRITE(*,'(1X,1P11E10.3)') (A(I,J),J=1,JJ)
      WRITE(*,'(1X,1P11E10.3)') (D(J),J=1,NX)
      WRITE(*,'(1X,11I5)')      (LI(I),I=1,II-1)
      WRITE(*,'(1X,11I5)')      (LJ(J),J=1,JJ-1)
C**   +------------------------------------------------------------+    **
      CALL LINEAR(A,LI,LJ,D,X,II,JJ,NX,ICASE,KKK,NIA,NJA,T,MI,MJ,IS,IC)
C**   +------------------------------------------------------------+    **
C**   | Write data after calling IDP                               |    **
C**   +------------------------------------------------------------+    **
      WRITE(*,'('' ========== DATA AFTER  CALLING LINEAR =========='')')
```

```
      DO 50 I=1,II
   50 WRITE(*,'(1X,1P11E10.3)') (A(I,J),J=1,JJ)
      WRITE(*,'(1X,1P11E10.3)') (D(J),J=1,NX)
      WRITE(*,'(1X,11I5)')      (LI(I),I=1,II-1)
      WRITE(*,'(1X,11I5)')      (LJ(J),J=1,JJ-1)
C**   +-------------------------------------------------------+    **
C**   | Output feasible optimal solution                      |    **
C**   +-------------------------------------------------------+    **
      IF(ICASE.NE.0) GO TO 10
      WRITE(*,'('' VALUES OF VARIABLES =''))')
      WRITE(*,'(1X,1P8E10.3)') (X(J),J=1,NX)
      WRITE(*,'('' OBJECTIVE FUNCTION  ='',1PE15.8)') -A(II,JJ)
      GO TO 10
      END
*************************************************************************
      SUBROUTINE LINEAR
     *(A,LI,LJ,D,X,II,JJ,NX,ICASE,KKK,NIA,NJA,T,MI,MJ,IS,IC)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NIA,NJA),LI(NIA),LJ(NJA),D(NX),X(NX)
      DIMENSION T(NIA,NJA),MI(NIA),MJ(NJA)
      DATA EPS,BIG/1.0D-6,1.0D10/
C**   ========================================================    **
C**   Linear programming with upper bound by simplex method         **
C**   --------------------------------------------------------    **
C**              1                          JJ-1   JJ              **
C**             +-------------------------------+------+           **
C**          1  |                               |      |           **
C**             |                               >=     | L(I)<0    **
C**             |              A                =   B  | L(I)=0    **
C**             |                               <=     | L(I)>0    **
C**        II-1 |                               |      |           **
C**             +-------------------------------+------+           **
C**         II  |              C                |  -Z  |           **
C**             +-------------------------------+------+           **
C**        II+1 |              C'               |  -Z' |  *  M     **
C**             +-------------------------------+------+           **
C**   --------------------------------------------------------    **
C**   Minimize: Z = \sum C(J)*X(LJ(J))                            **
C**   Subject to: \sum A(I,J)*X(LJ(J)) [-|+] X([-|+]LI(I)) = B(I) **
C**       and   :     D(K) >= X(K) >= 0                           **
C**     where:  I = 1 to II-1;    J = 1 to JJ-1;    K = 1 to NX   **
C**   --------------------------------------------------------    **
C**   Input  : A(II  ,JJ),LI(II-1),LJ(JJ-1),D(NX),II,JJ,NX,NIA,NJA,IS,IC
C**   Output : A(II+1,JJ),LI(II-1),LJ(JJ-1),D(NX),X(NX),ICASE,KKK **
C**   Working: T(II  ,JJ),MI(II-1),MJ(JJ-1)                       **
C**            NIA >= II+1;    NJA >= JJ                          **
C**   --------------------------------------------------------    **
C**   If IS.GT.0 then T unused & IS=file # to save A              **
C**   If IS.EQ.0 then T,MI,MJ unused                              **
C**   If IS.LT.0 then T is a working array to save A              **
C**   ========================================================    **
C01   +-------------------------------------------------------+    **
C**   | Call CHKOUT is for print out only, it can be deleted  |    **
C**   +-------------------------------------------------------+    **
      CALL CHKOUT(A,LI,LJ,D,LI,LJ,II,JJ,NX,NIA,NJA,IC,0,1,1)
C**   +-------------------------------------------------------+    **
C**   | KKKK,KKKO = Re-perform control variables              |    **
C**   | KKK       = Number of simplex pivot                   |    **
C**   +-------------------------------------------------------+    **
```

```
      KKKK=0
      KKKO=0
      KKK=0
C02   +-------------------------------------------------------+    **
C**   | Re-set L(I) from the input values of (-s,0,+s)         |    **
C**   | (1) Change all constraints to <= B(I), such that:     |    **
C**   |     L(I) > 0 : for slack variable X(I),      I<=NX    |    **
C**   |     L(I) < 0 : for artificial variable X(I), I> NX    |    **
C**   | (2) Make sure that D(I) >= 0 for non-artificial variable |    **
C**   +-------------------------------------------------------+    **
      NY=NX
      DO 130 I=1,II-1
      IF(LI(I).LT.0) THEN
        DO 125 J=1,JJ
        A(I,J)=-A(I,J)
  125   CONTINUE
        LI(I)=-LI(I)
      ELSE IF(LI(I).EQ.0) THEN
        NY=NY+1
        LI(I)=-NY
      ENDIF
  130 CONTINUE
C**   -------------------------------------------------------    **
      DO 150 J=1,NX
      IF(D(J).LT.0.0) STOP 1111
  150 CONTINUE
C03   +-------------------------------------------------------+    **
C**   | Save A(II,JJ) on (1) file# IS>0 or (2) T(II,JJ) if IS<0  |    **
C**   | For IS=0, A(II,JJ) will not be saved and no re-perform   |    **
C**   +-------------------------------------------------------+    **
      IF(IS.GT.0) THEN
        REWIND IS
        WRITE (IS) ((A(I,J),I=1,II),J=1,JJ)
      ELSE IF(IS.LT.0) THEN
        DO 170 J=1,JJ
        DO 165 I=1,II
        T(I,J)=A(I,J)
  165   CONTINUE
  170   CONTINUE
      ENDIF
C04   +-------------------------------------------------------+    **
C**   | Save |LI(I)| & |LJ(J)| on MI(I) & MJ(J) for re-perform   |    **
C**   +-------------------------------------------------------+    **
      IF(IS.NE.0) THEN
        DO 180 I=1,II-1
        MI(I)=IABS(LI(I))
  180   CONTINUE
        DO 190 J=1,JJ-1
        MJ(J)=IABS(LJ(J))
  190   CONTINUE
      ENDIF
C05   +-------------------------------------------------------+    **
C**   | Set up simplex tableau :                              |    **
C**   | (1) Reset A(II+1,*)=0 : Objective fun. for artifical var.|    **
C**   | (2) Set LJ(J)<0 : To disqualify J as basic variable, if  |    **
C**   |     (2a) Artificial variables  (J>NX)                 |    **
C**   |     (2b) Non-artificial variables (J<=NX) but D(J) = 0   |    **
C**   | KK=II+1 : Phase 1 objective function at row II+1      |    **
C**   | KK=II   : Phase 2 objective function at row II        |    **
C**   +-------------------------------------------------------+    **
```

```
  200 KK=II+1
      A(KK,JJ)=0.0
      DO 210 J=1,JJ-1
        A(KK,J)=0.0
        LJJ=IABS(LJ(J))
        IF(LJJ.GT.NX) THEN
          LJJ=-LJJ
        ELSE IF(D(LJJ).EQ.0.0) THEN
          LJJ=-LJJ
        ENDIF
        LJ(J)=LJJ
  210 CONTINUE
C06   +-------------------------------------------------------+    **
C**   | (3) Change B(I)=A(I,JJ) to >= 0     (for I=1 to II-1) |    **
C**   +-------------------------------------------------------+    **
      DO 270 I=1,II-1
      IF(A(I,JJ).LT.0.0) THEN
        DO 230 J=1,JJ
        A(I,J)=-A(I,J)
  230   CONTINUE
        LI(I)=-LI(I)
      ENDIF
C07   +-------------------------------------------------------+    **
C**   | (4) For artificial var (I>NX) : LI(I) must be < 0     |    **
C**   +-------------------------------------------------------+    **
      IF(LI(I).GT.NX) LI(I)=-LI(I)
C08   +-------------------------------------------------------+    **
C**   | (5) Basic var X(L(I))=A(I,JJ) must be <= D(LI(I))     |    **
C**   |     otherwise change basic var S to D-S': i.e. change :  |    **
C**   |     (5a) A(I,JJ)  to A(I,JJ)-D(LI(I))                |    **
C**   |     (5b) D(LI(I)) to -D(LI(I)) to indicate bounded var   |    **
C**   |     (5c) LI(I)    to -LI(I)  to indicate non-basic var   |    **
C**   +-------------------------------------------------------+    **
      LII=LI(I)
      IF(LII.GT.0) THEN
        IF(A(I,JJ).GT.D(LII)) THEN
          A(I,JJ)=A(I,JJ)-D(LII)
          D(LII)=-D(LII)
          LI(I)=-LII
        ENDIF
      ENDIF
C09   +-------------------------------------------------------+    **
C**   | (6) For any constraint not satisfied (LI(I) <= 0)     |    **
C**   |     substract from A(II+1,J) by A(I,J)                |    **
C**   +-------------------------------------------------------+    **
      IF(LI(I).LE.0) THEN
        DO 260 J=1,JJ
        A(KK,J)=A(KK,J)-A(I,J)
  260   CONTINUE
      ENDIF
  270 CONTINUE
C10   +-------------------------------------------------------+    **
C**   | Feasibility check                                     |    **
C**   +-------------------------------------------------------+    **
      ICASE=0
      DO 280 J=1,JJ-1
C**   +-------------------------------------------------------+    **
C**   | LJ(J)<0 indicate X(J) must be 0 & can not be a basic var |    **
C**   | So, test only for J<JJ and LJ(J)>0                    |    **
C**   | (0) Stay at phase 1 (go to 300) :   if any A(II+1,J)<0   |    **
```

```
C**   | (1) Feasible (ICASE=0, phase 2) :   if all A(II+1,J)=0   |    **
C**   | (2) May be infeasible : if not (0) and any A(II+1,J)>0   |    **
C**   +---------------------------------------------------------+    **
      IF(LJ(J).LE.0) GO TO 280
      IF(ABS(A(KK,J)).LE.EPS) GO TO 280
      IF(A(KK,J).LT.0.0) GO TO 300
      ICASE=-1
  280 CONTINUE
C**   +---------------------------------------------------------+    **
C**   | (2a) A(II+1,JJ)< 0 : infeasible (ICASE=-1 & go to 710)   |    **
C**   | (2b) A(II+1,JJ)>=0 : feasible, set LJ(J)<0 if A(II+1,J)>0|    **
C**   +---------------------------------------------------------+    **
      IF(ICASE.LT.0) THEN
        IF(A(KK,JJ).LT.-EPS) GO TO 710
        DO 285 J=1,JJ-1
        IF(A(KK,J).GT.0.0) LJ(J)=-IABS(LJ(J))
  285   CONTINUE
      ENDIF
C**   +---------------------------------------------------------+    **
C**   | Feasible case : get into phase 2 to find the optimal sol |    **
C**   +---------------------------------------------------------+    **
      ICASE=0
      KK=II
C11   +---------------------------------------------------------+    **
C**   | Begin simplex pivot  (KK=II+1:phase 1) (KK=II:phase 2)   |    **
C**   +---------------------------------------------------------+    **
C**   | Find XMIN=A(KK,KJ)=min.A(KK,1:JJ-1) -> XMIN<0 & LJ(KJ)>0 |    **
C**   | Case 1:  P(LJ(KJ)) = +1   (P(LJ(KJ))=SIGN(1,D(LJ(KJ))))  |    **
C**   | Case 2:  P(LJ(KJ)) = -1                                  |    **
C**   | If all A(KK,1:JJ-1) >= 0 : End of phase 1/2 -> go to 700 |    **
C**   +---------------------------------------------------------+    **
  300 IF(KKK.EQ.0)CALL CHKOUT(A,LI,LJ,D,LI,LJ,II,JJ,NX,NIA,NJA,IC,0,1,1)
      XMIN=-EPS
      KJ=0
      DO 310 J=1,JJ-1
      IF(A(KK,J).GE.XMIN.OR.LJ(J).LE.0) GO TO 310
        XMIN=A(KK,J)
        KJ=J
  310 CONTINUE
      IF(KJ.LE.0) GO TO 700
C12   +---------------------------------------------------------+    **
C**   | Find XMIN=min.XI      (for KI=1 to II-1)                 |    **
C**   | Case A: A(KI,KJ)>0 -> XI=A(KI,JJ)/A(KI,KJ)               |    **
C**   | Case B: A(KI,KJ)<0 -> XI=(A(KI,JJ)-D(LI(I)))/A(KI,KJ)    |    **
C**   | Case C: (KI=0)     -> XI=D(LJ(KJ))                       |    **
C**   +---------------------------------------------------------+    **
      LJK=LJ(KJ)
      XMIN=ABS(D(LJK))
      KI=0
      DO 450 I=1,II-1
      IF(ABS(A(I,KJ)).LE.EPS) GO TO 450
        IF(A(I,KJ).GT.0.0) THEN
          XI=A(I,JJ)/A(I,KJ)
        ELSE
C**       +---------------------------------------------------+    **
C**       | Case B:  for slack var LI(I)>0 and bounded var only  |    **
C**       +---------------------------------------------------+    **
          LII=LI(I)
          IF(LII.LE.0) GO TO 450
          IF(D(LII).GE.BIG) GO TO 450
```

```
          XI=(A(I,JJ)-D(LII))/A(I,KJ)
        ENDIF
C**     ------------------------------------------------------------    **
        IF(XI.GT.XMIN) GO TO 450
        IF(XI.EQ.XMIN.AND.KI.NE.0) THEN
          IF(LI(I).GT.LI(KI)) GO TO 450
        ENDIF
          XMIN=XI
          KI=I
  450 CONTINUE
C13     +-------------------------------------------------------+    **
C**     | If XMIN >= BIG : Un-bounded for new basic var (go to 740)|    **
C**     +-------------------------------------------------------+    **
        IF(XMIN.GE.BIG) GO TO 740
C14     +-------------------------------------------------------+    **
C**     | Cases 1A,2A,1B,2B (KI>0)                              |    **
C**     +-------------------------------------------------------+    **
        IF(KI.GT.0) THEN
          LKI=LI(KI)
          LIK=IABS(LKI)
          BKK=ISIGN(1,LKI)
C**       +-----------------------------------------------------+    **
C**       | Sweep LI(KI),LJ(KJ) and Set LJ(KJ) to <0 for artif. var|    **
C**       +-----------------------------------------------------+    **
          LI(KI)=LJK
          LJ(KJ)=LIK
          IF(LIK.GT.NX) LJ(KJ)=-LIK
C15       +-----------------------------------------------------+    **
C**       | Cases 1B,2B : Change basic var to non-bas var X -> D-X'|    **
C**       | (A(KI,KJ)<0)  X(LIK)'= D(LIK)-X(LIK) = 0, P(LIK) = -1  |    **
C**       +-----------------------------------------------------+    **
          IF(A(KI,KJ).LT.0.0) THEN
            A(KI,JJ)=A(KI,JJ)-D(LIK)
            D(LIK)=-D(LIK)
            BKK=-1.0
          ENDIF
        ENDIF
C16     +-------------------------------------------------------+    **
C**     | Case 2A,2B (KI>0) : X(LJK) < D(LJK), P(LJK) = -1 --> +1  |    **
C**     | Case 1C    (KI=0) : X(LJK)'= 0,      P(LJK) = +1 --> -1  |    **
C**     | Case 2C    (KI=0) : X(LJK) = 0,      P(LJK) = -1 --> +1  |    **
C**     +-------------------------------------------------------+    **
        IF(D(LJK).LT.0.0.OR.KI.EQ.0) THEN
          DJ=ABS(D(LJK))
          DO 540 I=1,KK
            A(I,JJ)=A(I,JJ)-A(I,KJ)*DJ
            A(I,KJ)=-A(I,KJ)
  540     CONTINUE
          D(LJK)=-D(LJK)
        ENDIF
C17     +-------------------------------------------------------+    **
C**     | Cases 1A,2A,1B,2B (KI>0) : Perform Gauss elimination     |    **
C**     +-------------------------------------------------------+    **
        IF(KI.GT.0) CALL GAUSS(A,II,JJ,KI,KJ,BKK,KK,LKI,NIA,NJA)
C18     +-------------------------------------------------------+    **
C**     | End of simplex loop : go to Begin of loop (300)          |    **
C**     +-------------------------------------------------------+    **
        KKK=KKK+1
        CALL CHKOUT(A,LI,LJ,D,MI,MJ,II,JJ,NX,NIA,NJA,IC,KKK,KI,KJ)
        GO TO 300
```

```
C19   +-------------------------------------------------------+    **
C**   | Re-perform phase 1 simplex 5-times maximum            |    **
C**   +-------------------------------------------------------+    **
  700 IF(KK.EQ.II) GO TO 710
      IF(KKKK.GE.5.OR.KKKO.EQ.KKK) GO TO 750
      KKKK=KKKK+1
      KKKO=KKK
      IF(IS.NE.0) THEN
      CALL RECOMP(A,LI,LJ,D,II,JJ,NX,NIA,NJA,T,MI,MJ,IS)
      CALL CHKOUT(A,LI,LJ,D,MI,MJ,II,JJ,NX,NIA,NJA,IC,0,1,1)
      ENDIF
      GO TO 200
C20   +-------------------------------------------------------+    **
C**   | Feasible solution in X(*) (also for Unbounded/Infeasible)|  **
C**   +-------------------------------------------------------+    **
  710 DO 720 J=1,NX
      X(J)=MAX(0.0D0,-D(J))
  720 CONTINUE
      DO 730 I=1,II-1
      IF(LI(I).GT.0) X(LI(I))=A(I,JJ)
  730 CONTINUE
      DO 735 J=1,NX
      X(J)=MIN(X(J),ABS(D(J)))
  735 CONTINUE
      RETURN
C21   +-------------------------------------------------------+    **
C**   | Unbounded solution                                    |    **
C**   +-------------------------------------------------------+    **
  740 ICASE=1
      GO TO 710
C22   +-------------------------------------------------------+    **
C**   | Infeasible solution                                   |    **
C**   +-------------------------------------------------------+    **
  750 ICASE=-2
      GO TO 710
      END
*************************************************************************
      SUBROUTINE GAUSS(A,II,JJ,KI,KJ,BKK,KK,LKI,NIA,NJA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NIA,NJA)
C**   ============================================================    **
C**   Perform Gauss elimination pivot at A(KI,KJ)                     **
C**   A(*,KJ) will be used by new non-basic var   (BKK = -1 or +1)    **
C**   LKI=LI(KI) < 0 : New non-bas var is art var (has 1*M in obj)    **
C**   ============================================================    **
      AKK=A(KI,KJ)
      A(KI,KJ)=BKK
      DO 630 J=1,JJ
      A(KI,J)=A(KI,J)/AKK
  630 CONTINUE
      DO 640 I=1,KK
      IF(I.EQ.KI) GO TO 640
        AIK=A(I,KJ)
        A(I,KJ)=0.0
        IF(I.EQ.II+1.AND.LKI.LT.0) A(I,KJ)=1.0
        DO 635 J=1,JJ
        A(I,J)=A(I,J)-AIK*A(KI,J)
  635   CONTINUE
  640 CONTINUE
```

```
      RETURN
      END
************************************************************************
      SUBROUTINE RECOMP(A,LI,LJ,D,II,JJ,NX,NIA,NJA,T,MI,MJ,IS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NIA,NJA),LI(NIA),LJ(NJA),D(NX)
      DIMENSION T(NIA,NJA),MI(NIA),MJ(NJA)
      DATA EPS/1.0D-6/
C**   ============================================================    **
C**   Perform Gauss elimination from the original matrix              **
C**   For phase 1 only : To reduce roundoff error                     **
C**   ============================================================    **
  800 IF(IS.GT.0) THEN
        REWIND IS
        READ (IS) ((A(I,J),I=1,II),J=1,JJ)
      ELSE IF(IS.LT.0) THEN
        DO 820 J=1,JJ
        DO 815 I=1,II
        A(I,J)=T(I,J)
  815   CONTINUE
  820   CONTINUE
      ENDIF
C**   +----------------------------------------------------------+    **
C**   | LI(I) = Basic variables;   LJ(J) = Non-basic variables   |    **
C**   +----------------------------------------------------------+    **
      DO 830 I=1,II-1
      LI(I)=IABS(LI(I))
  830 CONTINUE
      DO 840 J=1,JJ-1
      LJ(J)=IABS(LJ(J))
  840 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Set MI(I) = -MI(I) < 0  if MI(I) not in LI(1:II-1)       |    **
C**   +----------------------------------------------------------+    **
      NI=0
      DO 850 I=1,II-1
        DO 845 KI=1,II-1
        IF(MI(I).EQ.LI(KI)) GO TO 850
  845   CONTINUE
        MI(I)=-MI(I)
        NI=NI+1
  850 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Set MJ(J) = -MJ(J) < 0  if MJ(J) not in LJ(1:JJ-1)       |    **
C**   +----------------------------------------------------------+    **
      NJ=0
      DO 860 J=1,JJ-1
        DO 855 KJ=1,JJ-1
        IF(MJ(J).EQ.LJ(KJ)) GO TO 860
  855   CONTINUE
        MJ(J)=-MJ(J)
        NJ=NJ+1
  860 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Move signed MI(I) & MJ(J) to LI(I) & LJ(J)               |    **
C**   | Reset MI(I) & MJ(J) to original positive values          |    **
C**   +----------------------------------------------------------+    **
      DO 865 I=1,II-1
      LI(I)=MI(I)
```

```
  865 MI(I)=IABS(MI(I))
      DO 870 J=1,JJ-1
      LJ(J)=MJ(J)
  870 MJ(J)=IABS(MJ(J))
C**   +--------------------------------------------------------+    **
C**   | Find pivot element A(KI,KJ) : LI(KI) < 0 and LJ(KJ) < 0  |    **
C**   +--------------------------------------------------------+    **
      IF(NI.NE.NJ) STOP 2222
      IF(NI.EQ.0) GO TO 960
      DO 950 K=1,NI
        AKK=0.0
        DO 920 J=1,JJ-1
        IF(LJ(J).GE.0) GO TO 920
          DO 910 I=1,II-1
          IF(LI(I).GE.0) GO TO 910
          IF(AKK.GE.ABS(A(I,J))) GO TO 910
            AKK=ABS(A(I,J))
            KI=I
            KJ=J
  910     CONTINUE
  920   CONTINUE
        IF(AKK.LE.EPS) GO TO 960
C**     +--------------------------------------------------------+    **
C**     | Elimination by pivot row : Sweep -LI(KI) <--> -LJ(KJ)  |    **
C**     +--------------------------------------------------------+    **
        LIK=-LI(KI)
        LI(KI)=-LJ(KJ)
        LJ(KJ)=LIK
C**     ----------------------------------------------------------    **
C**     CALL GAUSS(A,II,JJ,KI,KJ,1.0D0,II,1,NIA,NJA)   ! Also OK      **
C**     ----------------------------------------------------------    **
        AKK=A(KI,KJ)
        A(KI,KJ)=1.0
        DO 930 J=1,JJ
        A(KI,J)=A(KI,J)/AKK
  930   CONTINUE
        DO 940 I=1,II
        IF(I.EQ.KI) GO TO 940
          AIK=A(I,KJ)
          A(I,KJ)=0.0
          DO 935 J=1,JJ
          A(I,J)=A(I,J)-AIK*A(KI,J)
  935     CONTINUE
  940   CONTINUE
  950 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Reform simplex tableau                                 |    **
C**   | (1) Reset P(LI(I)) = +1                                |    **
C**   | (2) Change var X --> D-X'  if P(LJ(J)) = -1            |    **
C**   +--------------------------------------------------------+    **
  960 DO 970 I=1,II-1
      LII=LI(I)
      IF(LII.GT.NX) GO TO 970
        D(LII)=ABS(D(LII))
  970 CONTINUE
      DO 990 J=1,JJ-1
      LJJ=LJ(J)
      IF(LJJ.GT.NX) GO TO 990
      IF(D(LJJ).GE.0.0) GO TO 990
        DJ=-D(LJJ)
```

```
        DO 980 I=1,II
        A(I,JJ)=A(I,JJ)-A(I,J)*DJ
        A(I,J)=-A(I,J)
  980   CONTINUE
  990 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE CHKOUT(A,LI,LJ,D,MI,MJ,II,JJ,NX,NIA,NJA,IC,KKK,KI,KJ)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NIA,NJA),LI(NIA),LJ(NJA),D(NX),MI(NIA),MJ(NJA)
C**   ========================================================    **
C**   Print out simplex tableau
C**   ========================================================    **
      IF(IC.LE.0) RETURN
      WRITE(IC,'(3H  +,8A)')('---------',J=1,JJ-1),'-+-----------+-----'
      DO 40 I=1,II-1
      WRITE(IC,'(3H  |,F7.4,$)') A(I,1)
      DO 30 J=2,JJ-1
   30 WRITE(IC,'(F9.4,$)') A(I,J)
   40 WRITE(IC,'(4H   |,F8.4,4H   |,I4)') A(I,JJ),LI(I)
      WRITE(IC,'(3H  +,8A)')('---------',J=1,JJ-1),'-+-----------+-  ^ '
      WRITE(IC,'(3H  |,F7.4,$)') A(II,1)
      DO 50 J=2,JJ-1
   50 WRITE(IC,'(F9.4,$)') A(II,J)
      WRITE(IC,'(4H   |,F8.4,8H   |  LI)') A(II,JJ)
      WRITE(IC,'(3H  |,F7.4,$)') A(II+1,1)
      DO 60 J=2,JJ-1
   60 WRITE(IC,'(F9.4,$)') A(II+1,J)
      WRITE(IC,'(4H   |,F8.4,8H   *  M )') A(II+1,JJ)
      WRITE(IC,'(3H  +,8A)')('---------',J=1,JJ-1),'-+-----------+-----'
      WRITE(IC,'(3H  |,I6,$)') LJ(1)
      DO 70 J=2,JJ-1
   70 WRITE(IC,'(I9,$)') LJ(J)
      WRITE(IC,'(17H    <--   LJ    |,//)')
      DO 80 J=1,JJ-1
   80 WRITE(IC,'(F8.4,$)') D(J)
      WRITE(IC,'(7H <-- DJ,/)')
      RETURN
      END
*************************************************************************
```

習題

1. 試以單一法求下列線性規劃問題之解。

$$
\begin{aligned}
\text{極小化：}\quad & z = -8x_1 - 4x_2 \\
\text{受制於：}\quad & 4x_1 + 3x_2 \;\le\; 12 \\
& 2x_1 - \;\; x_2 \;=\; 4 \\
& 3x_1 + 5x_2 \;\le\; 15 \\
& x_1 \ge 0\,,\;\; x_2 \ge 0
\end{aligned}
$$

2. 寫出上列基本問題對應之對偶問題，並以單一法求其解。

3. 試證式(14.22)之線性規劃問題可化為下列四個問題：(注意其中有二個問題沒有目標函數)

$$\begin{array}{ccc|c} A & 0 & B & \\ 0 & -T & C & \\ \hline C & B & 0 & \end{array} \qquad \begin{array}{ccc|c} -T & 0 & -C & 0 \\ 0 & A & -B & 0 \\ \hline B & C & 0 & 0 \end{array}$$

$$\begin{array}{ccc|c} A & 0 & B & \\ 0 & -T & C & \\ C & B & 0 & \\ \hline 0 & 0 & 0 & \end{array} \qquad \begin{array}{ccc|c} A & 0 & B & 0 \\ 0 & -T & C & 0 \\ C & B & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

參考文獻

1. A. Charnes, W. W. Cooper, and A. Henderson, *An Introduction to Linear Programming,* Wiley, New York, 1953.

2. G. R. Walsh, *An Introduction to Linear Programming,* Wiley, New York, 1985.

3. Bruce A. Murtagh, *Advanced Linear Programming: Computation and Practice,* McGraw-Hill, New York, 1981.

第十五章

整數線性規劃問題

15.1 前言

一般線性規劃問題所求得最佳解之變數值為連續性(continuous)之實數值，但若限制某些變數值須為整數(integer)或離散數(discrete，即為一些指定的個值)，即稱為混合整數或離散數線性規劃問題(Mixed integer/discrete linear programming)。此處混合所指者為實數與整數混合，或實數與離散數混合。特別指出混合係因純整數線性規劃問題可以用更有效的方法，即轉換為零壹規劃問題求解。

工程應用上大部分規劃問題之變數值常須為整數或離散數，例如梁或柱之尺寸，型鋼之面積與二次矩等。一般做法常將實數規劃問題之最佳解之變數(實數)值改為最接近之整數或離散數，此種近似方式在變數值很大時常可得到滿意之結果。但如變數值不大時則不一定會有合理之結果。

整數線性規劃問題常用之解法有二：(1)Land與Doig之分支與限值法(branch and bound method)；(2)Gomory之切面法(cutting plane method)。其中之分支與限值法亦適用於離散數，因此本章所提供程式採用此法，故可分析同時含有實數、整數與離散數之變數。本章對切面法亦僅做簡單介紹。

分支與限值法或切面法基本上都是做很多(實數)線性規劃問題。切面法為逐次增加新的束制式以促使原非整數之變數值變為整數值。而分支與限值法則對每一非整數之變數值，以$x_3 = 3.76$為例，分成二個規劃問題：(1)增加$x_3 \leq 3$之限制；(2)增加$x_3 \geq 4$之限制。這就是所謂的分支。但經此分支所得之新問題為數頗多，如果全都分析將頗費時。但事實上

有些分支問題在少部分整數值之限制時已無合理解，因此沒有必要再繼續分支以增加對其他非整數變數之限制。另外當某分支問題已求得整數變數均為整數值且為滿足所有束制之合理解時(簡稱整數合理解)，則其他分支之非整數合理解(滿足束制式但整數變數非全為整數值)之目標函數值已比最佳之整數合理解者為差時，也沒有必要再對此分支繼續再分支，因為增加束制式之分支問題，其目標函數值只會更差。這就是所謂的限值。

一般線性規劃問題之求解程式如果沒有考慮變數有上限之限制，則利用分支與限值法時，對變數之上限或下限限制都要以增加束制式之方式為之，因此各分支問題之束制式數目即不相同，程式也會較為複雜。本章利用之線性規劃程式已考慮變數有上限限制，因此利用分支與限值法時，僅需更改變數之上限限值，束制式數目在各分支問題均相同，因此程式也相對地較為單純。

15.2 整數線性規劃問題

考慮下列之整數線性規劃問題：

$$\text{最小化} \quad z = \sum_{j=1}^{N_v} c_j x_j \tag{15.1}$$

$$\text{受制於} \quad \sum_{j=1}^{N_v} a_{ij} x_j \le b_i\ , \qquad i = 1, \ldots, N_c \tag{15.2}$$

$$x_j = 0, 1, 2, \ldots\ , \qquad j = 1, \ldots, N_i \tag{15.3}$$

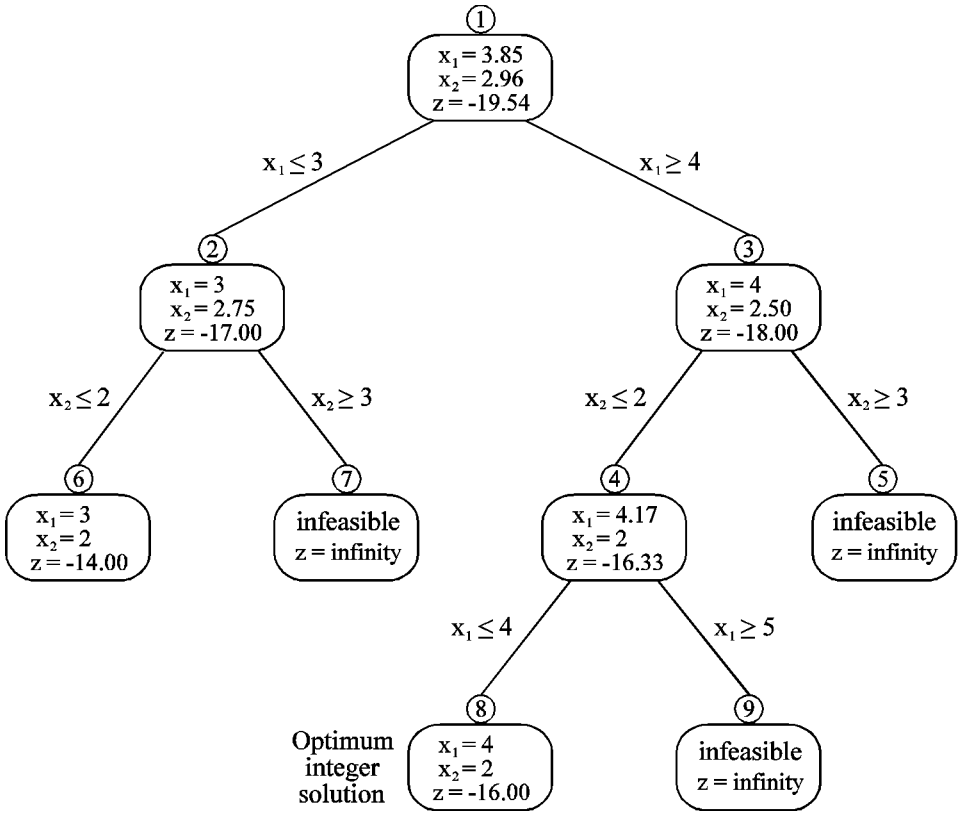
$$x_j \ge 0\ , \qquad j = N_i + 1, \ldots, N_v \tag{15.4}$$

其中N_c為束制式之數目；N_v為變數之數目；N_i為整數變數之數目，整數變數可不必如上式全部集中在實數變數之前。式(15.2)中束制亦可為$\ge$或$=$之型式，式(15.3)與式(15.4)中之變數除整數或≥ 0之正值限制外，亦可加上限限制：即$d_j \ge x_j \ge 0$。若式(15.3)之部分變數值須限為$x_j = 0, v_{1j}, v_{2j}, \ldots, v_{mj}, d_j$，則該問題成為混合離散數線性規劃問題。本章所提供程式可包含下列各種變數：(1)實數變數；(2)限值為0或1之整數變數；(3)整數變數；(4)離散變數。且變數不必按次序集中排列。由於其適用範圍很廣，沒有必要做詳細之區分，故除非特別指明，本章將統稱這些問題為整數線性規劃問題。

15.3 分支與限值法

以下將利用下列之簡單例題說明分支與限值法之有關細節：由於分支問題頗多，將每一問題按分析先後依次從1開始編號，並按分支關係繪成如圖一之樹枝(tree)圖，每一問題在樹枝圖中為一個節點(node)，將某問題之節點(稱母節點)與其分支問題之節點(稱子節點)用直線相連，即得樹枝圖。在分支與限值法中每一母節點均有二個子節點，分別稱為左節點與右節點，亦有稱為(兒)子節點與女(兒)節點者。本章將以左右節點稱之。並規定$x_i \le k$者在左節點；$x_i \ge k+1$者在右節點。

$$
\begin{aligned}
\text{最小化} \quad z &= -2x_1 - 4x_2 \\
\text{受制於} \quad & 6x_1 + 2x_2 \le 29 \\
& -2x_1 + 8x_2 \le 16 \\
& x_1, x_2 = 0, 1, 2, \ldots
\end{aligned}
$$



圖一 分支與限值法

(1) 上列問題 (節點 1) 以線性規劃問題求解可得：$x_1 = 3.85$ ，$x_2 = 2.96$ ，$z = -19.54$ 。因其變數值非整數，接著分析其分支問題：

(2) 左節點 (節點 2) 增加 $x_1 \leq 3$ 之限制；分析結果為：$x_1 = 3$ ，$x_2 = 2.75$ ，$z = -17.00$ ，因非整數解，其子節點須對 x_2 做限制。

(3) 右節點 (節點 3) 增加 $x_1 \geq 4$ 之限制，分析結果為：$x_1 = 4$ ，$x_2 = 2.50$ ，$z = -18.00$ ，亦非整數解，其子節點亦對 x_2 做限制。

現有二個問題均須做其分支問題，應選何者先做其分支問題，基本上並無絕對之最佳準則，以下為常用之原則，亦為本章所提供之程式所採用者：選取目標函數值最小之節點，因其較有可能由其子節點求得目標函數最小之最佳解。注意若某節點為不合理解，則其子節點亦不可能有合理解，因子節點之限制條件比母節點多而更不容易滿足，故不必做該節點之分支問題。

按上述原則選節點 $(2, 3)$ 之目標函數 $(-17.00, -18.00)$ 最小之節點 3 為母節點， 續做其左右子節點：

(4) 左節點 (節點 4) 增加 $x_2 \leq 2$ 之限制，連同母節點之限制 $x_1 \geq 4$ ，分析結果為：$x_1 = 4.17$ ，$x_2 = 2$ ，$z = -16.33$ ，仍非整數解，其子節點須對 x_1 做限制。

(5) 右節點 (節點 5) 增加 $x_2 \geq 3$ 之限制，連同母節點之限制 $x_1 \geq 4$ ，分析結果為：不合理解。

注意節點 4 之變數 x_1 ，雖然在其母節點為整數值 4 ，但在該子節點又變成非整數之 4.17 。這是頗正常而常有之情形。由此可見，雖然母節點已得 $x_1 = 4$ 之整數解，但不能直接令 $x_1 = 4$ ，而將限制條件 $x_1 \geq 4$ 去掉。

現除節點 5 為不合理解外，尚有節點 $(2, 4)$ 可選為母節點，按其目標函數值 $(-17.00, -16.33)$ 之最小者選節點 2 ，續做其左右子節點：

(6) 左節點 (節點 6) 增加 $x_2 \leq 2$ 之限制，連同母節點之限制 $x_1 \leq 3$ ，分析結果為：$x_1 = 3$ ，$x_2 = 2$ ，$z = -14.00$ ，為整數合理解。

(7) 右節點 (節點 7) 增加 $x_2 \geq 3$ 之限制，連同母節點之限制 $x_1 \leq 3$ ，分析結果為：不合理解。

對於節點 6 之合理解，應與已求得之最佳解比較並保留最佳者之節點號碼為 $n_{opt} = 6$ ，與目標函數值為 $z_{opt} = -14.00$ ，以為後續又有整數合理解時比較之用。

至目前為止，僅剩節點 4 可選為母節點，因為其他節點屬於下列四種情形之一而均不必做其子節點之分支問題：(1) 節點 $1, 2, 3$ 已做過其子節點

；(2) 節點 $5,7$ 為不合理解，其子節點亦不可能有合理解；(3) 節點 6 為整數合理解，其子節點雖亦可能為合理解，但其目標函數值不會比母節點更佳，故亦不必再做其子節點之分支問題。(4) 另外一種不在本問題出現之情形為：該節點之目標函數值不比已求得之最佳整數合理解為佳。因為其子節點之目標函數不會比母節點更佳。

接著做節點 4 之左右子節點：

(8) 左節點 (節點 8) 增加 $x_1 \leq 4$ 之限制，連同母節點之限制 $x_2 \leq 2$ ，及祖母節點之限制 $x_1 \geq 4$ ，分析結果為：$x_1 = 4$ ，$x_2 = 2$ ，$z = -16.00$ ，為整數合理解。

(9) 右節點 (節點 9) 增加 $x_1 \geq 5$ 之限制，連同母節點之限制 $x_2 \leq 2$ ，及祖母節點之限制 $x_1 \geq 4$ ，分析結果為：不合理解。

注意對變數新增之限制條件，必須包含其上代所有祖先之新增限制，亦即如有重複或重疊之限制應受最嚴之限制，因此某變數之上限為對該變數新增之所有上限之最小值，而其下限則為對該變數新增之所有下限之最大值。本章採用之線性規劃程式考慮變數均有上限限制， 對變數有上下限值者，如 $x_j^u \geq x_j \geq x_j^l$ ，可用新變數 x_j' 取代 x_j ，以 $x_j = x_j' + x_j^l$ 代入原問題之各式，即可得對應新變數之各式。令 $d_j = x_j^u - x_j^l$ ，則新變數之上下限即可改為 $d_j \geq x_j' \geq 0$ 之上限限制及正值限制。

注意節點 8 為整數合理解，其目標函數值 $z = -16.00$ ，比已求得之最佳解 $z_{opt} = -14.00$ 更佳，故最佳解之節點改為 $n_{opt} = 8$ ，最佳目標函數值改為 $z_{opt} = -16.00$ 。現因已無母節點須做其分支問題，故目前所得之最佳解即為原整數規劃問題之最佳解：$x_1 = 4$ ，$x_2 = 2$ ，$z_{opt} = -16.00$ 。

對於離散數之變數，若母節點之某變數值 x_j 介於 v_k 與 v_{k+1} 之間，而不等於這些值時，相當於整數變數值非為整數，須對該變數做二種分支問題：其左節點之新增限制為 $x_j \leq v_k$ ；右節點之新增限制為 $x_j \geq v_{k+1}$ 。其餘做法與整數者相同。

15.4 節點之資料表示法

在程式 DLP 中節點 I 之目標函數值存於 $ZNOD(I)$ 。其母節點號碼存於 $MVX(1,I)$ 。其子節點擬增加上或下限束制之變數號碼存於 $MVX(2,I)$ ，該值如為 0 表示本 (母) 節點為整數解，即整數變數全為整數。其左 (子) 節點之上限值存於 $MVX(3,I)$ ，對於離散數則儲存其離散值之指標，例如

$x_j \le v_k$ 時，$MVX(3,I)=k$。

另外當某節點為不合理解時，不必做其子節點之分支問題，故可不用 $MVX(2,I)$ 儲存變數號碼，因此令 $MVX(2,I)=-1$（由程式 $INTCHK$ 執行），以標示不必做其子節點。同樣在某節點已做過其左右節點後，可將 $MVX(2,I)$ 變號改為負值（由程式 $SAVNOD$ 執行），以標示不必做其子節點。以此方式設定後，找下一個母節點時，只須從 $MVX(2,I)\ge 0$ 之節點中找 $ZNOD(I)$ 為最小之節點即可（詳副程式 $NXTNOD$）。下表為上節例題之節點資料。

Node	Mother	JV	JX	Solution		
I	$MVX(1,I)$	$MVX(2,I)$	$MVX(3,I)$	$ZNOD(I)$	x_1	x_2
1	0	−1	3	−19.54	3.85	2.96
2	1	−2	2	−17.00	3	2.75
3	1	−2	2	−18.00	4	2.50
4	3	−1	4	−16.33	4.17	2
5	3	−1	0	−	−	−
6	2	0	0	−14.00	3	2
7	2	−1	0	−	−	−
8	4	0	0	−16.00	4	2
9	4	−1	0	−	−	−

15.5 變數類型與離散數之儲存方式

本節首先說明程式中以 $ITYP(1,J)$ 所標示之變數類型：實數以0標示；0或1整數以1標示；整數以2標示；離散數以 $-m$ 標示。其中 m 值表示該離散數有 m 個中間指定值，不包含最小之0值與最大之上限值 d_j，且須 $m>0$。另外在求得線性規劃之解後，係由副程式 $INTCHK$ 檢查變數值是否為整數或離散數，並將檢查之結果標示於 $ITYP(3,J)$：(1)以 -1 表示：(1a)實數變數；(1b)整數變數值為整數；(1c)離散變數值為離散數。(2)以 ≥ 0 表示：(2a)整數變數值非整數值；(2b)離散變數值非離散數；此時 $ITYP(3,J)$ 等於比變數值小之最大整數，或比變數值小之最大離散值之指標。利用此標示，$INTCHK$ 即可自 $ITYP(3,J)\ge 0$ 之變數中，找出一個變數 JV 與 $JX=ITYP(3,JV)$，以做為分析分支問題之用；如為不合理解時，則令 $JV=-1$。此處所得之 JV 與 JX 即傳給程式 $SAVNOD$，

由其儲存於$MVX(2,NODE)$與$MVX(3,NODE)$。

當$ITYP(1,J)$=3時，變數J為離散數，其離散值$v_1,v_2,\ldots,v_m$須儲存在$VAL(JP)$至$VAL(JP+m-1)$中。而JP則存於$ITYP(2,J)$。注意v_k須由小到大排列且須大於0。例如下列資料所存放者為離散變數x_2之4個中間指定值$(1.2,2.5,4.2,6.3)$與離散變數x_4之3個中間指定值$(0.8,1.8,3.4)$。注意其中$ITYP(1,1)$=0，表示x_1為實數變數；$ITYP(1,3)$=2，表示x_3為整數變數；$ITYP(1,5)$=1，表示x_5為0或1整數變數。這三種變數之$ITYP(2,J)$不必使用。

$VAL(K)$	1.2	2.5	4.2	6.3	0.8	1.8	3.4
K	1	2	3	4	5	6	7

$ITYP(1,J)$	0	−4	2	−3	1
$ITYP(2,J)$		1		5	
J	1	2	3	4	5

15.6 切面法

切面法基本上也是利用一般實數變數之線性規劃法求解，然後根據整數變數之非整數解所對應之束制式，利用下述做法，得出一個新的束制條件，做為原問題之新增束制式，再以(實數)線性規劃問題求解。如仍有非整數解，再以同樣方式繼續增加新的束制式求解，直到整數變數值全為整數為止。現假設某整數變數x_3之解為9.6，在最後之運算式中，其對應之束制式為：

$$0x_1+2x_2+x_3+0x_4+1.7x_5-2.2x_6+2.3x_7=9.6$$

由上式顯然可知$x_2=x_5=x_6=x_7=0$為非基本變數，x_1與x_4可能為基本變數。這些變數是否為基本變數對以下做法並無關係。有關係者為這些變數是否為整數變數。現假設x_7為實數變數，其餘為整數變數。現可將上式改寫成：

$$(2+0)x_2+(1+0)x_3+(1+0.7)x_5+(-3+0.8)x_6+2.3x_7 \;=\; (9+0.6)$$

或

$$0.7x_5 + 0.8x_6 + 2.3x_7 \quad = \quad 0.6 + (9 - 2x_2 - x_3 - x_5 + 3x_6)$$

因x_5、x_6與x_7均為正值變數，上式等號左邊之係數亦均為正值(如x_7之係數為負值，應先將所有係數變號後再處理；如實數變數之係數有正也有負，則該束制式無法用以產生新束制式，應改用其他束制式)，故等號左邊(與右邊)之值一定為正值。因此右邊括號內之值必須≥ -0.6，否則即不可能在x_5、x_6與x_7為正值時使上式成立。若括號內之變數均為整數變數，則括號內之值必為整數，則其不僅須≥ -0.6，且還須≥ 0。因此等式左邊之值即不僅須≥ 0，且還須≥ 0.6。注意實數變數均應留在等式左邊，如x_7。因此可得下列之(較嚴密)束制式：

$$0.7x_5 + 0.8x_6 + 2.3x_7 \quad \geq \quad 0.6 \tag{15.5}$$

上述之束制式即稱為切面。因為其限制較嚴(由≥ 0改為≥ 0.6)，故可將部分(無整數限制下之)合理區域切除。因此當增加的束制式夠多時，可以促使實數線性規劃問題求得整數解。

仍以第15.3節之例題(如下之表格)用切面法求解如下：

```
+-----------------------------------+----------+-----
|-2.0000   8.0000   1.0000    .0000  | 16.0000  |  -5
| 6.0000   2.0000    .0000   1.0000  | 29.0000  |  -6
+-----------------------------------+----------+-  ^
|-2.0000  -4.0000    .0000    .0000  |   .0000  |  LI
|-4.0000 -10.0000  -1.0000  -1.0000  |-45.0000  *  M
+-----------------------------------+----------+-----
|     1        2        3        4    <--   LJ    |
```

上表經求解可得下列之結果：

```
+-----------------------------------+----------+-----
| -.0385   -.1154    .1154    .0385  |  2.9615  |   2
| -.1538    .0385   -.0385    .1538  |  3.8462  |   1
+-----------------------------------+----------+-  ^
| -.4615   -.3846    .3846    .4615  | 19.5385  |  LI
|  .0000    .0000    .0000    .0000  |   .0000  *  M
+-----------------------------------+----------+-----
|    -6       -5        3        4    <--   LJ    |
```

由上表可知$x_1 = 3.8462$，$x_2 = 2.9615$，$x_3 = x_4 = 0$。x_1與x_2均非整數，選其中小數部分較大之x_2之第1束制式為：

$$x_2 + 0.1154x_3 + 0.0385x_4 \quad = \quad 2.9615 \tag{15.6}$$

注意上式並未包含變數x_5與x_6，因這二個變數均為虛設變數，其值必須為0。按前述方法可得下列之新增束制式：

$$0.1155x_3 + 0.0386x_4 \quad \geq \quad 0.9614 \tag{15.7}$$

上式中左邊之係數略增，右邊之常數則略減，以免束制過分嚴格而產生不合理束制。經加入上式之束制後即得下列之新問題：

```
+-----------------------------------+-----------+-----
|-2.0000   8.0000   1.0000    .0000  | 16.0000   |  -6
| 6.0000   2.0000    .0000   1.0000  | 29.0000   |  -7
|  .0000    .0000    .1155    .0386  |   .9614   |  -5
+-----------------------------------+-----------+-  ^
|-2.0000  -4.0000    .0000    .0000  |   .0000   |  LI
|-4.0000 -10.0000  -1.1155  -1.0386  |-45.9614   *  M
+-----------------------------------+-----------+-----
|     1        2        3        4    <--   LJ    |
```

再經求解可得下列之結果：

```
+-----------------------------------+-----------+-----
| -.0385   -.1154    .9990   -.0001  |  2.0011   |   2
| -.1538    .0385   -.3330    .1667  |  4.1663   |   1
|  .0000    .0000  -8.6580    .3342  |  8.3238   |   3
+-----------------------------------+-----------+-  ^
| -.4615   -.3846   3.3300    .3330  | 16.3370   |  LI
|  .0000    .0000    .0000    .0000  |   .0000   *  M
+-----------------------------------+-----------+-----
|    -7       -6        5        4    <--   LJ    |
```

由上表可知$x_1 = 4.1663$，$x_2 = 2.0011$，$x_3 = 8.3238$，$x_4 = x_5 = 0$。x_1與x_3均非整數(x_2可近似視為整數)，可選用對應之第2或第3束制式，以產生新束制式。現因x_5為實數變數，且其係數在該二束制式中均為負值，因此該二束制式之係數須予變號，而成為式(15.8)與式(15.9)，再以前述方式處理後可得束制式(15.10)與式(15.11)。式(15.8)之常數項(0.8336)較(0.6761)大，故選為新增束制式，因此得後列之新問題。

$$-x_1 - 0.1667x_4 + 0.3330x_5 \quad = \quad -4.1663 \tag{15.8}$$

$$-x_3 - 0.3342x_4 + 8.6580x_5 \quad = \quad -8.3238 \tag{15.9}$$

$$0.8334x_4 + 0.3331x_5 \quad \geq \quad .8336 \tag{15.10}$$

$$0.6659x_4 + 8.6581x_5 \quad \geq \quad .6761 \tag{15.11}$$

```
+-----------------------------------------+----------+-----
|-2.0000   8.0000   1.0000    .0000    .0000   | 16.0000  |  -7
| 6.0000   2.0000    .0000   1.0000    .0000   | 29.0000  |  -8
|  .0000    .0000    .1155    .0386  -1.0000   |   .9614  |  -9
|  .0000    .0000    .0000    .8334    .3331   |   .8336  |  -6
+-----------------------------------------+----------+-  ^
|-2.0000  -4.0000    .0000    .0000    .0000   |   .0000  |  LI
|-4.0000 -10.0000  -1.1155  -1.0720    .6669   |-46.7950  *  M
+-----------------------------------------+----------+-----
|     1        2        3        4        5    <--   LJ    |
```

再經求解可得下列之結果：

```
+-----------------------------------------+----------+-----
| -.0385   -.1154    .9990   -.0001    .9990   |  2.0012  |   2
| -.1538    .0385   -.3330    .2000   -.3996   |  3.9996  |   1
|  .0000    .0000  -8.6580    .4010  -8.7916   |  7.9895  |   3
|  .0000    .0000    .0000  -1.1999    .3997   |  1.0002  |   4
+-----------------------------------------+----------+-  ^
| -.4615   -.3846   3.3300    .3996   3.1969   | 16.0039  |  LI
|  .0000    .0000    .0000    .0000    .0000   |   .0000  *  M
+-----------------------------------------+----------+-----
|    -8       -7       -9        6        5    <--   LJ    |
```

由上表可知$x_1 = 3.9996$，$x_2 = 2.0012$，$x_3 = 7.9895$，$x_4 = 1.0002$，$x_5 = x_6 = 0$。x_1與x_2均可近似視為整數，因此即得最佳解之目標函數值為-16.0039。用較多位小數值計算可以得較準之結果。應用上亦可在變數值與整數相差不多而在容許範圍之內時停止計算以節省時間。

15.7 p個條件至少滿足q個之束制式

假設式(15.12)之p個條件至少要有q個條件須滿足時，可對每一條件各增加一限值為0或1之整數變數y_i，將式(15.12)改以式(15.13)取代，再增加一束制條件式(15.14)，即可達到此目的。式(15.13)中之係數$\overline{d}_i$為g_i之可能最大值或更大之值，實際應用應不難決定，例如選$\overline{d}_i = 100$，或$\overline{d}_i = 1000$等。注意當$y_i = 1$時式(15.13)等於式(15.12)之束制；當$y_i = 0$時式(15.13)一定會成立，即對應之式(15.12)不存在而無作用。

$$g_i \leq 0\,, \qquad i = 1, 2, \ldots, p \tag{15.12}$$

$$g_i - \overline{d}_i\,(1 - y_i) \leq 0\,, \qquad i = 1, 2, \ldots, p \tag{15.13}$$

$$\sum_{i=1}^{p} y_i = q \tag{15.14}$$

上列之最簡單情形為：p=2 ，q=1 ，即$g_1 \le 0$或$g_2 \le 0$二者之一須滿足。此特例可直接用式(15.14)之條件，即$y_1 + y_2 = 1$ ，將y_2以$1 - y_1$取代，即可只增加一個限值為0或1之整數變數y_1 ，而將式(15.12)之二條件改為：

$$g_1 - \overline{d}_1 (1 - y_1) \le 0 \tag{15.15}$$

$$g_2 - \overline{d}_2 y_1 \le 0 \tag{15.16}$$

但注意：當$p > 2$時，以p=3為例，y_3不能以$q - y_1 - y_2$取代。因y_1與y_2為0或1時，$q - y_1 - y_2$不一定為0或1 ，而可能有其他值，故不能取代限值為0或1之y_3 。

15.8 目標函數在某條件時加一固定值

本節再介紹一種可以用0或1整數變數處理之情形：當某變數x_i大於s時，最小化目標函數即須增加一固定值$\bar{c}_i$ ，但當$x_i \le s$時，目標函數即不用增加此固定值(類似目標函數為變數x_i之步階函數 (step function)乘係數$\bar{c}_i$)。其處理方式為：增加一限值為0或1之整數變數y ，將目標函數增加一項為$\bar{c}_i y$ ，並增加下列之束制式(式中之係數$\overline{d}_i$與上節類似，為$(x_i - s)$之可能最大值或更大之值)：

$$(x_i - s) - \overline{d}_i y \le 0 \tag{15.17}$$

上述處理方法之正確性說明如下：當$x_i \le s$時，上式不管$y = 0$或$y = 1$均會成立，但為使目標函數為最小，y必須等於0 ，故$\bar{c}_i$即不會加到目標函數值。反之當$x_i > s$時，上列束制條件必須$y = 1$才會成立，因此目標函數值即加上$\bar{c}_i y = \bar{c}_i$ 。

15.9 由0或1變數決定某條件須否滿足之束制式

假設某條件$g \le 0$在$y = 1$時必須滿足；在$y = 0$時不一定要滿足。現若可確定g不會大於$\overline{d}$ ，參考第15.7節之做法，可用束制式$g - \overline{d}(1 - y) \le 0$達成。因$y = 1$時該束制式即為$g \le 0$ ；而$y = 0$時g就不受≤ 0之限制。若須滿足之條件為$g < 0$ ，則改以$g + \epsilon \le 0$代替，其中ϵ為很小之正值，當g為整數時可用$\epsilon = 1$ 。各種情形之組合共有八種均列於下表。表中亦

列示另一類用法為：以第一種束制式為例，該束制式可使條件$g > 0$成立時$y = 0$；條件$g > 0$不成立時y可以等於0或1。

if	$then$	if	$then$	by	$assume$
$y = 1$	$g \le 0$	$g > 0$	$y = 0$	$g - \overline{d}(1-y) \le 0$	$g \le \overline{d}$
$y = 1$	$g \ge 0$	$g < 0$	$y = 0$	$g + \underline{d}(1-y) \ge 0$	$g \ge -\underline{d}$
$y = 1$	$g < 0$	$g \ge 0$	$y = 0$	$g + \epsilon - \overline{d}(1-y) \le 0$	$g < \overline{d}$
$y = 1$	$g > 0$	$g \le 0$	$y = 0$	$g - \epsilon + \underline{d}(1-y) \ge 0$	$g > -\underline{d}$
$y = 0$	$g \le 0$	$g > 0$	$y = 1$	$g - \overline{d}\,y \le 0$	$g \le \overline{d}$
$y = 0$	$g \ge 0$	$g < 0$	$y = 1$	$g + \underline{d}\,y \ge 0$	$g \ge -\underline{d}$
$y = 0$	$g < 0$	$g \ge 0$	$y = 1$	$g + \epsilon - \overline{d}\,y \le 0$	$g < \overline{d}$
$y = 0$	$g > 0$	$g \le 0$	$y = 1$	$g - \epsilon + \underline{d}\,y \ge 0$	$g > -\underline{d}$

15.10 由某條件產生0或1變數值之束制式

由上二節之處理方式可用以根據條件$g \le 0$ 是否成立產生0或1之y值，如下表所示之組合共有四種。方法一類似第15.8節之處理方式，僅用一個束制式，但在目標函數增加一項，因此在求得最佳解後應予調整回原來之目標函數值。方法二採用第15.9節之二種條件之組合，因此須同時用二個束制式。例如第一種束制式可使：條件$g \le 0$成立時$y = 1$；不成立時$y = 0$。該束制式亦可用於：$y = 1$時條件$g \le 0$須滿足；但$y = 0$時條件$g \le 0$不可成立，即條件$g > 0$須滿足。

iif	$then$	$by\ sec(15.8)$	$or\ sec(15.9)$	$assume$
$g \le 0$	$y = 1$	$minimize \quad -\bar{c}\,y$	$g - \epsilon + \underline{d}\,y \ge 0$	$g > -\underline{d}$
$g > 0$	$y = 0$	$g - \overline{d}(1-y) \le 0$	$g - \overline{d}(1-y) \le 0$	$g \le \overline{d}$
$g \ge 0$	$y = 1$	$minimize \quad -\bar{c}\,y$	$g + \epsilon - \overline{d}\,y \le 0$	$g < \overline{d}$
$g < 0$	$y = 0$	$g + \underline{d}(1-y) \ge 0$	$g + \underline{d}(1-y) \ge 0$	$g \ge -\underline{d}$
$g \le 0$	$y = 0$	$minimize \quad \bar{c}\,y$	$g - \epsilon + \underline{d}(1-y) \ge 0$	$g > -\underline{d}$
$g > 0$	$y = 1$	$g - \overline{d}\,y \le 0$	$g - \overline{d}\,y \le 0$	$g \le \overline{d}$
$g \ge 0$	$y = 0$	$minimize \quad \bar{c}\,y$	$g + \epsilon - \overline{d}(1-y) \le 0$	$g < \overline{d}$
$g < 0$	$y = 1$	$g + \underline{d}\,y \ge 0$	$g + \underline{d}\,y \ge 0$	$g \ge -\underline{d}$

15.11 離散數線性規劃程式

［表一］ 離散數線性規劃法副程式與試用程式

```
************************************************************************
      PROGRAM DLPTST
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(51,61),LLI(51),LLJ(61),D(110),X(110)
      DIMENSION A(51,61),LI(51),LJ(61)
      DIMENSION T(51,61),MI(51),MJ(61)
      DIMENSION DLU(3,61),ITYP(3,61),VAL(1000)
      DIMENSION ZNOD(1000),MVX(3,1000)
      DATA NIA,NJA,NVAL/51,61,1000/,IS,IC/0,6/
C**   ============================================================    **
C**   Input data -----+------+--------------------+----------         **
C**               SEQ | SETS | VARIABLES          |  FORMAT          **
C**              -----+------+--------------------+----------         **
C**                1  |  1   | IS,IC              |  2I5             **
C**                2  |  1   | II,JJ,NY           |  3I5             **
C**                3  |  II  | (A(I,J),J=1,JJ)    |  8F10.0          **
C**                4  |  1   | (D(J),J=1,NY)      |  8F10.0          **
C**                5  |  1   | (LI(I),I=1,II-1)   |  16I5            **
C**                6  |  1   | (LJ(J),J=1,JJ-1)   |  16I5            **
C**                7  |  1   | (ITYP(J),J=1,JJ-1) |  16I5            **
C**                8  |  *   | (VAL(K,*),K=1,m)   |  8F10.0          **
C**              -----+------+--------------------+----------         **
C**   IS    = File# to save origi. tableau (no re-perform if IS=0)    **
C**   IC    = File# to save debug output (no debug output if IC=0)    **
C**   II,JJ = Numbers of rows and columns of simplex tableau          **
C**   NY    = Maximum variable # which have upper bound to input      **
C**           (may include slack var but not artificial var)         **
C**   ------------------------------------------------------------    **
C**   A(I,J)            = Simplex tableau                             **
C**   I=1,II-1;J=1,JJ-1 : For coeff. of variables : A(I,J)           **
C**   I=1,II-1;J=JJ     : For RHS constants       : B(I) = A(I,JJ)    **
C**   I=II    ;J=1,JJ-1 : For objective function  : C(J) = A(II,J)    **
C**   ------------------------------------------------------------    **
C**   D(J)  = Upper bound value for variable I                        **
C**   ------------------------------------------------------------    **
C**   LI(I) = Constaint equation type                                 **
C**         = +1  for lesser  or equal constraint                     **
C**         = -1  for greater or equal constraint                     **
C**         =  0  for equal constarint                                **
C**   LJ(J) = Variable # for column J of the table                    **
C**         = 0  will be reset to J by this program                   **
C**   ITYP(J) = 0 : Real variable                                     **
C**           = 1 : 0/1 integer var (D(J)=1 set by this program)      **
C**           = 2 : Integer  var -> 0,1,2,...,D(J)                    **
C**           =-m : Discrete var -> 0,VAL(1),VAL(2),..,VAL(m),D(J)    **
C**   VAL(1:m,*) =  Discrete values for discrete variable             **
C**   ============================================================    **
   10 READ (*,'(11I5)') IS,IC
      IF(IS.GT.0) OPEN (IS,FILE='IDP.TMP',STATUS='NEW'
     *                 ,FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
      READ (*,'(11I5)') II,JJ,NY
      IF(II.EQ.0.OR.II.GE.NIA.OR.JJ.GT.NJA) STOP
      DO 15 I=1,II
   15 READ (*,'(11F10.0)') (A(I,J),J=1,JJ)
```

```
      READ (*,'(11F10.0)') (D(J),J=1,NY)
      READ (*,'(11I5)')    (LI(I),I=1,II-1)
      READ (*,'(11I5)')    (LJ(J),J=1,JJ-1)
      READ (*,'(11I5)')    (ITYP(1,J),J=1,JJ-1)
C**   +--------------------------------------------------------+    **
C**   | Comput NX and fill D(NY+1:NX)=0                        |    **
C**   | Setup LI(II-1),LJ(JJ-1),D(NX) for simplex tableau      |    **
C**   +--------------------------------------------------------+    **
      DO 20 J=1,JJ-1
      IF(LJ(J).EQ.0) LJ(J)=J
   20 CONTINUE
      NX=JJ-1
      DO 25 I=1,II-1
      IF(LI(I).EQ.0) GO TO 25
      NX=NX+1
      IF(IABS(LI(I)).EQ.1) LI(I)=ISIGN(NX,LI(I))
   25 CONTINUE
      DO 30 J=1,NX
      IF(J.GT.NY) D(J)=0.0
      IF(D(J).LE.0.0) D(J)=1.0D30
   30 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Read discrete values VAL(1:m,*) for ITYP=-m             |    **
C**   +--------------------------------------------------------+    **
      KNEXT=1
      DO 40 J=1,JJ-1
      IF(ITYP(1,J).EQ.1) D(J)=1.0
      IF(ITYP(1,J).LT.0) THEN
        KSTRT=KNEXT
        KNEXT=KSTRT-ITYP(1,J)
        IF(KNEXT.GT.NVAL) STOP 'VAL(*) TOO SMALL'
        ITYP(2,J)  =KSTRT
        READ (*,'(8F10.0)') (VAL(K),K=KSTRT,KNEXT-1)
      ENDIF
   40 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Write data before calling IDP                          |    **
C**   +--------------------------------------------------------+    **
      WRITE(*,'('' ********** DATA BEFORE CALLING IDP **********'')')
      DO 45 I=1,II
   45 WRITE(*,'(1X,1P11E10.3)') (A(I,J),J=1,JJ)
      WRITE(*,'(1X,1P11E10.3)') (D(J),J=1,NX)
      WRITE(*,'(1X,11I5)')      (LI(I),I=1,II-1)
      WRITE(*,'(1X,11I5)')      (LJ(J),J=1,JJ-1)
      WRITE(*,'(1X,11I5)')      (ITYP(1,J),J=1,JJ-1)
C**   +--------------------------------------------------------+    **
      CALL DLP(A,LI,LJ,D,X,II,JJ,NIA,NJA,AA,LLI,LLJ,T,MI,MJ,IS,IC
     *        ,ICASE,DLU,ITYP,VAL,ZNOD,MVX,NVAR,NX)
C**   +--------------------------------------------------------+    **
C**   | Write data after calling IDP                           |    **
C**   +--------------------------------------------------------+    **
      WRITE(*,'('' ========== DATA AFTER  CALLING IDP =========='')')
      DO 50 I=1,II
   50 WRITE(*,'(1X,1P11E10.3)') (AA(I,J),J=1,JJ)
      WRITE(*,'(1X,1P11E10.3)') (D(J),J=1,NX)
      WRITE(*,'(1X,11I5)')      (LLI(I),I=1,II-1)
      WRITE(*,'(1X,11I5)')      (LLJ(J),J=1,JJ-1)
      WRITE(*,'(1X,11I5)')      (ITYP(1,J),J=1,JJ-1)
C**   +--------------------------------------------------------+    **
C**   | Output feasible optimal solution                       |    **
```

```
C**   +-------------------------------------------------------+    **
      IF(ICASE.NE.0) GO TO 10
      WRITE(*,'('' VALUES OF VARIABLES =''))')
      WRITE(*,'(1X,1P8E10.3)') (X(J),J=1,NX)
      WRITE(*,'('' OBJECTIVE FUNCTION  =''),1PE15.8)') -AA(II,JJ)
      GO TO 10
      END
*********************************************************************
      SUBROUTINE DLP(AA,LLI,LLJ,D,X,II,JJ,NIA,NJA,A,LI,LJ,T,MI,MJ,IS,IC
     *              ,ICASE,DLU,ITYP,VAL,ZNOD,MVX,NVAR,NX)
C**   ===========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(NIA,NJA),LLI(NIA),LLJ(NJA),D(1),X(1)
      DIMENSION A(NIA,NJA),LI(NIA),LJ(NJA)
      DIMENSION T(NIA,NJA),MI(NIA),MJ(NJA)
      DIMENSION DLU(3,1),ITYP(3,1),VAL(1)
      DIMENSION ZNOD(100),MVX(3,100)
C**   ===========================================================    **
C**   DIMENSION DLU(3,NVAR),ITYP(3,NVAR),VAL(1),D(NX),X(NX)          **
C**   -----------------------------------------------------------    **
C*I   AA(II,JJ)   = Simplex tableau                                  **
C*I   LLI(II-1)   = -1/0/1 : Constraint type GE/EQ/LE                **
C*O               = -s/0/s : s/a = slack/artificail variable         **
C*I   LLJ(JJ-1)   = 0/v : Variable for columns                       **
C*O               = v   : Set by this subroutine when input is 0     **
CIO   D(NX)       = Upper bound for each variable in simplex         **
C*O   X(NX)       = Solution vector                                  **
C*I   II,JJ       = Number of rows and columns in simplex tableau    **
C*I   NIA,NJA     = Dimensions of A(NIA,NJA)                         **
C*W   A,LI,LJ     = Computing array for simplex method               **
C*W   T,MI,MJ     = Working array for simplex method                 **
C**   -----------------------------------------------------------    **
C*W   DLU(3  ,IV) = Given upper bound = D(IV) for IV <= NVAR          **
C*W   DLU(1:2,IV) = Lower:Upper bound of variables(integer bound)     **
C*I   ITYP(1, IV) = Variable type :                                  **
C*I               = 0 : Real                                         **
C*I               = 1 : Integer 0 or 1                               **
C*I               = 2 : Integer 0, 1, 2, ...                         **
C*I               =-m : Discrete value in table VAL(1:m,*)           **
C*I   ITYP(2, IV) = Pointer to the table VAL(*) for ITYP(1,IV)=-m     **
C*I   ITYP(3, IV) = Lower integer value of non-integer variable      **
C*I   VAL(*)      = Discrete values for discrete variables           **
C*W   ZNOD(NODE)  = Objective value at NODE                          **
C*W   MVX(3,NODE) = Mother/JV/JX                                     **
C*W   NVAR        = JJ - 1 = Number of variables (No slack var)      **
C*W   NX          = Number of variables include slack variables      **
C**   -----------------------------------------------------------    **
C*W   ZOPT        = Z_optimal so far                                 **
C*W   NOPT        = Node of the ZOPT                                 **
C*W   ZMIN        = Minimum Z from all active node                   **
C*W   NMIN        = Node of the ZMIN                                 **
C*W   NODE        = Number of nodes have been done                   **
C*W   IV/IX       = Integer variable/value for current node          **
C*W   JV/JX       = Integer variable/value for the left node         **
C*I   IS,IC       = File#/0/-1,File#/0 for subroutine LINEAR         **
C**   ===========================================================    **
      IV=0
      IX=0
      NODE=0
      NMIN=0
```

```
      NOPT=0
      ZMIN=1.0D29
      ZOPT=1.0D30
      CALL SETLIJ(LLI,LLJ,D,DLU,II,JJ,NX,NIA,NJA,NVAR)
C**   +-------------------------------------------------------+    **
C**   | Set DLU(1:2,Nvar) = Lower:Upper bound of all variables |    **
C**   | Set DLU(1:2,IV) for integer variable at left/right node |    **
C**   | Set A(NIA,NJA),LI(NIA),LJ(NJA),D(NX)                  |    **
C**   | Perform linear programming computation                |    **
C**   +-------------------------------------------------------+    **
   20 CALL SETBLU(NMIN,MVX,DLU,ITYP,VAL,NVAR)
      CALL ADDNOD(NODE,IV,IX,DLU,ITYP,VAL,NVAR)
      CALL SETSMP(A,LI,LJ,D,NX,II,JJ,NIA,NJA,AA,LLI,LLJ,DLU,NVAR,ICASE)
      IF(ICASE.GE.0)
     *CALL LINEAR(A,LI,LJ,D,X,II,JJ,NX,ICASE,KKK,NIA,NJA,T,MI,MJ,IS,IC)
                   Z=-A(II,JJ)
      IF(ICASE.LT.0) Z= 1.0D30
      IF(ICASE.GT.0) Z=-1.0D30
C*1   WRITE(*,'('' DLP   '',1P3E10.3,I5)') ZMIN,ZOPT,Z,ICASE
C**   +-------------------------------------------------------+    **
C**   | Reset X(1:NVAR) : From discrete values to real vaules |    **
C**   +-------------------------------------------------------+    **
      CALL GETXVL(X,DLU,NVAR,ICASE)
C**   +-------------------------------------------------------+    **
C**   | Zmin = Zopt : Return after re-compute optimal solution |    **
C**   +-------------------------------------------------------+    **
      IF(ZMIN.GE.ZOPT) RETURN
C**   +-------------------------------------------------------+    **
C**   | Integer check and Get next integer var/val JV/JX      |    **
C**   | Save result of new node at MVX(3,++NODE)              |    **
C**   +-------------------------------------------------------+    **
      CALL INTCHK(X,ITYP,VAL,NVAR,JV,JX,ICASE)
      CALL SAVNOD(NODE,ZNOD,MVX,Z,NMIN,JV,JX)
C**   +-------------------------------------------------------+    **
C**   | JV = 0 : All integer solution                         |    **
C**   +-------------------------------------------------------+    **
      IF(JV.EQ.0) THEN
        IF(Z.LT.ZOPT) THEN
          ZOPT=Z
          NOPT=NODE
        ENDIF
      ENDIF
      IF(NODE/2*2.EQ.NODE) GO TO 20
C**   +-------------------------------------------------------+    **
C**   | After analyze left and right nodes                    |    **
C**   | Find next node with minimum Z : NMIN,ZMIN,IV,IX       |    **
C**   +-------------------------------------------------------+    **
      CALL NXTNOD(NODE,ZNOD,MVX,NOPT,NMIN,ZMIN,IV,IX)
      GO TO 20
      END
*************************************************************************
      SUBROUTINE SETBLU(NMIN,MVX,DLU,ITYP,VAL,NVAR)
C**   ========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION MVX(3,1),DLU(3,NVAR),ITYP(3,NVAR),VAL(1)
C**   ========================================================    **
C**   Set DLU(1:2,IV) = Lower:Upper bound of variable IV          **
C**   --------------------------------------------------------    **
C**   MVX(1,NODE) = Node number of mother node                    **
C**   MVX(2,NODE) = JV : Next integer variable (active if JV>=0)  **
```

```
C**   MVX(3,NODE) = JX : Next integer value                          **
C**   ================================================================    **
C**   +----------------------------------------------------------+    **
C**   | Reset lower/upper bound of all variable                  |    **
C**   +----------------------------------------------------------+    **
      DO 20 IV=1,NVAR
      DLU(1,IV)=0.0
      DLU(2,IV)=DLU(3,IV)
   20 CONTINUE
      IF(NMIN.LE.1) RETURN
C**   +----------------------------------------------------------+    **
C**   | Tracking back to set lower/upper bound by mother nodes   |    **
C**   +----------------------------------------------------------+    **
      NODE=NMIN
   50 MO=MVX(1,NODE)
      IF(MO.EQ.0) RETURN
      IV=IABS(MVX(2,MO))
      IX=MVX(3,MO)
      IF(NODE/2*2.NE.NODE) IX=IX+1
C**   +----------------------------------------------------------+    **
C**   | Get XVAL = true value of variable IV at integer IX       |    **
C**   +----------------------------------------------------------+    **
      CALL INTVAL(IV,IX,XVAL,DLU,ITYP,VAL,NVAR)
C**   +----------------------------------------------------------+    **
C**   | NODE = 2, 4, 6, ...  for left node : X(IV) <= XVAL       |    **
C**   +----------------------------------------------------------+    **
      IF(NODE/2*2.EQ.NODE) THEN
        DLU(2,IV)=DMIN1(DLU(2,IV),XVAL)
      ELSE
        DLU(1,IV)=DMAX1(DLU(1,IV),XVAL)
      ENDIF
C**   +----------------------------------------------------------+    **
C**   | Back to Mother node at upper level                       |    **
C**   +----------------------------------------------------------+    **
      NODE=MO
      GO TO 50
      END
*************************************************************************
      SUBROUTINE ADDNOD(NODE,JV,JX,DLU,ITYP,VAL,NVAR)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DLU(3,NVAR),ITYP(3,NVAR),VAL(1)
C**   ================================================================    **
C**   Set upper/lower bound for left/right node                       **
C**   ================================================================    **
      IF(JV.LE.0) RETURN
      IF(NODE/2*2.EQ.NODE) JX=JX+1
      CALL INTVAL(JV,JX,XVAL,DLU,ITYP,VAL,NVAR)
      IF(NODE/2*2.NE.NODE) THEN
        DLU(2,JV)=DMIN1(DLU(2,JV),XVAL)
      ELSE
        DLU(1,JV)=DMAX1(DLU(1,JV),XVAL)
      ENDIF
C*1   WRITE(*,'('' ADDNOD'',1P3E10.3)') DLU
      RETURN
      END
*************************************************************************
      SUBROUTINE NXTNOD(NODE,ZNOD,MVX,NOPT,NMIN,ZMIN,JV,JX)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      DIMENSION ZNOD(NODE),MVX(3,NODE)
C**   ============================================================    **
C**   Find node NMIN with minimum ZNOD(NMIN) from active node         **
C**   ------------------------------------------------------------    **
C*O   NMIN  = Node number for next starting node                      **
C*O   ZMIN  = Objective value Z of the node NMIN                      **
C*O   JV,JX = Next integer variable/value (JV<0 : infeasible)         **
C**   ============================================================    **
      NMIN=NODE
      IF(NOPT.NE.0) NMIN=NOPT
      ZMIN=ZNOD(NMIN)
      DO 20 N=1,NODE
      IF(MVX(2,N).LT.0) GO TO 20
      IF(ZNOD(N).GE.ZMIN) GO TO 20
        NMIN=N
        ZMIN=ZNOD(N)
   20 CONTINUE
      JV=MVX(2,NMIN)
      JX=MVX(3,NMIN)
C*1   WRITE(*,'('' NXTNOD'',1PE10.3,4I5)')
C*1  *            (ZNOD(N),N,(MVX(K,N),K=1,3),N=1,NODE)
C*1   WRITE(*,'('' NXTNOD'',1PE10.3,3I5)') ZMIN,NMIN,JV,JX
      RETURN
      END
************************************************************************
      SUBROUTINE SAVNOD(NODE,ZNOD,MVX,Z,NMIN,JV,JX)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ZNOD(1),MVX(3,1)
C**   ============================================================    **
C**   At NODE=NODE+1 : Save node data : ZNOD(NODE) & MVX(3,NODE)      **
C**   ============================================================    **
      NODE=NODE+1
      ZNOD(NODE)=Z
      MVX(1,NODE)=NMIN
      MVX(2,NODE)=JV
      MVX(3,NODE)=JX
C**   +----------------------------------------------------------+    **
C**   | Set mother node to non-active (negative MVX(2,NMIN))     |    **
C**   +----------------------------------------------------------+    **
      MVX(2,NMIN)=-IABS(MVX(2,NMIN))
C*1   WRITE(*,'('' SAVNOD'',1PE10.3,4I5)')
C*1  *             ZNOD(NODE),NODE,(MVX(K,NODE),K=1,3)
      RETURN
      END
************************************************************************
      SUBROUTINE INTVAL(IV,IX,XVAL,DLU,ITYP,VAL,NVAR)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DLU(3,NVAR),ITYP(3,NVAR),VAL(1)
C**   ============================================================    **
C*I   IV   = Integer variable                                        **
C*I   IX   = Integer value of the var IV                             **
C*O   XVAL = Real value of the var IV = IX or VAL(IX+ITYP(2,IV)-1)   **
C*I   ITYP(1,IV) = 0 : Real value                                    **
C**              = 1 : 0 or 1                                        **
C**              = 2 : Integer value : 0, 1, 2, ...                  **
C**              =-m : Use discrete values for intermidiate values   **
C**                    in VAL(IYPE(2,IV):ITYP(2,IV)+m-1)             **
C**   ============================================================    **
```

```
      IF(ITYP(1,IV).EQ.0) THEN
        STOP 2222
      ELSE IF(ITYP(1,IV).GT.0) THEN
        XVAL=IX
      ELSE IF(ITYP(1,IV).LT.0) THEN
        ID=ITYP(2,IV)+IX-1
        IF(IX.LE.0) THEN
          XVAL=0.0
        ELSE IF(IX.LE.IABS(ITYP(1,IV))) THEN
          XVAL=VAL(ID)
        ELSE
          XVAL=DLU(3,IV)
        ENDIF
      ENDIF
      RETURN
      END
************************************************************************
      SUBROUTINE INTCHK(X,ITYP,VAL,NVAR,JV,JX,ICASE)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(NVAR),ITYP(3,NVAR),VAL(1)
      DATA EPS/1.0D-6/
C**   ============================================================    **
C*I   ICASE      = -1/0/1 : Infeasible/feasible/unbounded             **
C*O   ITYP(3,IV) = -1 : Variable is integer/discrete                  **
C*O              >= 0 : Variable is not integer/discrete             **
C*O   JV,JX      =  Next integer variable/value (JV=-1:infeasible)   **
C**   ============================================================    **
      JV=-1
      JX=0
      IF(ICASE.LT.0) RETURN
      DO 50 IV=1,NVAR
      ITYP(3,IV)=-1
      IF(ITYP(1,IV).EQ.0) THEN
        ITYP(3,IV)=-1
      ELSE IF(ITYP(1,IV).GT.0) THEN
        IX=X(IV)+EPS
        IF(DABS(IX-X(IV)).GT.EPS) ITYP(3,IV)=X(IV)
      ELSE IF(ITYP(1,IV).LT.0) THEN
        VALS=0.0
        ISTRT=ITYP(2,IV)
        ISTOP=ISTRT-ITYP(1,IV)-1
        DO 20 ID=ISTRT,ISTOP
          IF(X(IV).LE.VAL(ID)) THEN
            XI=(X(IV)-VALS)/(VAL(ID)-VALS)
            IX=XI+EPS
            IF(DABS(IX-XI).GT.EPS) ITYP(3,IV)=ID-ISTRT
            GO TO 50
          ENDIF
          VALS=VAL(ID)
   20   CONTINUE
      ENDIF
   50 CONTINUE
C**   +----------------------------------------------------------+    **
C**   | Find JV/JX : Non-integer variable/value for left node    |    **
C**   +----------------------------------------------------------+    **
C*1   WRITE(*,'('' INTCHK'',10I5)') (ITYP(1,IV),IV=1,NVAR)
C*1   WRITE(*,'('' INTCHK'',10I5)') (ITYP(2,IV),IV=1,NVAR)
C*1   WRITE(*,'('' INTCHK'',10I5)') (ITYP(3,IV),IV=1,NVAR)
      JV=0
```

```
      DO 60 IV=1,NVAR
      IF(ITYP(3,IV).LT.0) GO TO 60
        JV=IV
        JX=ITYP(3,IV)
C*1     WRITE(*,'('' INTCHK'',2I5)') JV,JX
        RETURN
   60 CONTINUE
      RETURN
      END
*********************************************************************
      SUBROUTINE SETSMP
     *           (A,LI,LJ,D,NX,II,JJ,NIA,NJA,AA,LLI,LLJ,DLU,NVAR,ICASE)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(NIA,NJA),LLI(NIA),LLJ(NJA)
      DIMENSION A(NIA,NJA),LI(NIA),LJ(NJA),D(NX)
      DIMENSION DLU(3,NVAR)
C**   ============================================================    **
C**   Input : AA,LLI,LLJ;  Output : A,LI,LJ,D of simplex tableau      **
C**   ------------------------------------------------------------    **
C*O   ICASE = -1 : for infeasible case due to XL > XU (= 0 else)      **
C**   ============================================================    **
      DO 50 J=1,JJ
      LJ(J)=LLJ(J)
      DO 40 I=1,II
      A(I,J)=AA(I,J)
   40 CONTINUE
   50 CONTINUE
      DO 60 I=1,II
      LI(I)=LLI(I)
   60 CONTINUE
      ICASE=0
      DO 80 J=1,JJ-1
      D(J)=DLU(2,J)-DLU(1,J)
      IF(D(J).LT.0.0) ICASE=-1
      IF(DLU(1,J).EQ.0.0) GO TO 80
      DO 70 I=1,II
      A(I,JJ)=A(I,JJ)-A(I,J)*DLU(1,J)
   70 CONTINUE
   80 CONTINUE
      RETURN
      END
*********************************************************************
      SUBROUTINE GETXVL(X,DLU,NVAR,ICASE)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(NVAR),DLU(3,NVAR)
C**   ============================================================    **
C**   Transform solution of Simplex tableau to Original equation      **
C**   ============================================================    **
      IF(ICASE.LT.0) RETURN
      DO 20 IV=1,NVAR
      X(IV)=X(IV)+DLU(1,IV)
   20 CONTINUE
C*1   WRITE(*,'('' GETVAL'',1P8E10.3)') X
C*1   WRITE(*,'('' GETVAL'',1P3E10.3)') DLU
      RETURN
      END
*********************************************************************
      SUBROUTINE SETLIJ(LI,LJ,D,DLU,II,JJ,NX,NIA,NJA,NVAR)
```

```
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION LI(NIA),LJ(NJA),D(NJA),DLU(3,1)
C**   ============================================================    **
C*I   LI(I)    =-1/0/1 : GE/EQ/LE constraint for row I                **
C*O           =-s/0/s : s = Slack variable number > NVAR              **
C*I   LJ(J)    =  0/K  : K = Variable number for column J              **
C*O           =  J/K  : J = Variable number for column J              **
C*I   D(K)     =  0/U  : U = Upper bound for variable K                **
C*O           =  M/U  : M = 1.0D30 for no-upper-bound variable K      **
C*O   DLU(3,K) =  D(K) : For K <= NVAR                                 **
C*I   II       =  Number of total rows in A(II,JJ) <= NIA              **
C*I   JJ       =  Number of total cols in A(II,JJ) <= NJA              **
C*O   NVAR     =  JJ - 1                                               **
C*O   NX       =  NVAR + Slack variables                               **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Set NVAR,NX,LI(II-1),LJ(JJ-1),D(NX) for simplex tableau  |    **
C**   | Save D(1:NVAR) in DLU(3:3,1:NVAR)  for int bounds setup  |    **
C**   +----------------------------------------------------------+    **
      DO 30 J=1,JJ-1
      IF(LJ(J).EQ.0) LJ(J)=J
   30 CONTINUE
      NVAR=JJ-1
      NX=JJ-1
      DO 35 I=1,II-1
      IF(LI(I).EQ.0) GO TO 35
      NX=NX+1
      IF(IABS(LI(I)).EQ.1) LI(I)=ISIGN(NX,LI(I))
   35 CONTINUE
      DO 38 J=1,NX
      IF(D(J).LE.0.0) D(J)=1.0D30
      IF(J.LE.NVAR) DLU(3,J)=D(J)
   38 CONTINUE
      RETURN
      END
*************************************************************************
```

習題

1. 試解下列之整數線性規劃問題並繪其樹枝圖。

$$
\begin{aligned}
\text{最小化} \quad z &= -2x_1 - 4x_2 \\
\text{受制於} \quad & 6x_1 + 2x_2 \le 27 \\
& -2x_1 + 8x_2 \le 16 \\
& x_1, x_2 = 0, 1, 2, \ldots
\end{aligned}
$$

2. 試將所附程式做下列之修改：在副程式$INTCHK$中，選取子節點之束制變數為：其變數值為非整數但最接近整數者。本章所附程式係選變數號碼較小者。

參考文獻

1. Land, A.H. and Doig, A., "An Automatic Method of Solving Discrete Linear Programming Problems", Econometrica, Vol.28, pp.497-520, 1960.

2. Gomory, R.E., "An Algorithm for the Mixed Integer Problem", The Rand Corporation, RM-2597, 1960.

3. McMillan, Claude Jr., *Mathematical Programming : An Introduction to the Design and Application of Optimal Decision Machines,* Jojn Wiley Sons, Inc., 1970.

第十六章

零壹規劃問題

16.1 前言

當線性規劃問題之所有變數之解僅能為0或1之值時，即稱為零壹線性規劃問題 (Zero–one linear programming)。前一章之整數線性規劃問題雖然也可以解零壹線性規劃問題，但因零壹線性規劃問題之解可得自一有限量之變數值組合情形中，如N_v個變數之組合總數為2^{N_v}，較之整數線性規劃問題之變數值組合(可能為無限多)為少，應該可以用更有效之方法以求解零壹線性規劃問題。本章將介紹之巴拉斯(Balas)法即能夠以有效之評估方法由2^{N_v}種組合中找出最佳解。

理論上將變數值直接代入束制式即可判斷該等變數值是否滿足束制條件，即是否合理，此方式稱為顯式評估。如不合理即不可能為最佳解；如為合理解，則必須再確定其目標函數值比所有其他合理解者為佳，才可確定該解為最佳解。理論上如對2^{N_v}種組合之變數值做顯式評估，除無合理解外，必可由合理解中找到最佳解。將某些變數固定為某些特定值時，稱這些變數為定值變數，稱其他變數為自由變數。若能由簡單計算比較而確定自由變數不管採用何值，均不可能滿足束制條件或其目標函數值不可能比已求得之合理解者更佳時，令N_f為自由變數之數目，即可不必對2^{N_f}個變數組合做顯式評估。此方式即稱隱式評估。利用隱式評估一般可以很有效率地確定很多變數組合不可能為最佳解，故只須做相當有限數量之顯式評估，即可求得最佳解。

零壹線性規劃問題亦可解下列非線性規劃問題：(1)零壹非線性規劃問題；(2)整數非線性規劃問題。因該二類非線性規劃問題可經一些轉換技巧而成為零壹線性規劃問題。有關之轉換技巧亦將於本章略加介紹。

16.2 零壹線性規劃問題

考慮下列之零壹線性規劃問題：

$$\text{最小化} \quad z = \sum_{j=1}^{N_v} c_j x_j + d \tag{16.1}$$

$$\sum_{j=1}^{N_v} a_{ij} x_j + b_i \geq 0 \,, \quad i = 1, \ldots, N_c \tag{16.2}$$

$$x_j = 0 \ \text{ or } \ 1 \,, \qquad j = 1, \ldots, N_v \tag{16.3}$$

為了計算與分析上之方便，式(16.1)中之c_j均不能為負值，且束制式均為≥ 0之條件，目標函數須為極小化。本章討論之問題均假設為此型式。

束制式如為≤ 0之條件，可將所有係數及常數變號，即可改為≥ 0之條件。目標函數如為極大化，亦可將所有係數及常數變號，即可改為極小化之目標函數。如c_j為負值，可將對應變數x_j改為$1 - x'_j$，即可使新變數x'_j之係數$-c_j$變成正值。注意目標函數中：常數項應改為$d + c_j$。束制式中：常數項應改為$b_i + a_{ij}$，x'_j之係數亦應變號改為$-a_{ij}$。

16.3 可由視察法求解之零壹線性規劃問題

本節將列舉四種可以由視察法求解或求得部分變數值之特殊問題：
(1)第一種特殊問題：束制式常數項均≥ 0。為變數值全為0之合理解。
下列之零壹線性規劃問題，可以明顯看出其最佳解為$x_1 = x_2 = x_3 = 0$，即變數值全為0，其對應之最小目標函數值為$z = 0$。因為(1)目標函數之係數$c_j = (5, 2, 1)$全為正值；及(2)束制式之常數項$b_i = (1, 5, 7)$全非負值，即≥ 0。故由(1)知任何變數由0變為1均會使z大於0而非最佳解；由(2)知所有變數為0時束制條件均會滿足。

$$\begin{aligned} z &= 5x_1 + 2x_2 + x_3 \\ & 5x_1 + 4x_2 - 2x_3 + 1 \geq 0 \qquad (\ 1 \geq 0\ ) \\ & 4x_1 - 2x_2 + 3x_3 + 5 \geq 0 \qquad (\ 5 \geq 0\ ) \\ & -x_1 + 2x_2 + 4x_3 + 7 \geq 0 \qquad (\ 7 \geq 0\ ) \end{aligned}$$

(2)第二種特殊問題：可經簡單計算確定無合理解。
經由下列之簡單計算可知某一束制式是否可能滿足：將b_i加上所有正值之

a_{ij}，若其和為負值，則可確定該束制條件不可能滿足。因該算法所得之值為變數值為0或1之條件下束制式左邊之可能最大值。故若能經此簡單計算確定任一束制條件不可能滿足，即可確定該問題無合理解。現計算下列三束制式之值為$-1+5+4=8$，$-5+4+3=2$，$-7+2+4=-1$，其中第3值為負(-1)，故知束制式3不可能滿足。因此該問題無合理解。

$$\begin{aligned} z \;=\; & 5x_1+2x_2+\;x_3 \\ & 5x_1+4x_2-2x_3-1\geq 0 \qquad (\;-1+5+4\geq 0\;) \\ & 4x_1-2x_2+3x_3-5\geq 0 \qquad (\;-5+4+3\geq 0\;) \\ & -x_1+2x_2+4x_3-7\geq 0 \qquad (\;-7+2+4\not\geq 0\;) \end{aligned}$$

上述二種問題確實過分簡單，實際應用可能不會遇到這樣簡單的問題，本節特別說明這二種特殊問題自然有其用處，而且大有用處。一旦遇到這樣的問題，馬上可以省掉$2^{N_v}-1$種可能之變數值組合之顯式評估。因此上述二種情形之簡單計算及判斷，對任何問題而言都應該視為必做之步驟。本章將介紹之巴拉斯法會以一定的規則，設定一些變數之值為0或1，這些變數稱為定值變數，其他變數則稱為自由變數。將定值變數之值代入原問題，即可得一僅含自由變數之新問題。由此方式所得之新問題為數可觀，因此就有很多機會遇到這二種特殊問題。事實上巴拉斯法就是靠這二種特殊問題而省去很多的變數值組合之顯式評估。例如第16.8節之範例，若將變數x_1固定為0，將$x_1=0$代入該式，即可得上述之第二種特殊問題(僅變數號碼差1)，由此可不經$2^3-1=7$種變數值組合之顯式評估，而確定$x_1=0$時不可能有合理解。

(3)第三種特殊問題：目標函數有上限，可經簡單計算知某變數須為0。下列問題，若對目標函數值加一上限條件為：$z<z_{max}$，設本題之$z_{max}=6$，則經由下列之簡單計算可知某一變數是否必須等於0：計算$d+c_j$，若該值大於或等於上限值z_{max}，則可確定變數x_j必須等於0。因若$x_j=1$則對應之$z=d+c_j\geq z_{max}$即不滿足上限條件。現按下列目標函數式計算$2+5=7$，$2+2=4$，$2+1=3$，其中之$7\geq 6$，故知變數x_1必須等於0。

$$\begin{aligned} z \;=\; & 5x_1+2x_2+\;x_3+2 \qquad (\quad z \quad <6\;) \\ & 5x_1+4x_2-2x_3-1\geq 0 \qquad (\;2+5\not<6\;) \\ & 4x_1-2x_2+3x_3-5\geq 0 \qquad (\;2+2<6\;) \\ & 2x_1+2x_2+4x_3-7\geq 0 \qquad (\;2+1<6\;) \end{aligned}$$

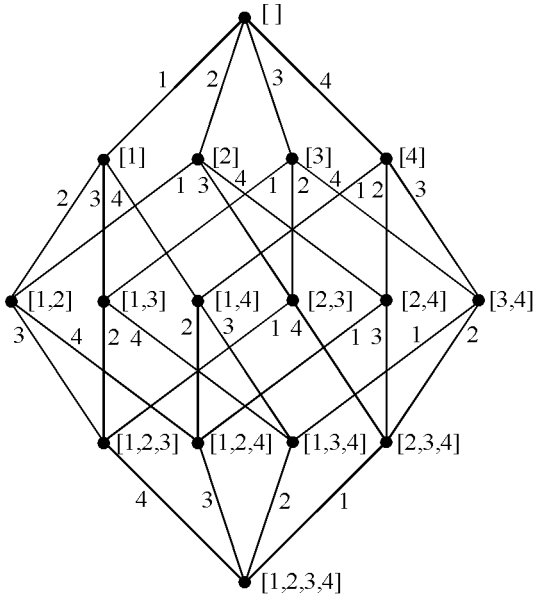
(4)第四種特殊問題：目標函數有上限，而其常數項已大於或等於上限值。例如上題之目標函數之上限若改為$z_{max}=1$，則因所有變數值為0時，所得之$z=d=2$為最小，就已經大於或等於z_{max}，故可確定其解不可能使目標函數小於上限值。注意本特殊問題也會被判斷為第三種特殊問題而逐次將所有變數固定為0，但其判斷用$d \geq z_{max}$比第三種用多次$d+c_j \geq z_{max}$簡單，故應先判斷是否為第四種特殊問題。

上述二種特殊問題對目標函數之上限條件，與前二種特殊問題一樣不太會在實際問題上出現。但在巴拉斯法之評估過程中總有機會找到合理解，則上述之z_{max}即為合理解中之最小目標函數值z_{opt}。可見一樣有很多機會遇到這種簡單問題。若能從N_v個變數中確定其中一個變數值，則可省掉2^{N_v-1}種可能之變數值組合之顯式評估。

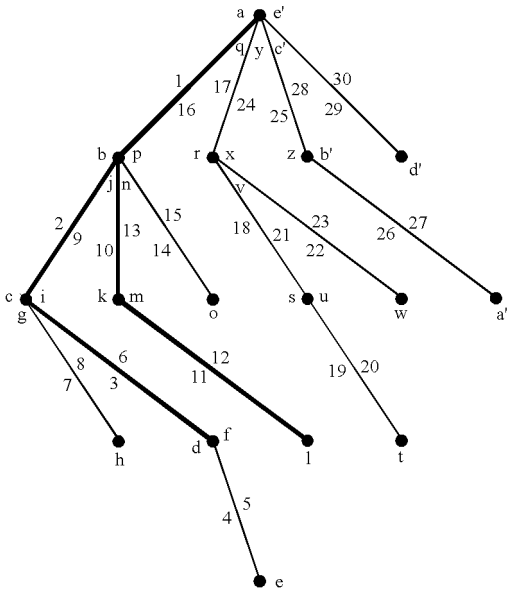
注意判斷第三種特殊問題遠較第二種特殊問題之計算簡單，故判斷是否為這四種特殊問題之順序為：第一種、第四種、第三種、第二種。

16.4 變數值組合之層次及網路

以變數數目$N_v=4$為例，變數值之組合總數為$2^4=(1+1)^4=1+4+6+4+1=16$，若將每一種組合視為一個網點(node)，將含有i個變數值為1之網點置於第i層：即第0層僅有一網點為$(0,0,0,0)$；第1層共有四網點為$(1,0,0,0)$，$(0,1,0,0)$，$(0,0,1,0)$，$(0,0,0,1)$；第2層共有六網點為$(1,1,0,0)$，$(1,0,1,0)$，$(1,0,0,1)$，$(0,1,1,0)$，$(0,1,0,1)$，$(0,0,1,1)$；第3層共有四網點為$(1,1,1,0)$，$(1,1,0,1)$，$(1,0,1,1)$，$(0,1,1,1)$；第4層僅有一網點為$(1,1,1,1)$。若上層之某一網點與下層之另一網點之變數值僅一個不相等(上層者為0；下層者為1)，則將該二網點用一網路(線)相連，即可成為圖一所示之層次及網路圖。圖中網點上標示於[]內者為變數值為1之變數號碼，稱網點變數，該網點則簡稱為某某變數網點；網路上所標示者為相連之二點中之變數值不相等之變數號碼，稱網路變數，該網路則簡稱為某變數網路。注意圖中上層網點之網點變數加上所經網路之網路變數即為下層網點之網點變數，亦即下層網點較上層者多一變數值為1之變數，此變數為網路變數。故巴拉斯法又稱巴拉斯累加法(Balas' additive algorithm)。注意網路圖中每一網點代表零壹線性規劃問題之一組可能解。



圖一 變數值組合之層次及網路



圖二 評估路線與評估網點

16.5 巴拉斯法之評估路線

巴拉斯法原則上係順著一條如圖二所示之評估路線，路線上標示之號碼為行經該路線之順序(取代圖一中標示之網路號碼)，網點上標示之字母順序則為評估該網點之順序。注意每一路線均有二個號碼，在網路左邊者屬下行路線，在右邊者屬上行路線。網點上亦有不等數目之字母，每一不同字母代表對網點之一次評估。根據已評估網點之資訊，在對同一網點做先後二次評估時，若將變數設定為不同狀態，即可反映已知之資訊，而使二次評估不同。以網點$[1]$為例：評估點b之x_1=1為定值變數，x_2, x_3, x_4為自由變數；評估點j之x_1=1，x_2=0為定值變數，x_3, x_4為自由變數；評估點n之x_1=1，x_2=x_3=0為定值變數，x_4為自由變數；評估點p之x_1=1，x_2=x_3=x_4=0為定值變數，無自由變數。自由變數值如全為0，即為評估點之變數值；如有部分自由變數值為1，則為該評估點之後但在回上層評估點之前所經評估點之變數值。例如在評估點j之後而在回上層評估點q之前所經評估點為k, l, m, n, o, p(分別在網點$[1,3]$、$[1,3,4]$、$[1,4]$與$[1]$)，其自由變數(x_3, x_4)之值：在k, m點為$(1,0)$、在l點為$(1,1)$、在o點為$(0,1)$、在n, p點為$(0,0)$；但定值變數(x_1, x_2)之值則均相同為$(1,0)$。若將j點之定值變數之值代入原問題之式中，即可得僅含自由變數

(x_3, x_4)之新問題。對該新問題可評估其是否為四種特殊問題之一，如果是第一種(變數為0之合理解)，即可確定以後三個網點不可能有更佳之合理解；如為第二種(不可能有合理解)，即可確定以後三個網點不可能有合理解；如為第三種(某變數值須為0)，設x_3變數須為0，即可確定沒有必要行經x_3網路之後之二個網點$[1,3]$與$[1,3,4]$；如為第四種(目標函數不小於已知最佳解)，即可確定以後三個網點不可能有更佳之解。對於第三種特殊問題，可跳過k,l,m之評估點而直接到評估點n。對於其他三種特殊問題，可跳過k,l,m,n,o,p之評估點而直接到評估點q。

16.6 以堆疊控制評估路線

由上節之說明可知巴拉斯法在評估某評估點時，對每一變數均設定為定值變數或自由變數，現以指標$ID(J)$按如下方式標示變數J之狀態：
(1)定值變數其值為1，簡稱為1值變數，用$ID(J)$=1標示。
(2)定值變數其值為0，簡稱為0值變數，用$ID(J)=-1$標示。
(3)自由變數，用$ID(J)$=0標示。

注意上節之評估路線有二個重要特性：(1)由上層評估點至下層評估點所走網路均不重複，(2)由下層評估點回上層評估點均走原網路。為了能夠掌控評估路線行經之網路與評估點，除變數J之狀態用$ID(J)$表示外，另外用一個堆疊(stack)存放所有定值變數之號碼，以方便找到回程網路及設定變數狀態。所謂堆疊為以先進後出(亦即後進先出)之方式存進或取出堆疊中之數據。因此若變數K較變數J先由自由變數變成固定變數而存進堆疊中，則變數K只能在變數J之後自堆疊中取出而恢復為自由變數。設堆疊為$IS(NS)$，NS為堆疊上之定值變數之數目。現以上節之評估路線，說明如何利用此堆疊與變數狀態來控制整條評估路線之行進。

以下說明請參考[表一]，表中之$a,b,c,\ldots$對應圖二之評估點之順序。
(1)在點a，所有變數均為自由變數，令$ID(1)$=$ID(2)$=$ID(3)$=$ID(4)$=0，$NS=0$。在點a向下可走網路有4條，均須為自由變數所對應之網路，稱為可行網路。按圖二選變數x_1=1之網路至點b(網路之選法另詳後述)。
(2)在點b，網點變數x_1=1改為1值變數並置於堆疊上(改令$ID(1)$=1，NS=NS+1=1，$IS(NS)$=1)，其餘$ID(*)$值不變。自由變數與可行網路有3，按圖二選變數x_2=1之網路至點c。
(3)在點c，網點變數x_2=1改為1值變數並置於堆疊上(改令$ID(2)$=1，

NS=NS+1=2，$IS(NS)$=2)，其餘$ID(*)$值不變。自由變數與可行網路有2，按圖二選變數x_4=1之網路至點d。

(4)在點d，網點變數x_4=1改為1值變數並置於堆疊上(改令$ID(4)$=1，NS=NS+1=3，$IS(NS)$=4)，其餘$ID(*)$值不變。現僅剩變數x_3=1之唯一可行網路至點e。

［表一］ 圖二評估路線中評估點之變數狀態與堆疊內容

node	x1 x2 x3 x4	ID(*)	IS(*)	IDS(*)
a	0 0 0 0	0 0 0 0		
b	1 0 0 0	1 0 0 0	1	1
c	1 1 0 0	1 1 0 0	1 2	1 2
d	1 1 0 1	1 1 0 1	1 2 4	1 2 4
e	1 1 1 1	1 1 1 1	1 2 4 3	1 2 4 3
f	1 1 0 1	1 1 -1 1	1 2 4 3	1 2 4 -3
g	1 1 0 0	1 1 0 -1	1 2 4	1 2 -4
h	1 1 1 0	1 1 1 -1	1 2 4 3	1 2 -4 3
i	1 1 0 0	1 1 -1 -1	1 2 4 3	1 2 -4 -3
j	1 0 0 0	1 -1 0 0	1 2	1 -2
k	1 0 1 0	1 -1 1 0	1 2 3	1 -2 3
l	1 0 1 1	1 -1 1 1	1 2 3 4	1 -2 3 4
m	1 0 1 0	1 -1 1 -1	1 2 3 4	1 -2 3 -4
n	1 0 0 0	1 -1 -1 0	1 2 3	1 -2 -3
o	1 0 0 1	1 -1 -1 1	1 2 3 4	1 -2 -3 4
p	1 0 0 0	1 -1 -1 -1	1 2 3 4	1 -2 -3 -4
q	0 0 0 0	-1 0 0 0	1	-1

(5)在點e，網點變數x_3=1改為1值變數並置於堆疊上(改令$ID(3)$=1，NS=NS+1=4，$IS(NS)$=3)，其餘$ID(*)$值不變。因已無向下之可行網路，必須順原路上行回上層至點f。

(6)在點f，回程之上行網路變數x_3須改為0值變數但仍在堆疊上(改令$ID(3)$=-1，NS=4不變)，其餘$ID(*)$值不變。因亦無向下之可行網路，必須再順原路上行回上層至點g。

(7)在點g，除回程上行網路變數x_4須改為0值變數(改令$ID(3)$=-1)外，在點f被設為0值變數之x_3，應為點g之自由變數，必須將其$ID(3)$改為0，並自堆疊取出。

事實上在向下無可行網路而必須回上層時，必須從堆疊取得回上層之原路線，例如在由點e回點f之前，令KV=$IS(NS)$=3，即可取得回程之x_3網路。但是在由點f回點g之前，令KV=$IS(NS)$=3，所取得之x_3網路並非回程網路，而是點f之下行網路(由$ID(3)$=-1得知)，此時正好可以利用此KV=3將對應變數x_3由0值變數恢復為自由變數並自

堆疊取出(令$ID(KV)$=0 ， NS=NS−1)。因尚未取得回程網路，繼續令KV=$IS(NS)$=4 ，現由$ID(4)$=1可知其為回程網路。按此KV=4將對應變數x_4由1值變數改為0值變數仍在堆疊上(令$ID(KV)$=−1 ， NS=3不變)。即可做點g之評估。點g評估後只能走變數x_4之唯一下行網路至點h。

(8)在點h ，網點變數x_4=1改為1值變數並置於堆疊上(改令$ID(4)$=1 ， NS=NS+1=4 ， $IS(NS)$=4)，其餘$ID(*)$值不變。因無下行網路，須順原路上行回上層至點i (令KV=$IS(NS)$=3 ，由$ID(3)$=1知其為回程網路)。

(9)在點i ，變數x_3=0改為0值變數仍在堆疊上(改令$ID(3)$=−1 ， NS=4不變)，其餘$ID(*)$值不變。因已無向下之可行網路，必須順原路上行回上層至點j ：令KV=$IS(NS)$=3 ，由$ID(3)$=−1知其非回程網路，將其由0值變數恢復為自由變數(令$ID(3)$=0 ， NS=NS−1=3) ，再令KV=$IS(NS)$=4 ，由$ID(4)$=−1知其亦非回程網路，將其由0值變數恢復為自由變數(令$ID(4)$=0 ， NS=NS−1=2) ，續令KV=$IS(NS)$=2 ，由$ID(2)$=1知其為回程網路。回點j以後即可照類似做法利用堆疊之存取，順利走遍所有評估點至e'為止。表中僅列出前半數評估點之數據，其餘者可自行練習補上。

如果令堆疊之值為負者(如表中最後一欄$IDS(*)$值)表示其為0值變數，即可不必用狀態指標$ID(*)$。但評估時較常用的資料為自由變數之係數，除非將自由變數號碼放在堆疊後之剩餘位置，否則評估分析時會很不方便。此種方式亦使堆疊運算較複雜，故本章提供之程式並未採用。

16.7 巴拉斯法之評估方法

巴拉斯法係順著如圖二所示之評估路線(詳前二節)， 對所經過之評估點求其目標函數值及束制值，以(顯式)評估是否為最佳解。同時對該評估點視為自由變數之新問題，以(隱式)評估其是否為四種特殊問題之一，如是即可跳過部分評估點。如果沒有可跳過之評估點，則評估路線會經過全部網點，故可找到最佳解或確定無合理解。

以下為巴拉斯法之評估步驟(詳副程式$FINDKV$)：

(1)若$Z \geq Zopt$ ，則令KV=0而結束評估。為第一或第四種特殊問題。

(2)對所有$ID(J)$=0之自由變數：若Z+$C(J) \geq Zopt$成立，則令KV=−J而結束評估。為第三種特殊問題。

(3)對所有$G(I)$<0之不符束制：若$G(I)$加上所有正值之自由變數係數$A(I,J)$之和仍小於0 ，則令KV=0而結束評估。為第二種特殊問題。

(4)對所有$ID(J)$=0之自由變數：計算各束制之$G(I)+A(I,J)$，將負值者相加，比較得該值為最大者之變數J，但若該值等於0，則選$C(J)$為最小者之變數J(因為此時表示下一評估點為合理解，故選最小$C(J)$以使其目標函數值為最小)，令KV=J而結束評估。

上述評估之結果如為KV=0，即表示沒有可下行網路，或已經本評估點之(隱式)評估確定經其下行網路之評估點無最佳解，因此可以跳過而不必做(顯式)評估。如評估結果為KV=$-J$，即表示自由變數J可改為0值變數。如評估結果為KV=J，即表示由自由變數J之下行網路可達下一評估點。根據KV所做運算如下(亦詳副程式$ZERONE$及上節說明)：

(1)若KV=J：將自由變數J改為1值變數，置J於堆疊，到下層評估點。即令$ID(J)$=1，NS=NS+1，$IS(NS)$=J。計算下層評估點之Z與$G(I)$。

(2)若KV=$-J$：將自由變數J改為0值變數，置J於堆疊，留在同網點。即令$ID(J)$=-1，NS=NS+1，$IS(NS)$=J。

(3)若KV=0：先自堆疊取得本評估點之0值變數將其改為自由變數，最後亦自堆疊取得回上層評估點之上行網路，將上行網路變數改為0值變數，到上層評估點。即令KV=$IS(NS)$，若$ID(KV)$=-1，亦即為下行網路之0值變數，由$ID(KV)$=0，NS=$NS-1$，改為自由變數，並從堆疊去除，重複此過程；若$ID(KV)$=1，亦即為上行網路之1值變數，由$ID(KV)$=-1，改為0值變數。計算上層評估點之Z與$G(I)$。

步驟(1)(3)須計算評估點之Z與$G(I)$時均由$COMPGZ$計算。因為由一評估點至另一評估點僅其中一個變數值改變，故計算Z或$G(I)$僅需加或減該變數之係數即可。如為較佳解則保留變數狀態$ID(*)$於$IM(*)$。

注意下行網路變數之決定並無絕對的最佳準則，甚至以隨機亂數做決定也可以求解。另外在求得合理解後，可以外加一限制條件為：目標函數小於已求得之最小目標函數值。此條件可再併入原束制條件之線性組合。Geoffrion曾建議以線性規劃之單一法計算該組合係數以求得最有力之限制式。一般而言，增列之限制條件提供較嚴密限制而可能加速求解，或使不合理解更容易確定。但其所增之計算量是否能較所減少之巴拉斯法之計算量為少，亦應為考慮之因素。本章所附之程式並未採用。

16.8 範例

前面數節已分析過巴拉斯法之各項細節，以下再以一範例做說明。

$$\begin{aligned} z &= 3x_1 + 5x_2 + 2x_3 + x_4 \\ &-x_1 + 5x_2 + 4x_3 - 2x_4 - 1 \geq 0 \\ &4x_1 + 4x_2 - 2x_3 + 3x_4 - 5 \geq 0 \\ &4x_1 - x_2 + 2x_3 + 4x_4 - 7 \geq 0 \end{aligned}$$

(1)各表中左半部分為原式以各定值變數之值代入後之新式。故僅剩自由變數之係數。d等於1值變數之c_j之和。g_i等於b_i與1值變數之a_{ij}之和。
(2)各表中右半部分為對評估點之新問題之隱式評估分析：
(2a)在d值右邊之值為目前已知之最佳值即z_{opt}，$*$表尚無合理解可視為無窮大值。若$d \geq z_{opt}$，即可令KV=0而結束評估，為第一或第四種特殊問題。點(d)與(l)即屬之，二評估點均為合理解。合理解之評估點為第一種特殊問題；非合理解之評估點為第四種特殊問題。
(2b)在點(g)中，表右$x3$下方之2+8為計算c_3+d之值，因其大於z_{opt}=9，故知$x3$必須為0，因此令KV=-3而結束評估，為第三種特殊問題。其他評估點(n)與(q)亦有類似計算，但均為c_j+$d < z_{opt}$之情形，故未結束評估，且評估點(j)與(k)之上述計算值已被後述之(2d)項之計算值所取代。
(2c)g_{max}值等於g值加上所有自由變數之正值係數a_{ij}。若有任一$g_i < 0$，即可令KV=0而結束評估，為第二種特殊問題。點(m)(n)與(q)即屬之。點(m)有二個g_{max}小於0，其他評估點各有一個。
(2d)右表中自由變數下方之值等於g加左表中對應之自由變數下方之係數a_{ij}。例如評估點(a)中：表右$x1$下方之$(-2,-1,-3)$等於$(-1,-5,-7)$加表左$x1$下方之$(-1,4,4)$。將這些值中之所有負值相加即得其上之數值。如$x1$下之-7為$(-2,-1,-3)$之和；$x2$下之-9為$(4,-1,-8)$中之負值$(-1,-8)$之和；由此可得所有自由變數下方之值為$(-7,-9,-12,-8)$，其中以變數$x1$之-7為最大，因此令KV=1而結束評估。點(a)(b)(c)(j)與(k)均屬之。所選之變數KV為在該變數之正下方有V之記號者。

上述KV為對評估點隱式評估之唯一結果，經評估過之評估點均有其值，如各表右下方所示。KV值如果為0，即可跳過$2(2^{N_f}-1)$個評估點之顯式評估，N_f為評估點之自由變數之數目。例如在評估點(d)與(n)之KV=0，其N_f=1，故各跳過2個評估點，即下方評估點(e)與(o)及回程評估點(f)與(p)之評估。評估點(q)之KV=0，其N_f=3，故一次即跳過14次評估。評估點(i)(l)與(m)之KV=0，其N_f=0，故已無評估點可跳過。因此如堆疊已滿則無自由變數可選，即N_f=0，則必然得KV=0。

KV 如為負值，通常可跳過 $2^{N_f}-1$ 個評估點之顯式評估，評估點(g)之 $N_f=1$ ，故僅跳過 1 個評估點，即下方評估點(h)之評估。

［表二］ 範例各評估點之評估分析

```
      x1  x2  x3  x4    d/g   g_max  x1  x2  x3  x4
    +-----------------+-----++-----+--V--------------+
(a) |  3   5   2   1  |  0  <   *  | -7  -9 -12  -8  |  X( 0  0  0  0)
    +-----------------+-----++-----+-----------------+
    | -1   5   4  -2  | -1  ||  8  | -2   4   3  -3  | ID( 0  0  0  0)
    |  4   4  -2   3  | -5  ||  6  | -1  -1  -7  -2  |
    |  4  -1   2   4  | -7  ||  0  | -3  -8  -5  -3  | IS()
    +-----------------+-----++-----+-----------------+
                                                      KV= 1

      1  x2  x3  x4    d/g   g_max  x1  x2  x3  x4
    +-----------------+-----++-----+------V----------+
(b) |      5   2   1  |  3  <   *  |     -4  -4  -4  |  X( 1  0  0  0)
    +-----------------+-----++-----+-----------------+
    |      5   4  -2  | -2  ||  7  |      3   2  -4  | ID( 1  0  0  0)
    |      4  -2   3  | -1  ||  6  |      3  -3   2  |
    |     -1   2   4  | -3  ||  3  |     -4  -1   1  | IS( 1)
    +-----------------+-----++-----+-----------------+
                                                      KV= 2

      1   1  x3  x4    d/g   g_max  x1  x2  x3  x4
    +-----------------+-----++-----+-------------V--+
(c) |          2   1  |  8  <   *  |         -2   0  |  X( 1  1  0  0)
    +-----------------+-----++-----+-----------------+
    |          4  -2  |  3  ||     |          7   1  | ID( 1  1  0  0)
    |         -2   3  |  3  ||     |          1   6  |
    |          2   4  | -4  ||  2  |         -2   0  | IS( 1  2)
    +-----------------+-----++-----+-----------------+
                                                      KV= 4

      1   1  x3   1    d/g   g_max  x1  x2  x3  x4
    +-----------------+-----++-----+-----------------+
(d) |          2      |  9  >=  9  |                 |  X( 1  1  0  1)
    +-----------------+-----++-----+-----------------+
    |          4      |  1  ||     |                 | ID( 1  1  0  1)
    |         -2      |  6  ||     |                 |
    |          2      |  0  ||     |                 | IS( 1  2  4)
    +-----------------+-----++-----+-----------------+
                                                      KV= 0

      1   1  x3   0    d/g   g_max  x1  x2  x3  x4
    +-----------------+-----++-----+---------V-------+
(g) |          2      |  8  <   9  <=       2+8       |  X( 1  1  0  0)
    +-----------------+-----++-----+-----------------+
    |          4      |  3  ||     |                 | ID( 1  1  0 -1)
    |         -2      |  3  ||     |                 |
    |          2      | -4  ||     |                 | IS( 1  2  4)
    +-----------------+-----++-----+-----------------+
                                                      KV=-3

      1   1   0   0    d/g   g_max  x1  x2  x3  x4
    +-----------------+-----++-----+-----------------+
(i) |                 |  8  <   9  |                 |  X( 1  1  0  0)
    +-----------------+-----++-----+-----------------+
    |                 |  3  ||     |                 | ID( 1  1 -1 -1)
    |                 |  3  ||     |                 |
    |                 | -4  ||     |                 | IS( 1  2  4  3)
    +-----------------+-----++-----+-----------------+
                                                      KV= 0
```

```
      1   0  x3  x4    d/g   g_max  x1  x2  x3  x4
     +-----------------+-----++-----+----------V------+
(j)  |          2   1  |  3  <   9  |         -4  -4  |  X( 1  0  0  0)
     +-----------------+-----++-----+-----------------+
     |          4  -2  | -2  ||  2  |          2  -4  | ID( 1 -1  0  0)
     |         -2   3  | -1  ||  2  |         -3   2  |
     |          2   4  | -3  ||  3  |         -1   1  | IS( 1  2)
     +-----------------+-----++-----+-----------------+
                                                       KV= 3
      1   0   1  x4    d/g   g_max  x1  x2  x3  x4
     +-----------------+-----++-----+-------------V--+
(k)  |              1  |  5  <   9  |              0  |  X( 1  0  1  0)
     +-----------------+-----++-----+-----------------+
     |             -2  |  2  ||     |              0  | ID( 1 -1  1  0)
     |              3  | -3  ||  0  |              0  |
     |              4  | -1  ||  3  |              3  | IS( 1  2  3)
     +-----------------+-----++-----+-----------------+
                                                       KV= 4
      1   0   1   1    d/g   g_max  x1  x2  x3  x4
     +-----------------+-----++-----+-----------------+
(l)  |                 |  6  >=  6  |                 |  X( 1  0  1  1)
     +-----------------+-----++-----+-----------------+
     |                 |  0  ||     |                 | ID( 1 -1  1  1)
     |                 |  0  ||     |                 |
     |                 |  3  ||     |                 | IS( 1  2  3  4)
     +-----------------+-----++-----+-----------------+
                                                       KV= 0
      1   0   1   0    d/g   g_max  x1  x2  x3  x4
     +-----------------+-----++-----+-----------------+
(m)  |                 |  5  <   6  |                 |  X( 1  0  1  0)
     +-----------------+-----++-----+-----------------+
     |                 |  2  ||     |                 | ID( 1 -1  1 -1)
     |                 | -3  || -3  < 0               |
     |                 | -1  || -1  < 0               | IS( 1  2  3  4)
     +-----------------+-----++-----+-----------------+
                                                       KV= 0
      1   0   0  x4    d/g   g_max  x1  x2  x3  x4
     +-----------------+-----++-----+-----------------+
(n)  |              1  |  3  <   6  >            1+3  |  X( 1  0  0  0)
     +-----------------+-----++-----+-----------------+
     |             -2  | -2  || -2  < 0               | ID( 1 -1 -1  0)
     |              3  | -1  ||  2  |                 |
     |              4  | -3  ||  1  |                 | IS( 1  2  3)
     +-----------------+-----++-----+-----------------+
                                                       KV= 0
      0  x2  x3  x4    d/g   g_max  x1  x2  x3  x4
     +-----------------+-----++-----+-----------------+
(q)  |      5   2   1  |  0  <   6  >    5+0 2+0 1+0  |  X( 0  0  0  0)
     +-----------------+-----++-----+-----------------+
     |      5   4  -2  | -1  ||  8  |                 | ID(-1  0  0  0)
     |      4  -2   3  | -5  ||  2  |                 |
     |     -1   2   4  | -7  || -1  < 0               | IS( 1)
     +-----------------+-----++-----+-----------------+
                                                       KV= 0
```

16.9 整數線性或非線性規劃問題

本節說明將整數線性或非線性規劃問題轉化為零壹線性或非線性規劃問題之技巧。將整數變數x用2進位表示，即令$x = \sum_{j=0}^{k} 2^j y_j$，將變數$x$改用$k+1$個新變數$y_o, y_1, \ldots, y_k$代替，因$y_j$僅能為0或1，將$x = \sum_{j=0}^{k} 2^j y_j$代入目標函數式與束制式，這些函式如為線性，則所得之各式亦為新變數y_j之線性式；如為非線性，則所得之各式為新變數y_j之非線性式。故原整數線性規劃問題即可變成零壹線性規劃問題；原整數非線性規劃問題 則可變成零壹非線性規劃問題。零壹非線性規劃問題再用下節技巧即可變成零壹線性規劃問題。

16.10 零壹非線性規劃問題

本節說明將零壹非線性規劃問題轉化為零壹線性規劃問題之技巧。因為當$x = 0$或1時，若$n > 0$，則$x^n = x$。故目標函數式或束制式中各變數之次方均可改為一次。例如$2x_1^2x_2^5 + 5x_1^2 + 3x_2^3 - x_2^2x_3x_4^4$可改為$2x_1x_2 + 5x_1 + 3x_2 - x_2x_3x_4$。但二個或二個以上變數的乘積項，如$x_1x_2$，就不是那麼簡單了。先考慮二個變數之乘積，如$x_1x_2$。其轉換方式為增加一新變數$y$，將乘積項用$y$取代，$y$之值亦僅限於0與1，並增加下列二個線性束制式：

$$x_1 + x_2 - y \leq 1 \tag{16.4}$$

$$x_1 + x_2 - 2y \geq 0 \tag{16.5}$$

由上二式之限制可使x_1, x_2與y符合下表之關係。

x_1	x_2	x_1x_2	y	有效用限制式	
0	0	0	0		$-2y \geq 0$
0	1	0	0		$1-2y \geq 0$
1	0	0	0		$1-2y \geq 0$
1	1	1	1	$2-y \leq 1$	

現考慮p個變數之乘積，如$x_1x_2\cdots x_p$。其轉換方式亦為增加一新變數y，將乘積項用y取代，y之值亦僅限於0與1，並增加下列二個線性束制式：

$$\sum_{j=1}^{p} x_j - \quad y \le p-1 \tag{16.6}$$

$$\sum_{j=1}^{p} x_j - p\,y \ge 0 \tag{16.7}$$

由上二式之限制可使$x_1, x_2, \ldots, x_p$與y符合下表之關係。表中$0 \le k < p$。

$\sum x_j$	$\prod x_j$	y	有效用限制式	
0	0	0		$-p\,y \ge 0$
k	0	0		$k - p\,y \ge 0$
$p-1$	0	0		$p-1-p\,y \ge 0$
p	1	1	$p - y \le p-1$	

例如將$2x_1x_2 + 5x_1 + 3x_2 - x_2x_3x_4$中之$x_1x_2$以$y_1$取代，$x_2x_3x_4$以$y_2$取代，即可改為線性式$2y_1 + 5x_1 + 3x_2 - y_2$。並增加下列四個線性束制式：

$$x_1 + x_2 - \quad y_1 \le 1 \tag{16.8}$$

$$x_1 + x_2 - 2y_1 \ge 0 \tag{16.9}$$

$$x_2 + x_3 + x_4 - \quad y_2 \le 2 \tag{16.10}$$

$$x_2 + x_3 + x_4 - 3y_2 \ge 0 \tag{16.11}$$

16.11 有條件之束制式

本節說明在零壹線性規劃問題中加入有條件之束制式之處理方法。若在零壹線性規劃問題中之束制條件(X1)與(X2)，只有在條件(Y1)與(Y2)同時成立或條件(Z1)成立時才需要滿足。亦即：IF((Y1.AND.Y2).OR.Z1) THEN (X1 and X2 must be .TRUE.)。其處理方法為：增加限值為0或1之變數y_1, y_2, z_1，先將條件式(Y1)、(Y2)與(Z1)改成為零壹規劃問題之新束制式(YY1)、(YY2)與(ZZ1)，之後，即可將有條件的束制式(X1)與(X2)改為無條件的新束制式(XX1)與(XX2)。注意這些新束制式含有許多變數之乘積項，須再用上節之方法做轉換。

$$X1 \ : \qquad a_{11}x_1 + a_{12}x_2 + b_1 \le c_1 \tag{16.12}$$

$$X2 \ : \qquad a_{21}x_1 + a_{23}x_3 + b_2 \le c_2 \tag{16.13}$$

$$Y1 \ : \qquad a_{31}x_1 + a_{33}x_3 + b_3 \ge d_1 \tag{16.14}$$

$$Y2 \ : \qquad a_{41}x_1 + a_{43}x_3 + b_4 \ge d_2 \tag{16.15}$$

$$Z1 \ : \qquad a_{51}x_1 + a_{53}x_4 + b_5 \ge d_3 \tag{16.16}$$

上列各式中須$c_i \ge 0$，$d_i \ge \epsilon > 0$，a_{ij}與b_i可為任意值。ϵ為很小之正值，但如a_{ij}與b_i均為整數，可用$\epsilon = 1$。亦注意前二束制式為$\le$；後三條件式為$\ge$。因a_{ij}與b_i可正可負，故任意不等式均可改成上列型式。等式則可改為一個$\ge$及一個$\le$之不等式。

$$XX1 \ : \qquad [1-(1-y_1y_2)(1-z_1)]\,(a_{11}x_1 + a_{12}x_2 + b_1) \le c_1 \tag{16.17}$$

$$XX2 \ : \qquad [1-(1-y_1y_2)(1-z_1)]\,(a_{21}x_1 + a_{23}x_3 + b_2) \le c_2 \tag{16.18}$$

$$YY1 \ : \qquad (1-y_1)\,(a_{31}x_1 + a_{33}x_3 + b_3) \le d_1 - \epsilon \tag{16.19}$$

$$YY2 \ : \qquad (1-y_2)\,(a_{41}x_1 + a_{43}x_3 + b_4) \le d_2 - \epsilon \tag{16.20}$$

$$ZZ1 \ : \qquad (1-z_1)\,(a_{51}x_1 + a_{53}x_4 + b_5) \le d_3 - \epsilon \tag{16.21}$$

上述做法之正確性說明如下(注意$\le d_i - \epsilon$係取代$< d_i$之條件)：

(1)若條件(Y1)成立，則由新束制式(YY1)之條件必可求得$y_1 = 1$；同樣若(Y2)成立，則得$y_2 = 1$；若(Z1)成立，則得$z_1 = 1$。

(2)考慮新束制式(XX1)中之$[1-(1-y_1y_2)(1-z_1)]$：
(2a)該值若等於0，則(XX1)一定會滿足，與束制式(X1)是否成立無關，相當於(X1)不存在而無束制作用；
(2b)該值若等於1，則(XX1)即等於(X1)，亦即(X1)存在而有束制作用。

(3)若欲使$[1-(1-y_1y_2)(1-z_1)] = 1$，必須$y_1y_2 = 1$或$z_1 = 1$。而欲使$y_1y_2 = 1$，必須$y_1$與$y_2$同時等於1。

由此可知：必須(Y1)與(Y2)同時成立得$y_1y_2 = 1$，或(Z1)成立得$z_1 = 1$，才會使$[1-(1-y_1y_2)(1-z_1)] = 1$而讓(X1)與(X2)存在。

注意用前一章之方法可將束制式(YY1)改為$(a_{31}x_1 + a_{33}x_3 + b_3) - d_1 + \epsilon - \overline{d}\,y_1 \le 0$，式中$\overline{d}$為$(a_{31}x_1 + \cdots) - d_1 + \epsilon$之可能最大值。束制式(XX1)改為$(a_{11}x_1 + a_{12}x_2 + b_1) - c_1 - \overline{d}(1-y_1y_2)(1-z_1) \le 0$，式中$\overline{d}$為$(a_{11}x_1 + \cdots) - c_1$之可能最大值。其他束制式亦可同樣更改。以此方式可減少許多非線性之乘積項。

16.12 排除部分變數之組合解之束制式

本節說明在零壹線性規劃問題中之另一種可行做法為：規定部分變數如$(x_1, x_3, x_5, x_7, x_9)$不能等於某些固定值如$(1,1,1,0,0)$，其他變數則不限其值。其處理方式為增加一如下之束制條件：

$$(x_1 + x_3 + x_5) - (x_7 + x_9) \leq 2 \tag{16.22}$$

上式中不等式右邊常數項之值等於前述固定值為1之變數數目減1。其正確性可由以下說明看出：考慮上式左邊之值：當$(x_1, x_3, x_5, x_7, x_9) = (1,1,1,0,0)$時等於$3$，該值為左邊可能之最大值，但正好比右邊$2$多$1$而不滿足該式。但若其中任一變數改為非指定之值，如$x_1$由$1$改為$0$，或$x_7$由$0$改為$1$，均會使左邊之值等於$2$而滿足該式。更多的值改變，則左邊的值更小亦滿足該式。

［表三］ 解零壹線性規劃問題之程式

```
************************************************************************
      PROGRAM ZERO1T
C**   ============================================================    **
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION A(80,90),B(80),C(90),G(80),IX(90),ID(90),IM(90),IS(90)
C**   ============================================================    **
   10 READ (*,'(2I7)') NC,NV
      READ (*,'(11F7.0)') (C(J),J=1,NV)
      DO 20 I=1,NC
   20 READ (*,'(11F7.0)') (A(I,J),J=1,NV),B(I)
C**   ------------------------------------------------------------    **
      CALL ZERONE(A,B,C,G,IX,ID,IM,IS,ZOPT,NC,NV,80)
C**   ------------------------------------------------------------    **
      WRITE(*,'(11F7.0)') (C(J),J=1,NV)
      DO 30 I=1,NC
   30 WRITE(*,'(11F7.0)') (A(I,J),J=1,NV),B(I)
      WRITE(*,'(11F7.0)') (G(I),I=1,NC)
C**   ------------------------------------------------------------    **
      WRITE(*,'('' Solution vector  :''/(11I7))') (IX(J),J=1,NV)
      WRITE(*,'('' Optimal function :  ''11F7.0)') ZOPT
      GO TO 10
      END
************************************************************************
      SUBROUTINE ZERONE(A,B,C,G,IX,ID,IM,IS,ZOPT,NC,NV,NRA)
C**   ============================================================    **
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION A(NRA,NV),B(NC),C(NV),G(NC),IX(NV),ID(NV),IM(NV),IS(NV)
C**   ============================================================    **
C**   Minimize Z = C(J)*IX(J)                                         **
C**   Subject to : A(I,J)*IX(J) + B(I) >= 0                           **
C**              : IX(J) = 0 or 1                                     **
C**   ============================================================    **
C*O   G(I)  = A(I,J)*IX(J) + B(I)                                     **
```

```
C*O   IX(J) = 0 or 1 : The solution vector                              **
C*W         = 1 : Indicate the input C(J) is negative                   **
C*W   ID(J) = 0 : Variable J is a free variable                         **
C*W         = 1 : Variable J is fixed at 1                              **
C*W         =-1 : Variable J is fixed at 0                              **
C*W   IM(J) = ID(J) corresponding to ZOPT                               **
C*W   IS(*) = Fixed variables stack                                     **
C*W   NS    = Number of fixed variables in the stack                    **
C*W   Z     = Objective function under consideration                    **
C*O   ZOPT  = The best objective function found so far                  **
C*I   NC    = Number of constraints                                     **
C*I   NV    = Number of variables                                       **
C*I   NRA   = Row dimension of matrix A(NRA,NV)                         **
C*W   KV    = The last fixed variable to be put or get from stack       **
C**   ============================================================     **
C**   +----------------------------------------------------------+     **
C**   | Initialize : Set ID(*)=0 : i.e. all variables are free   |     **
C**   |              Change variable X(J) to 1-X(J) if C(J) < 0  |     **
C**   |                 and set IX(J)=1 to indicate this change  |     **
C**   |              Set NS=0 : i.e. no fixed variable in stack  |     **
C**   |              Set G(I)=B(I) & Z=0 and ZOPT=0 or infinity  |     **
C**   +----------------------------------------------------------+     **
      DO 20 J=1,NV
      IM(J)=0
      ID(J)=0
      IX(J)=0
      IF(C(J).LT.0.0) THEN
        IX(J)=1
        C(J)=-C(J)
        DO 15 I=1,NC
        B(I)=B(I)+A(I,J)
        A(I,J)=-A(I,J)
   15   CONTINUE
      ENDIF
   20 CONTINUE
C**   ----------------------------------------------------------     **
      NS=0
      Z=0.0
      ZOPT=0.0
      DO 30 I=1,NC
      G(I)=B(I)
      IF(G(I).LT.0.0) ZOPT=1.0D30
   30 CONTINUE
C**   +----------------------------------------------------------+     **
C**   | Find variable KV (if any) to be fixed at 1 or 0          |     **
C**   +----------------------------------------------------------+     **
   50 CALL FINDKV(A,C,G,ID,NC,NV,NRA,Z,ZOPT,KV)
C**   +----------------------------------------------------------+     **
C**   | IF KV > 0 : Go down one level & Fix variable KV at 1     |     **
C**   +----------------------------------------------------------+     **
      IF(KV.GT.0) THEN
        NS=NS+1
        IS(NS)=KV
        ID(KV)=1
        WRITE(*,'('' KV  : '',2I7)') KV,ID(KV)
C**   +----------------------------------------------------------+     **
C**   | If KV < 0 : Stay at this level & Fix variable KV at 0    |     **
C**   +----------------------------------------------------------+     **
      ELSE IF(KV.LT.0) THEN
        KV=-KV
```

```
        NS=NS+1
        IS(NS)=KV
        ID(KV)=-1
        WRITE(*,'('' KV  : '',2I7)') KV,ID(KV)
        GO TO 50
C**   +--------------------------------------------------------+    **
C**   | If KV = 0 : Free variables in stack & Go back one level  |    **
C**   +--------------------------------------------------------+    **
      ELSE
C**     +------------------------------------------------------+    **
C**     | Return if no fixed variable in stack                 |    **
C**     +------------------------------------------------------+    **
   70   IF(NS.EQ.0) GO TO 100
C**     +------------------------------------------------------+    **
C**     | Get from stack : variable KV to be free or fixed at 0  |    **
C**     +------------------------------------------------------+    **
        KV=IS(NS)
        WRITE(*,'('' KV  : '',2I7)') -KV,ID(KV)
C**     +------------------------------------------------------+    **
C**     | If variable KV fixed at 0 : Free variable KV         |    **
C**     +------------------------------------------------------+    **
        IF(ID(KV).LT.0) THEN
          NS=NS-1
          ID(KV)=0
          GO TO 70
        ENDIF
C**     +------------------------------------------------------+    **
C**     | Else : Go back one level & Fix variable KV at 0      |    **
C**     +------------------------------------------------------+    **
        ID(KV)=-1
        WRITE(*,'('' KV  : '',2I7)') KV,ID(KV)
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Update G(I), Z & ZOPT : base on status of fixed var KV   |    **
C**   +--------------------------------------------------------+    **
      CALL COMPGZ(A,C,G,ID,NC,NV,NRA,Z,ZOPT,IM,KV)
      GO TO 50
C**   +--------------------------------------------------------+    **
C**   | Return : If no fixed variable in stack                 |    **
C**   +--------------------------------------------------------+    **
  100 IF(ZOPT.LT.1.0D30) ZOPT=0.0
C**   +--------------------------------------------------------+    **
C**   | Compute G(NC) : base on IM(NV)                         |    **
C**   +--------------------------------------------------------+    **
      DO 110 I=1,NC
      G(I)=B(I)
  110 CONTINUE
      DO 150 J=1,NV
      IF(IM(J).GT.0) THEN
        DO 120 I=1,NC
        G(I)=G(I)+A(I,J)
  120   CONTINUE
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Reset A,B,C to original input values : base on IX(NV)    |    **
C**   +--------------------------------------------------------+    **
      IF(IX(J).GT.0) THEN
        C(J)=-C(J)
        DO 130 I=1,NC
        B(I)=B(I)+A(I,J)
```

```
        A(I,J)=-A(I,J)
  130   CONTINUE
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Compute IX(NV) & ZOPT : base on IM(NV) & IX(NV)         |    **
C**   +--------------------------------------------------------+    **
      IF(IM(J).GT.0) IX(J)=1-IX(J)
      IF(IX(J).GT.0) ZOPT=ZOPT+C(J)
  150 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE FINDKV(A,C,G,ID,NC,NV,NRA,Z,ZOPT,KV)
C**   ============================================================    **
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION A(NRA,NV),G(NC),C(NV),ID(NV)
C**   ============================================================    **
C**   Input : A,C,G,ID,NC,NV,NRA,Z,ZOPT;        Output : KV          **
C**   ------------------------------------------------------------    **
C*O   (1) KV = 0 : If  Z >= Zopt (true for feasible solution)        **
C*O   (4)          or  Z >= Zopt (no more improvement)               **
C*O   (2)          or  No feasible solutions at all nodes            **
C*O   (u)          or  No free variable (no ID(*) = 0)               **
C*O   (3) KV < 0 : The variable to be fixed at 0 in this level       **
C*O   (d) KV > 0 : The variable to be fixed at 1 in next level       **
C**   ============================================================    **
      KV=0
C**   +--------------------------------------------------------+    **
C**   | Return KV=0 by case (1) or (4) : Z >= Zopt              |    **
C**   +--------------------------------------------------------+    **
      IF(Z.GE.ZOPT) RETURN
C**   +--------------------------------------------------------+    **
C**   | Return KV=-J by case (3) : Any Z+C(J) >= Zopt           |    **
C**   +--------------------------------------------------------+    **
      IF(ZOPT.LT.1.0D30) THEN
        DO 20 J=1,NV
        IF(ID(J).NE.0) GO TO 20
        IF(Z+C(J).LT.ZOPT) GO TO 20
        KV=-J
        RETURN
   20   CONTINUE
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | Return KV=0 by case (2) : Any G(I)+Sum.Pos.A(I,J) < 0   |    **
C**   +--------------------------------------------------------+    **
      DO 40 I=1,NC
      GI=G(I)
      IF(GI.LT.0.0) THEN
        DO 30 J=1,NV
        IF(ID(J).EQ.0.AND.A(I,J).GT.0.0) GI=GI+A(I,J)
   30   CONTINUE
        IF(GI.LT.0.0) RETURN
      ENDIF
   40 CONTINUE
C**   +--------------------------------------------------------+    **
C**   | Compute Infeasiblity value = GJ = Sum.Neg.(G(I)+A(I,J))  |    **
C**   +--------------------------------------------------------+    **
      GJMAX=-1.0D30
      DO 60 J=1,NV
      IF(ID(J).NE.0) GO TO 60
```

```
      GJ=0.0
      DO 50 I=1,NC
      GJ=GJ+AMIN1(G(I)+A(I,J),0.0)
   50 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Select KV by max.GJ : If GJ<0 : next node is infeasible |    **
C**   +-------------------------------------------------------+    **
      IF(GJ.GT.GJMAX) THEN
        GJMAX=GJ
        KV=J
C**   +-------------------------------------------------------+    **
C**   | Select KV by min.C(J) : If GJ=0 : next node is feasible |    **
C**   | Next 2 lines are not reqd if C(J) is in increasing order |    **
C**   +-------------------------------------------------------+    **
      ELSE IF(GJ.EQ.0) THEN
        IF(C(J).LT.C(KV)) KV=J
      ENDIF
    60 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Return KV>0 by case (d) : KV = variable to be fixed at 1 |    **
C**   | Return KV=0 by case (u) : No free variable             |    **
C**   +-------------------------------------------------------+    **
      RETURN
      END
*************************************************************************
      SUBROUTINE COMPGZ(A,C,G,ID,NC,NV,NRA,Z,ZOPT,IM,KV)
C**   ============================================================    **
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION A(NRA,NV),G(NC),C(NV),ID(NV),IM(NV)
C**   ============================================================    **
C**   Explicit enumerate the node by computing G(NC) & Z              **
C**   ------------------------------------------------------------    **
C**   Input  : A,C,G,ID,NC,NV,NRA,Z,ZOPT,IM,KV                        **
C**   Output : G,Z,ZOPT,IM                                            **
C**   ============================================================    **
C**   +-------------------------------------------------------+    **
C**   | Update G(I) & Z and Set ISF=1 if all G(I) >= 0 ?       |    **
C**   +-------------------------------------------------------+    **
      ISF=1
      IF(ID(KV).EQ.1) THEN
        Z=Z+C(KV)
        DO 30 I=1,NC
        G(I)=G(I)+A(I,KV)
        IF(G(I).LT.0.0) ISF=0
   30   CONTINUE
      ELSE
        Z=Z-C(KV)
        DO 40 I=1,NC
        G(I)=G(I)-A(I,KV)
        IF(G(I).LT.0.0) ISF=0
   40   CONTINUE
      ENDIF
C**   +-------------------------------------------------------+    **
C**   | For feasible solution : Save best one in Zopt & IM(NV)  |    **
C**   +-------------------------------------------------------+    **
      IF(ISF.NE.0.AND.Z.LT.ZOPT) THEN
        ZOPT=Z
        DO 50 J=1,NV
        IM(J)=ID(J)
   50   CONTINUE
```

```
        WRITE(*,'('' Zopt follows : '',F7.0)')
      ENDIF
      WRITE(*,'('' ID   :'',10I7)') ID
      WRITE(*,'('' G(*) :'',10F7.0)') (G(I),I=1,NC)
      WRITE(*,'('' Z    :'',F7.0)') Z
      RETURN
      END
*************************************************************************
```

習題

1. 試將下列整數非線性規劃問題改為零壹線性規劃問題。

$$\begin{aligned} \text{最小化} \quad z &= 3x_1 + 2x_2 + 2x_1^2 - 2x_1x_2 + 3x_2^2 \\ & x_1 \leq 3 \\ & x_2 \leq 2 \\ & x_1\ , x_2 = 0, 1, 2, \ldots \end{aligned}$$

2. 試解上題所得之零壹線性規劃問題。

3. 試將所附程式改為不用$ID(*)$，並改用堆疊$IDS(*)$取代$ID(*)$，將0值變數J之狀態用$IDS(NS) = -J$表示。

參考文獻

1. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", Operations Research, Vol.13, No.4, pp.517-546, 1965.

2. Geoffrion, Arthur M., "Integer Programming by Implicit Enumeration and Balas' Method", SIAM Review, Vol.9, No.2, pp.178-190, 1967.

3. Watters, L. J., "Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems", Operations Research, Vol.15, No.6, pp.1171-1174, 1967.

4. McMillan, Claude Jr., *Mathematical Programming : An Introduction to the Design and Application of Optimal Decision Machines,* Jojn Wiley Sons, Inc., 1970.

第十七章

快速富利葉轉換法

17.1 前言

自從1965年Cooley與Tukey發表快速富利葉轉換程式以來，富利葉轉換(Fourier Transform)即更普遍地被應用於各領域，如動力學、聲光學、醫學、數值分析、訊號處理、雷達、電磁學、通訊等，不啻是富利葉轉換的一個小小革命。快速富利葉轉換係利用富利葉轉換所獨特具有之週期性，使其可以較一般標準轉換，快數十倍以上之速度求出。因此富利葉轉換遠較其他轉換，如拉卜拉斯(Laplace Transform)等，更樂於被採用。事實上，拉卜拉斯轉換亦可利用富利葉轉換求得，文內將略加討論。

一般對於暫態反應都採用拉卜拉斯轉換，因其可以直接考慮初始條件；而富利葉轉換則僅用於求解穩態反應。實際上，由富利葉轉換所得之穩態解，即為特解部分，如再加上補解使其滿足初始條件即可求得暫態反應之全解。本章將詳述其求解過程，並提供副程式$FFTTRS$，可用以求解動力系統之穩態及暫態反應。

一般理論常僅涉及週期函數之富利葉級數及非週期函數之富利葉積分；而實際數值計算者，則為離散富利葉轉換。這三者雖極相似，但亦具有頗多相異性。如不深入了解，常不能隨心所欲地加以運用。本章即試著從各種角度來探討這三者之關係，以便能正確使用這些轉換。

快速富利葉轉換之原理簡單而奧妙，本章將做一詳細說明。相信知其奧秘後必嘆為觀止。本章提供一可能是最短的副程式$FFTA$，可以計算任意個數之複數之富利葉轉換。並提供實數及共軛複數之富利葉轉換副程式$FFTR$及$FFTC$，正餘弦轉換 副程式$FFTCS$，半正餘弦轉換副程式$FFTSC$，及相關之二維轉換程式$FFT2D$及$FFT2SC$。

17.2 頻率域分析法

一般在求解動力方程式 (17.1)(已正規化)時，可利用Duhamel積分直接求解如式(17.2)。

$$\ddot{y}(t) + 2\xi\Omega\dot{y}(t) + \Omega^2 y(t) = f(t) \tag{17.1}$$

$$y(t) = \frac{1}{\sqrt{1-\xi^2}\Omega}\int_0^t f(\tau)e^{-\xi\Omega(t-\tau)}\sin[\sqrt{1-\xi^2}\Omega(t-\tau)]\,d\tau \tag{17.2}$$

這樣稱做時間域 (Time domain)的分析法，因其直接在時間域做積分來求解(當然時間域分析亦可用一般之常微分方程數值解法)。

但亦可假設$f(t)$為一週期性函數 (其週期T可任意假設，但須使T包含欲求取反應之範圍)，以富利葉級數 (Fourier series)展開成：

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n\cos(\frac{2\pi n}{T}t) + \sum_{n=1}^{\infty} b_n\sin(\frac{2\pi n}{T}t) \tag{17.3}$$

式中

$$a_n = \frac{2}{T}\int_0^T f(t)\cos(\frac{2\pi n}{T}t)\,dt \tag{17.4}$$

$$b_n = \frac{2}{T}\int_0^T f(t)\sin(\frac{2\pi n}{T}t)\,dt \tag{17.5}$$

亦即將該荷重分解成：(a)含一定荷重$\frac{1}{2}a_0$(表示平均荷重)及(b)一系列頻率為$\omega_n = 2\pi n/T$，振幅為a_n及b_n的諧和荷重 (Harmonic loading)。

如此將$f(t)$分解成各諧和分量之後，即可求出結構物受各諧和分量作用之反應，再將各反應量相加，即得特解部分之總反應如下：

$$\begin{aligned} y^p(t) &= \Omega^{-2}\{\frac{1}{2}a_0 + \sum_{n=1}^{\infty}\frac{a_n[(1-\beta_n^2)\cos\omega_n t + 2\xi\beta_n\sin\omega_n t]}{(1-\beta_n^2)^2 + (2\xi\beta_n)^2} \\ &\quad + \sum_{n=1}^{\infty}\frac{b_n[(1-\beta_n^2)\sin\omega_n t - 2\xi\beta_n\cos\omega_n t]}{(1-\beta_n^2)^2 + (2\xi\beta_n)^2}\} \end{aligned} \tag{17.6}$$

式中

$$\beta_n = \omega_n/\Omega \tag{17.7}$$

$$\omega_n = n\omega_1 = \frac{2\pi n}{T} \tag{17.8}$$

上式應再加上下式之補解，即可得式(17.1)之通解。式中A, B可由初始條件$y(0)$及$\dot{y}(0)$求得。

$$y^c(t) = e^{-\xi\Omega t}(A\sin(\sqrt{1-\xi^2}\Omega t) + B\cos(\sqrt{1-\xi^2}\Omega t)) \tag{17.9}$$

將$f(t)$分解成各種頻率的諧和分量，再分析求解式(17.1)的方法稱為頻率域 (Frequency domain)分析法。

17.3 富利葉級數的複數表示法

利用Euler公式可將富利葉級數的cos和sin項改寫成複數指數形式。因為

$$\sin n\theta = \frac{1}{2i}(e^{in\theta} - e^{-in\theta}) \tag{17.10}$$

$$\cos n\theta = \frac{1}{2}(e^{in\theta} + e^{-in\theta}) \tag{17.11}$$

所以

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(\frac{2\pi n}{T}t) + \sum_{n=1}^{\infty} b_n \sin(\frac{2\pi n}{T}t) \tag{17.3}$$

$$= \sum_{n=-\infty}^{\infty} C_n e^{i\omega_n t} \tag{17.12}$$

式中

$$C_n = \frac{1}{T}\int_0^T f(t)e^{-i\omega_n t}dt \tag{17.13}$$

$$C_n = \frac{1}{2}(a_n - ib_n) \tag{17.14}$$

$$C_{-n} = \frac{1}{2}(a_n + ib_n) \tag{17.15}$$

既然$f(t)$可表示成式(17.12)之複數指數形式，因此可視$e^{i\omega_n t}$為一單位作用函數，代入運動方程式(17.1)中可得

$$y(t) = H_n e^{i\omega_n t} \tag{17.16}$$

式中

$$H_n = \frac{1}{(\Omega^2 - \omega_n^2) + 2\xi\Omega\omega_n i} \tag{17.17}$$

式(17.1)系統受式(17.12)之$f(t)$作用之特解部分之反應可用疊加原理得

$$y^p(t) = \sum_{n=-\infty}^{\infty} H_n C_n e^{i\omega_n t} \tag{17.18}$$

注意式(17.6)中，受一正弦(或餘弦)分量作用之反應，包含正弦及餘弦兩種分量；而由式(17.18)知，受一複數指數形式之分量$e^{i\omega_n t}$作用之反應，則僅包含一個相同指數形式之分量，且其反應係數H_n較為簡潔，故應用上較為方便。

但式(17.18)中之係數C_n及H_n均為複數，應用在實數領域較為奇怪：因為作用函數$f(t)$如為實數，則其反應$y(t)$必然亦為實數。對於此點，茲簡單說明如下：

(1)先注意在複數指數形式之式(17.13)及式(17.18)中，n值包含負值。而當$f(t)$為實數時，C_n與C_{-n}一定互為共軛複數，故$C_n e^{i\omega_n t}$與$C_{-n}e^{-i\omega_n t}$亦互為共軛複數，因此兩者相加後必能消去虛數部分。這也就是為何式(17.13)必須包含正負n值，而式(17.4)與式(17.5)只包含正n值之原因。
(2)再注意對於式(17.1)之實係數系統(或其他實係數系統)，其反應係數H_n與H_{-n}必為共軛複數對。故$H_n C_n e^{i\omega_n t}$與$H_{-n}C_{-n}e^{-i\omega_n t}$亦必為共軛複數對，兩者相加必然會消去虛數部分，而僅得到實數之反應，因此複數用於實數問題並無抵觸現象。

17.4 富利葉級數之數值計算式－離散富利葉轉換

在富利葉級數中

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{i\omega_n t} \tag{17.12}$$

$$C_n = \frac{1}{T}\int_0^T f(t)e^{-i\omega_n t}dt \tag{17.13}$$

C_n值之計算可採用梯形面積法，將$t=0$至T間分成N等分求函數和。令

$$\Delta t = t_1 = \frac{T}{N}\,, \qquad \Delta\omega = \omega_1 = \frac{2\pi}{T}$$

及

$$t_k = kt_1 = k\Delta t, \qquad \omega_n = n\omega_1 = n\Delta\omega$$

得

$$\omega_n t_k = (\Delta\omega\,\Delta t)nk = (\frac{2\pi}{N})nk \tag{17.19}$$

則可將式(17.12)與式(17.13)寫成

$$f_k = \sum_{n=-\infty}^{\infty} C_n W^{-kn} \tag{17.20}$$

$$C_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k W^{nk}\,, \qquad n = 0, \pm1, \pm2, \ldots, \pm N/2 \tag{17.21}$$

$$= 0\,, \qquad |n| > N/2 \quad (\text{式(17.21)之誤差太大}) \tag{17.22}$$

式中

$$W = e^{-2\pi i/N} \tag{17.23}$$

比較式(17.13)與式(17.21)可發現下述特性：

(1)式(17.21)為式(17.13)之積分式以梯形面積法近似之公式，已由積分型式改為求和型式。

(2)式(17.21)算得之C_n具有週期性，其週期為N，亦即

$$C_n = C_{N+n} = C_{2N+n} = \cdots, \qquad C_{-n} = C_{N-n} = C_{2N-n} = \cdots \tag{17.24}$$

(3)式(17.13)算得之C_n並無週期性，且當n值愈大時C_n值愈小。

(4)當$f(t)$為實數時，式(17.13)與式(17.21)之C_n與C_{-n}均為共軛複數對。

根據上述特性，式(17.21)之C_n對應於式(17.13)之C_n之範圍為$n = 0, \pm 1, \pm 2, \ldots, \pm N/2$。此範圍外之$C_n$利用式(17.21)僅由$N$個$f(t)$值已無法準確地算出，必須假設為零，如式(17.22)所示。因此由式(17.20)計算f_k之加總範圍應改為$n = 0, \pm 1, \pm 2, \ldots, \pm N/2$，但由於$C_n$及$C_{-n}$之週期性，其加總範圍亦可改為$n = 0, 1, 2, \ldots, N-1$，但須注意$C_{N-n}$仍代表$C_{-n}$之值$(n \leq N/2)$，此點在計算$H_n C_n$時尤應注意，因此

$$f_k = \sum_{n=0}^{N-1} C_n W^{-kn} \tag{17.25}$$

$$C_n = \frac{1}{N} \sum_{k=0}^{N-1} f_k W^{nk}, \qquad n = 0, 1, 2, \ldots, N-1 \tag{17.26}$$

另注意式(17.26)中之f_k為$t = 0$至T間之片段連續函數，取$t = k\Delta t$各點之值，以梯形面積法求C_n之積分。而式(17.25)中之C_n為ω之不連續函數，僅當$\omega = n\Delta\omega$時有值，其餘之值為零。式(17.25)與式(17.26)一般稱為離散富利葉轉換(對)。

17.5 非週期性函數之富利葉積分

假若$f(t)$為非週期性函數，如地震力，則可將其視為週期$T = \infty$的函數來處理。如此富利葉級數須重新推導，使能適用於無限長週期的函數，其轉換表示式為一積分式，故改稱為富利葉積分。在富利葉級數中

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{i\omega_n t} \tag{17.12}$$

$$C_n = \frac{1}{T} \int_0^T f(t) e^{-i\omega_n t} dt \tag{17.13}$$

當$T \to \infty$時，令$C_n = F(\omega_n)\Delta\omega$，代入上式得

$$f(t) = \sum_{n=-\infty}^{\infty} F(\omega_n)e^{i\omega_n t}\Delta\omega \tag{17.27}$$

$$F(\omega_n)\Delta\omega = \frac{1}{T}\int_0^T f(t)e^{-i\omega_n t}dt \tag{17.28}$$

利用$\Delta\omega = 2\pi/T$並令其趨近於$d\omega$，得

$$f(t) = \int_{-\infty}^{\infty} F(\omega)e^{i\omega t}d\omega \tag{17.29}$$

$$F(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{17.30}$$

式(17.29)與式(17.30)一般稱為富利葉積分(對)。

實用上，上二式之富利葉積分須用數值積分法計算，並做下面之考慮：

(1)假設計算反應之關注範圍為$t = 0$至T。

(2)設$t = 0$至T間以外之$f(t) = 0$。如果不為零仍可直接設為零，因為$t > T$之$f(t)$不影響$t = 0$至T間之反應，而$t < 0$之$f(t)$之影響可用$t = 0$之初始條件計入。因此，式(17.30)之積分範圍變為$0 \to T$。

(3)將$t = 0$至T間分為N等分，即令$\Delta t = T/N$。

(4)式(17.30)之$F(\omega)$為連續函數，但可以只計算$\omega = n\Delta\omega$之$F(n\Delta\omega)$，令其中之$\Delta\omega = 2\pi/T$。

(5)由以上四項考慮處理後，式(17.29)與式(17.30)之富利葉積分變成

$$f_k = \frac{2\pi}{T}\sum_{n=-\infty}^{\infty} F_n W^{-kn} \tag{17.31}$$

$$F_n = \frac{T}{2\pi N}\sum_{k=0}^{N-1} f_k W^{nk}\,, \qquad n = 0, \pm1, \pm2, \ldots, \pm N/2 \tag{17.32}$$

$$= 0\,, \qquad |n| > N/2 \quad (\text{式(17.32)之誤差太大}) \tag{17.33}$$

式中

$$W = e^{-2\pi i/N} \tag{17.34}$$

(6)式(17.32)之F_n雖為週期性，但可用F_n之範圍為$n = 0, \pm1, \pm2, \ldots, \pm N/2$；$|n| > N/2$之$F_n$值以式(17.32)僅用$N$個$f_k$計算，其誤差太大，不能採用，須設為零，如式(17.33)所示。

(7)式(17.31)之加總範圍應改為$n = 0, \pm1, \pm2, \ldots, \pm N/2$。但由於$F_n$及

W^{-kn}之週期性，其加總範圍亦可改為$n=0,1,2,\ldots,N-1$。因此

$$f_k = \frac{2\pi}{T}\sum_{n=0}^{N-1} F_n W^{-kn} \tag{17.35}$$

$$F_n = \frac{T}{2\pi N}\sum_{k=0}^{N-1} f_k W^{nk}, \qquad n=0,1,2,\ldots,N-1 \tag{17.36}$$

(8) 比較式(17.25)(17.26)與式(17.35)(17.36)，如令

$$C_n = (\frac{2\pi}{T})F_n \tag{17.37}$$

則二式完全相同。因此觀念上非週期性之函數$f(t)$，亦可視為週期T之週期函數，而照週期函數處理，只要週期T包含反應分析之關注範圍。

(9) $C_n=(2\pi/T)F_n=\Delta\omega F_n$可視為將連續分布函數$F(\omega)$在$\omega_n-\Delta\omega/2\le\omega\le\omega_n+\Delta\omega/2$範圍內之值集中在$\omega_n$一點。

(10) 因所能算得之F_n之最大頻率為$\Delta\omega N/2=\pi N/T$，大於此頻率之F_n均被假設為零。故所選用之N/T應夠大或$\Delta t=T/N$應夠小。

(11) 當T或$\Delta\omega$固定時，N愈大，則C_n或F_n愈準確，能算得之頻率範圍較高。當N固定時，T愈大($\Delta\omega$愈小)，則C_n或F_n愈不準，能算得之頻率範圍較低。當$\Delta t=T/N$固定時，N與T愈大($\Delta\omega$愈小)，則能算得之頻率範圍不變。

(12) 一般地震反應分析時，$N,T,\Delta t$各值通常取如[表一]所示。其範圍約為$N=256$至4096，$T=10.24$至40.96秒，$\Delta t=0.005$至0.10秒。

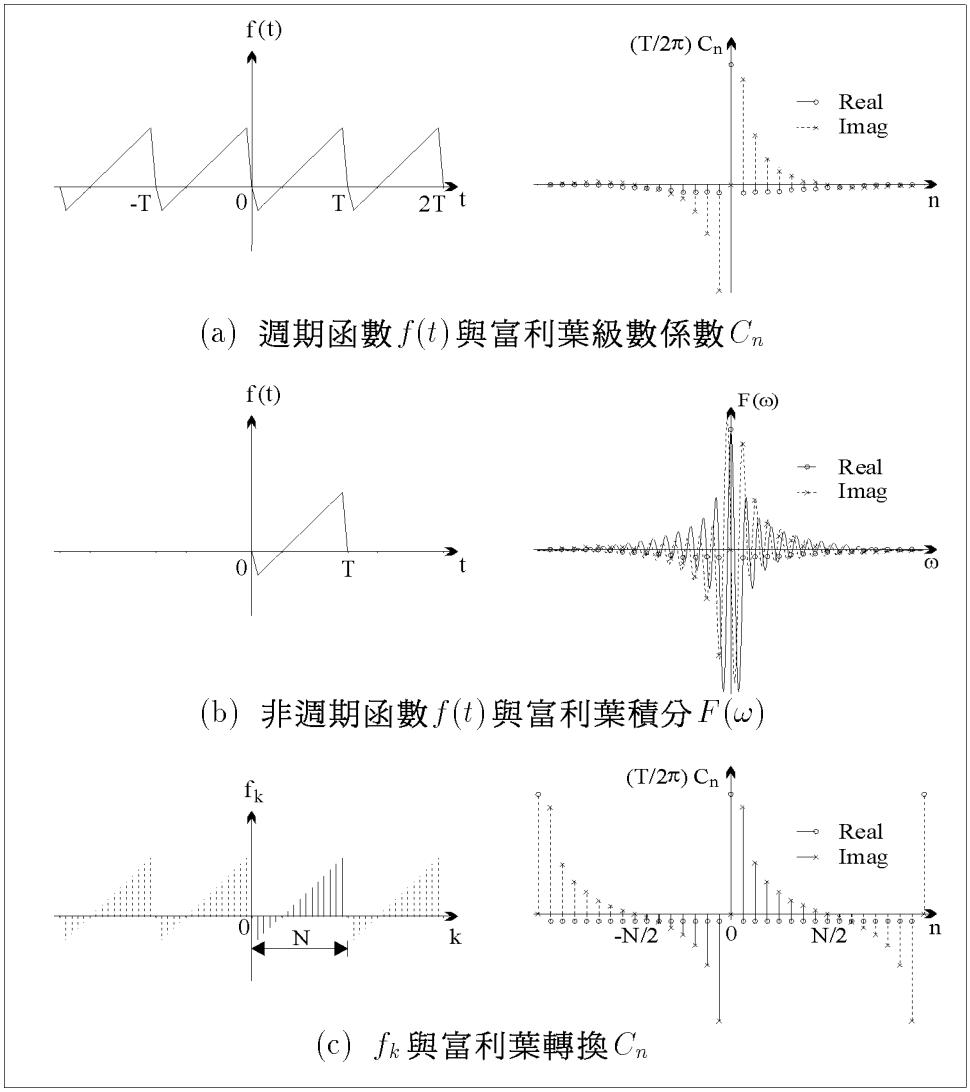
[表一] 常用之週期$T=N\Delta t$(秒)

N\Δt	0.005	0.010	0.020	0.050	0.100
256	-	-	-	12.80	25.60
512	-	-	10.24	25.60	51.20
1024	-	10.24	20.48	51.20	-
2048	10.24	20.48	40.96	-	-
4096	20.48	40.96	-	-	-

17.6 富利葉級數與富利葉積分之異同

式(17.12)之富利葉級數，與式(17.29)之富利葉積分，經離散化後分別成為式(17.25)及式(17.35)之離散富利葉轉換。其中C_n與F_n之關係以式

(17.37)對應後，式(17.25)及式(17.35)實際上是一樣的。但兩者觀念上有許多相異處，大部分已於前面述及，茲再綜合列舉其異同點如下以為參考之用(並參照圖一)：



圖一 富利葉級數、積分與轉換

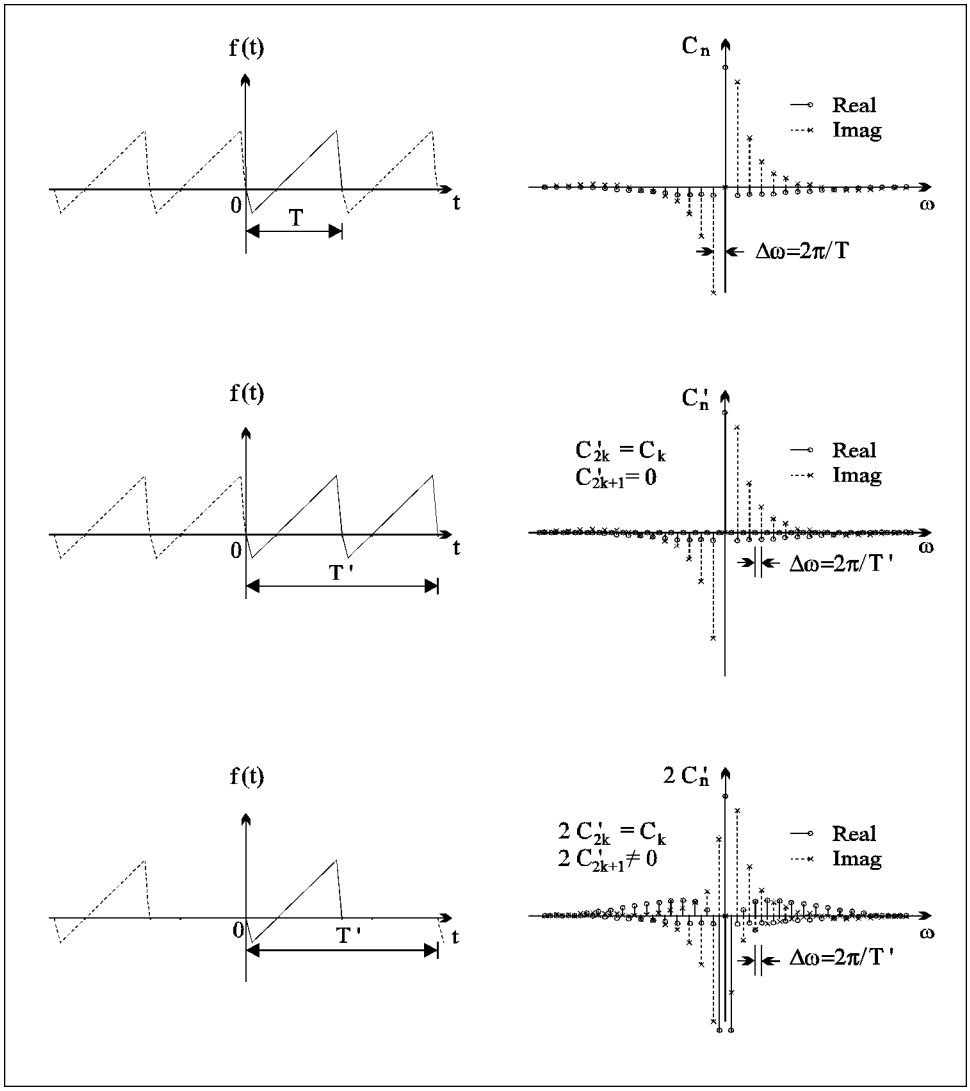
(1) C_n為離散函數 (為許多等間距之Delta function)，$F(\omega)$為連續函數。F_n為連續函數在$\omega = \omega_n$點上之函數值。C_n則可視為將$F(\omega)$在$\omega_n - \Delta\omega/2 \leq \omega \leq \omega_n + \Delta\omega/2$間之值集中在$\omega_n$點上。

(2) C_n與F_n均非週期函數，且當$n \to \infty$時，$C_n \to 0$，$F_n \to 0$。但因積分計算之近似誤差，造成C_n與F_n亦成週期函數。因此必須令$|n| > N/2$之

C_n 與 F_n 為零。

(3) 當 $f(t)$ 為實數時，C_n 與 C_{-n} 及 F_n 與 F_{-n} 均為共軛複數對。

(4) 由式 (17.12) 算得之 $f(t)$ 為週期函數，由式 (17.29) 算得之 $f(t)$ 為非週期函數；但由式 (17.25) 與式 (17.35) 算得之 f_k 均為週期函數。式 (17.12) 本來就是計算級數和，式 (17.25) 只是去掉較高頻之 C_n ，即取有限項之和，其誤差僅為少計入高頻之貢獻而已。而式 (17.29) 之 $f(t)$ 係由積分算得，並非週期函數，但改由式 (17.35) 以求和代替積分後必然使 $f(t)$ 成為週期函數，其週期為 $T = 2\pi/\ \Delta\omega$ 。當 $\Delta\omega$ 愈小時可使週期 T 變長，參見圖二。而當 $\Delta\omega \to 0$ 時，$T \to \infty$ 即可使 $f(t)$ 成為非週期函數。



圖二 週期變化對轉換係數之影響

(5)對於非週期且非有限範圍之函數之處理：一般計算反應均有一固定之關注範圍，設其為$t=0$至T_1。則可截取$t=0$至T_1間之函數，然後假設此段函數每隔T秒重複出現。即考慮其為週期T之週期函數，而可利用富利葉級數之觀念處理；亦可假設此段範圍以外($t<0$及$t>T_1$)之函數值為零，即可利用富利葉積分之觀念處理。兩者在計算式(17.25)及式(17.35)時，可採用較大之T計算。

17.7 利用FFT解動力反應之注意事項

在第一節已述及動力方程式(17.18)

$$\ddot{y}(t)+2\xi\Omega\dot{y}(t)+\Omega^2 y(t)=f(t) \tag{17.1}$$

受週期函數$f(t)$作用之特解為式(17.18)

$$y^p(t) = \sum_{n=-\infty}^{\infty} H_n C_n e^{i\omega_n t} \tag{17.18}$$

式中

$$C_n = \frac{1}{T}\int_0^T f(t)e^{-i\omega_n t}dt \tag{17.13}$$

$$H_n = \frac{1}{(\Omega^2-\omega_n^2)+2\xi\Omega\omega_n i} \tag{17.17}$$

及補解為

$$y^c(t)=e^{-\xi\Omega t}(A\sin(\sqrt{1-\xi^2}\Omega t)+B\cos(\sqrt{1-\xi^2}\Omega t)) \tag{17.9}$$

對於受非週期函數$f(t)$作用時，應改用富利葉積分，但觀念上仍可視為週期函數處理。不論用富利葉級數或富利葉積分，最後都可化為離散富利葉轉換。因此特解成為

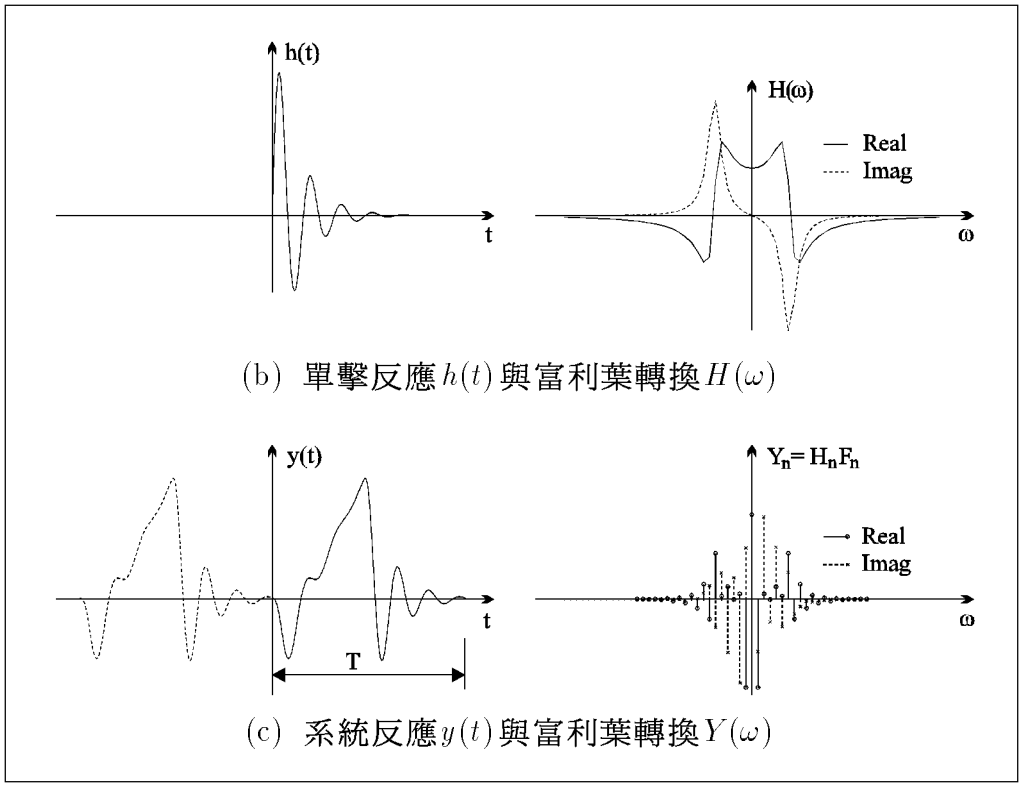
$$y_k = \sum_{n=0}^{N-1} Y_n W^{-kn} \tag{17.38}$$

式中

$$Y_n = H_n C_n \tag{17.39}$$

$$C_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k W^{nk}, \qquad n=0,1,2,\ldots,N-1 \tag{17.26}$$

$$H_n = \frac{1}{(\Omega^2-\omega_n^2)+2\xi\Omega\omega_n i} \tag{17.17}$$



圖三 結構系統反應分析

但須注意式(17.17)計算H_n之ω_n為(參見圖三)

$$\begin{aligned} \omega_n &= n\Delta\omega, && n = 0, 1, \ldots, N/2-1 \\ &= (n-N)\Delta\omega, && n = N/2, \ldots, N-1 \end{aligned} \tag{17.40}$$

對於實數之$f(t)$，除$n=0$及$n=N/2$外，C_{N-n}與C_n互為共軛複數，H_n經式(17.40)之考慮後，H_{N-n}與H_n亦互為共軛複數，因此Y_{N-n}與Y_n亦互為共軛複數。對於$n=0$之情形：C_0與H_0之虛部均為零，故Y_0之虛部亦為零。至於$n=N/2$之情形：$C_{N/2}$之虛部為零，但$H_{N/2}$之虛部並不等於零，故$Y_{N/2}$之虛部即不等於零，則y_k之虛部將不一定為零。雖然$H_{N/2}$之值一般已甚小，影響不大，但仍以能使所有之H_{N-n}與H_n互為共軛複數較符合理論要求，因此可令$Imag(H_{N/2})=0$，亦即令

$$Imag(H_{N/2}) = 0 \qquad \text{可使} \qquad Imag(Y_{N/2}) = 0 \tag{17.41}$$

17.8 暫態反應之計算

利用頻率域之分析法，除可求得穩態反應 (Steady–state response) 外，亦可加上補解利用初始條件求得暫態反應 (Transient response)：

$$y(t) = y^p(t) + y^c(t) \tag{17.42}$$

由式 (17.38) 知

$$y^p(t) = \sum_{n=0}^{N-1} Y_n W^{-kn} \tag{17.43}$$

$$\dot{y}^p(t) = \sum_{n=0}^{N-1} i\omega_n Y_n W^{-kn} \tag{17.44}$$

代 $t = 0$，可得

$$y^p(0) = \sum_{n=0}^{N-1} Y_n = \sum_{n=0}^{N-1} Y_{rn} = y_0^p \tag{17.45}$$

$$\dot{y}^p(0) = \sum_{n=0}^{N-1} i\omega_n Y_n = -\sum_{n=0}^{N-1} \omega_n Y_{in} = \dot{y}_0^p \tag{17.46}$$

以下之式 (17.47) 為式 (17.9) 之另一種型式

$$y^c(t) = \frac{e^{-\xi\Omega t}}{\sqrt{1-\xi^2}}[\frac{\dot{y}_0^c}{\Omega}\sin(\sqrt{1-\xi^2}\Omega t) + y_0^c\cos(\sqrt{1-\xi^2}\Omega t - \phi)] \tag{17.47}$$

上式微分，並令

$$\sin\phi = \xi, \qquad \cos\phi = \sqrt{1-\xi^2} \tag{17.48}$$

可得

$$\dot{y}^c(t) = \frac{\Omega e^{-\xi\Omega t}}{\sqrt{1-\xi^2}}[\frac{\dot{y}_0^c}{\Omega}\cos(\sqrt{1-\xi^2}\Omega t + \phi) - y_0^c\sin(\sqrt{1-\xi^2}\Omega t)] \tag{17.49}$$

上二式代 $t = 0$，可得

$$y^c(0) = \frac{1}{\sqrt{1-\xi^2}} y_0^c\cos(-\phi) = y_0^c \tag{17.50}$$

$$\dot{y}^c(0) = \frac{\Omega}{\sqrt{1-\xi^2}}\frac{\dot{y}_0^c}{\Omega}\cos(\phi) = \dot{y}_0^c \tag{17.51}$$

可見式(17.47)已將特解在$t=0$之初始位移y_0^c及初始速度$\dot{y}_0^c$考慮在內。若已知暫態反應$y(t)$之初始條件為$y(0)=y_0$，$\dot{y}(0)=\dot{y}_0$則由

$$y_0 = y_0^c + y_0^p \tag{17.52}$$

$$\dot{y}_0 = \dot{y}_0^c + \dot{y}_0^p \tag{17.53}$$

可得

$$y_0^c = y_0 - y_0^p \tag{17.54}$$

$$\dot{y}_0^c = \dot{y}_0 - \dot{y}_0^p \tag{17.55}$$

代入式(17.47)，因此可得

$$y^c(t) = \frac{e^{-\xi\Omega t}}{\sqrt{1-\xi^2}}\left[\frac{\dot{y}_0 - \dot{y}_0^p}{\Omega}\sin(\sqrt{1-\xi^2}\,\Omega t) + (y_0 - y_0^p)\cos(\sqrt{1-\xi^2}\,\Omega t - \phi)\right] \tag{17.56}$$

式中$y_0,\ \dot{y}_0$為已知之初始條件，$y_0^p,\ \dot{y}_0^p$為特解部分之初始值，可用式(17.45)與式(17.46)算得。

17.9 無阻尼系統之反應

對於無阻尼系統

$$H_n = \frac{1}{\Omega^2 - \omega_n^2} \tag{17.57}$$

當某一個$\bar{n}$值(設為$\bar{n} \le N/2$)正好使$\bar{n}\Delta\omega = \Omega$或$T = \bar{n}(2\pi/\Omega)$時，$H_{\bar{n}} \to \infty$。理論上，對於無阻尼系統受一與其自然週期相同週期之簡諧外力時，必然產生共振，其反應將隨時間增長而變大，且不再具有週期性。實際做反應分析時，$f(t)$之週期性純係假設，並非真有簡諧外力存在，因此也不應有共振反應發生。另由補解可知，無阻尼系統之補解之函數$\sin(\Omega t)$正好與特解$n=\bar{n}$之函數$\sin(\bar{n}\Delta\omega t)$相同。因此實際運算時，可直接令$H_{\bar{n}}=0$，最後再考慮初始條件把補解加入即會包含對應於頻率$\bar{n}\Delta\omega$之反應。另一簡單做法為避免選用之週期$T$等於$\bar{n}(2\pi/\Omega)$。

17.10 利用富利葉轉換求拉卜拉斯轉換及反轉換

實函數$y(t)$之拉卜拉斯轉換為

$$G(s) = \int_0^\infty g(t)e^{-st}dt \tag{17.58}$$

其中s為複數，如令

$$s = c + i\omega \tag{17.59}$$

則式(17.58)變成

$$G(c+i\omega) = \int_0^\infty [g(t)e^{-ct}]e^{-i\omega t}dt \tag{17.60}$$

並令$t<0$時，$g(t)=0$，則上式積分下限可改為$-\infty$。因此$g(t)$之拉卜拉斯轉換等於$g(t)e^{-ct}$之富利葉轉換。

拉卜拉斯反轉換可引用式(17.60)之關係而得如下所述之做法：首先令$s=c+i\omega$而將$G(s)$寫成$G(c+i\omega)$再令其等於$F(\omega)$。接著求$F(\omega)$之富利葉反轉換得$f(t)$。因$f(t)=g(t)e^{-ct}$故得$g(t)=e^{ct}f(t)$即為$G(s)$之拉卜拉斯反轉換。

拉卜拉斯轉換時，c值必須選大於其轉換函數$G(s)$之所有極點之實部值。雖然任何滿足該條件之c值均可選用，但較大c值有好處也有缺點。好處為大的c值將使$g(t)e^{-ct}$較迅速趨近於零，而可選用較小之積分範圍。缺點則為於反轉換求$g(t)$時，大的c值使較大t值之$g(t)$不準確，因較大t值之$g(t)e^{-ct}$太小而具有較大之捨去誤差。故c值不宜太大，亦不宜太小。

17.11 快速富利葉轉換之原理

前述各種轉換中，若為$\sum_o^N$之求和型式除$\sum$前之係數不同外，均與下式類似(式中$W=e^{-2\pi i/N}$，並令$W(k,l)=W^{lk}$)：

$$F(k) = \sum_{l=0}^{N-1} f(l) * W^{lk}\,, \qquad k=0,1,\ldots,N-1 \tag{17.61}$$

上式為富利葉轉換之一般計算公式，相當於$N\times N$方陣$[W(k,l)]$乘以N元向量$\{f(l)\}$算得N元向量$\{F(k)\}$。其所需運算量為$N*N$個複數乘及加。而所謂快速富利葉轉換(FFT)係利用W^{lk}具有循環之特性，將運算量減為$N*(N1+N2+\cdots+Nm)$個複數乘及加，其中$N=N1*N2*\cdots*Nm$。如以$N=1024$為例，$N1=N2=\cdots=N9=N10=2$，則兩者運算量之比為$1024:20$或約$50:1$。可見其快速而名之。

首先說明一種以複基底表示整數之方法。日常生活之秒、分、時、日即為典型之複基底表示法。為方便計，2日5時10分30秒寫為(30,10,5,2)

。因60秒為1分，60稱為秒之基底，餘類推。如以一星期為一循環，則所有基底可合併寫成(60,60,24,7)。利用此基底最多可表示至一個星期之時間秒數，而(30,10,5,2)所代表之秒數等於$30 + 10*60 + 5*60*60 + 2*60*60*24 = 191430$。一般而言，以$(N1, N2, \ldots, Nm)$為複基底可表示0至$N-1$間之任意整數值。其中$N = N1*N2*\cdots*Nm$。另外$FORTRAN$之$DIMENSION\ A(0{:}1, 0{:}4, 0{:}6)$相當於宣告指標之基底為(2,5,7)，即為典型之複基底表示指標之方法。

為了便於說明，茲規定符號之用法如下：$i1, j1, k1, l1$之對應基底為$N1$；$i2, j2, k2, l2$之對應基底為$N2$；餘類推。因此$(l1, l2, k5, k4, i3)$之對應基底為$(N1, N2, N5, N4, N3)$。並規定括號內最左邊之基底為最低位基底；最右邊之基底為最高位基底；餘類推。

根據上述複基底表示法之符號規定，並以$N = N1*N2*N3*N4*N5$為例，式(17.61)可寫成：

$$F(k5, k4, k3, k2, k1)$$

$$= \sum_{l1=0}^{N1-1} \sum_{l2=0}^{N2-1} \sum_{l3=0}^{N3-1} \sum_{l4=0}^{N4-1} \sum_{l5=0}^{N5-1} f(l1, l2, l3, l4, l5) * W^{(l1,l2,l3,l4,l5)*(k5,k4,k3,k2,k1)} \quad (17.62)$$

現因

$$\begin{aligned}
&(l1, l2, l3, l4, l5) * (k5, k4, k3, k2, k1) \\
&= (0, 0, 0, 0, l5) * (k5, k4, k3, k2, k1) \\
&\quad + (0, 0, 0, l4, 0) * (k5, k4, k3, k2, k1) \\
&\quad + (0, 0, l3, 0, 0) * (k5, k4, k3, k2, k1) \\
&\quad + (0, l2, 0, 0, 0) * (k5, k4, k3, k2, k1) \\
&\quad + (l1, 0, 0, 0, 0) * (k5, k4, k3, k2, k1) \\
&= (0, 0, 0, 0, l5) * (k5, 0, 0, 0, 0) + (0, 0, 0, 0, l5) * (0, k4, k3, k2, k1) \\
&\quad + (0, 0, 0, l4, 0) * (k5, k4, 0, 0, 0) + (0, 0, 0, l4, 0) * (0, 0, k3, k2, k1) \\
&\quad + (0, 0, l3, 0, 0) * (k5, k4, k3, 0, 0) + (0, 0, l3, 0, 0) * (0, 0, 0, k2, k1) \\
&\quad + (0, l2, 0, 0, 0) * (k5, k4, k3, k2, 0) + (0, l2, 0, 0, 0) * (0, 0, 0, 0, k1) \\
&\quad + (l1, 0, 0, 0, 0) * (k5, k4, k3, k2, k1)
\end{aligned}$$

$$
\begin{aligned}
&= (0,0,0,0,l5)*(k5,0,0,0,0)+N*(l5)*(k4,k3,k2,k1)\\
&+(0,0,0,l4,0)*(k5,k4,0,0,0)+N*(l4)*(k3,k2,k1)\\
&+(0,0,l3,0,0)*(k5,k4,k3,0,0)+N*(l3)*(k2,k1)\\
&+(0,l2,0,0,0)*(k5,k4,k3,k2,0)+N*(l2)*(k1)\\
&+(l1,0,0,0,0)*(k5,k4,k3,k2,k1) \qquad (17.63)
\end{aligned}
$$

最後一個等號之結果係由 $(0,0,0,0,l5) = N1{*}N2{*}N3{*}N4{*}(l5)$ 餘類推及 $(0,k4,k3,k2,k1) = N5*(k4,k3,k2,k1)$ 餘類推及 $N = N1{*}N2{*}N3{*}N4{*}N5$ 等關係推導而得。再利用 $W^{(x+y+z)} = W^xW^yW^z$ ，及 $W^N = 1$ 之關係，式 (17.62) 可寫成

$$
\begin{aligned}
&F(k5,k4,k3,k2,k1)\\
&= \sum_{l1=0}^{N1-1}\Big(\sum_{l2=0}^{N2-1}\Big(\sum_{l3=0}^{N3-1}\Big(\sum_{l4=0}^{N4-1}\Big(\sum_{l5=0}^{N5-1} f(l1,l2,l3,l4,l5)*W^{(0,0,0,0,l5)*(k5,0,0,0,0)}\Big)\\
&\qquad *W^{(0,0,0,l4,0)*(k5,k4,0,0,0)}\Big)\\
&\qquad *W^{(0,0,l3,0,0)*(k5,k4,k3,0,0)}\Big)\\
&\qquad *W^{(0,l2,0,0,0)*(k5,k4,k3,k2,0)}\Big)\\
&\qquad *W^{(l1,0,0,0,0)*(k5,k4,k3,k2,k1)} \qquad (17.64)
\end{aligned}
$$

上式又可分解成下列五個子轉換階段及最後一個基底反向易位處理：

$$
\begin{aligned}
F_1(l1,l2,l3,l4,k5) &= \sum_{l5=0}^{N5-1} f\ (l1,l2,l3,l4,l5)*W^{(0,0,0,0,l5)*(k5,0,0,0,0)}\\
F_2(l1,l2,l3,k4,k5) &= \sum_{l4=0}^{N4-1} F_1(l1,l2,l3,l4,k5)*W^{(0,0,0,l4,0)*(k5,k4,0,0,0)}\\
F_3(l1,l2,k3,k4,k5) &= \sum_{l3=0}^{N3-1} F_2(l1,l2,l3,k4,k5)*W^{(0,0,l3,0,0)*(k5,k4,k3,0,0)}\\
F_4(l1,k2,k3,k4,k5) &= \sum_{l2=0}^{N2-1} F_3(l1,l2,k3,k4,k5)*W^{(0,l2,0,0,0)*(k5,k4,k3,k2,0)}\\
F_5(k1,k2,k3,k4,k5) &= \sum_{l1=0}^{N1-1} F_4(l1,k2,k3,k4,k5)*W^{(l1,0,0,0,0)*(k5,k4,k3,k2,k1)}\\
F\ (k5,k4,k3,k2,k1) &= F_5(k1,k2,k3,k4,k5) \qquad (17.65)
\end{aligned}
$$

就第一階段而言，可將 $(l1,l2,l3,l4)$ 相同者視為一組，每一組有 $N5$ 個 f 及 $N5$ 個 F_1 。每一組中之 $N5$ 個 F_1 僅與同一組中之 $N5$ 個 f 有關，因此每一組

之$N5$元向量$\{F_1\}$等於$N5\times N5$之方陣乘以同一組之$N5$元向量$\{f\}$，故此階段共需做$N*N5$個複數乘及加。同理，第二階段，可將$(l1,l2,l3)$及$(k5)$相同者視為一組，每一組有$N4$個F_1及$N4$個F_2，此階段共需做$N*N4$個複數乘及加。第三階段，可將$(l1,l2)$及$(k5,k4)$相同者視為一組，每一組有$N3$個F_2及$N3$個F_3，此階段共需做$N*N3$個複數乘及加。第四階段，可將$(l1)$及$(k5,k4,k3)$相同者視為一組，每一組有$N2$個F_3及$N2$個F_4，此階段共需做$N*N2$個複數乘及加。第五階段，可將$(k5,k4,k3,k2)$相同者視為一組，每一組有$N1$個F_4及$N1$個F_5。此階段共需做$N*N1$個複數乘及加。故所需做之總運算量即由原來之$N*N$減少至$N*(N1+N2+N3+N4+N5)$。

這裡須提醒注意的是：能夠分解成式(17.65)之五個階段之重要條件有二：(1)式(17.63)最後一個等號右邊之所有偶數項均為N之整數倍。而其所以會正好如此之原因為W^{lk}中k與l之基底必須高低位基底反向易位排列。即k之基底為$(N5,N4,N3,N2,N1)$；而l之基底為$(N1,N2,N3,N4,N5)$。(2)為$W^{nN}=W^0=1$。即函數W^x為週期N之週期函數(n,x為任意整數)。

分段計算過程中之中間結果可以用各種方便之順序存置(即採用方便之基底排列順序)，例如式(17.64)亦可用下列五個子轉換階段計算之，但最後之轉換結果$\{F(k)\}$之基底$(N5,N4,N3,N2,N1)$，必須與輸入數據$\{f(l)\}$之基底$(N1,N2,N3,N4,N5)$反向易位。因此才需式(17.65)之基底反向易位處理。而利用下式則可以免除基底反向易位之處理。

$$
\begin{aligned}
F_1(l1,l2,l3,l4,k5) &= \sum_{l5=0}^{N5-1} f\ (l1,l2,l3,l4,l5)*W^{(0,0,0,0,l5)*(k5,0,0,0,0)}\\
F_2(l1,l2,l3,k5,k4) &= \sum_{l4=0}^{N4-1} F_1(l1,l2,l3,l4,k5)*W^{(0,0,0,l4,0)*(k5,k4,0,0,0)}\\
F_3(l1,l2,k5,k4,k3) &= \sum_{l3=0}^{N3-1} F_2(l1,l2,l3,k5,k4)*W^{(0,0,l3,0,0)*(k5,k4,k3,0,0)}\\
F_4(l1,k5,k4,k3,k2) &= \sum_{l2=0}^{N2-1} F_3(l1,l2,k5,k4,k3)*W^{(0,l2,0,0,0)*(k5,k4,k3,k2,0)}\\
F\ (k5,k4,k3,k2,k1) &= \sum_{l1=0}^{N1-1} F_4(l1,k5,k4,k3,k2)*W^{(l1,0,0,0,0)*(k5,k4,k3,k2,k1)}
\end{aligned}
\tag{17.66}
$$

式(17.65)與式(17.66)各有優劣。如用式(17.65)，轉換結果可以放回原來之資料位置。即所謂就原位置 (In place)計算。但最後需做基底反向易位

處理。此項基底反向易位處理亦可將基底反向易位後之結果$\{F\}$放在原來資料位置$\{F_5\}$，但須滿足下述條件：基底反向後之基底須與反向前之基底相同。如用式(17.66)，轉換結果難以放回原來之資料位置，因此另外需要N個暫時運作空間做為儲存中途資料之替代位置，但不需做基底反向易位處理。當基底全為2時，建議利用式(17.65)，因可不必使用暫時運作空間；但當基底不為2時，建議利用式(17.66)，雖然需使用暫時運作空間，但可免除基底反向易位處理，運算較為簡潔。

17.12 程式之考慮

本章所附副程式$FFTA$共有三個版本：(1)簡潔版；(2)實用版；(3)省時版。簡潔版係利用式(17.66)直接寫成，該程式必須用到暫時運作空間。實用版則係針對基底為2之情形以式(17.65)就原位置計算，而可不用暫時運作空間，並於所有2之基底做完後即做基底反向易位，如有其他奇數基底再用式(17.66)做轉換，因此需使用暫時運作空間。另外對於基底為2之情形，以NOW=$N3$=2為例，如成對計算$F_3(l1,l2,0,k4,k5)$與$F_3(l1,l2,1,k4,k5)$二值，由下式可知，僅需做一次乘算，比簡潔版需做二次乘算可減少一半之時間。注意式中$x=(0,0,1,0,0)*(k5,k4,0,0,0)$，而$(0,0,1,0,0)*(k5,k4,1,0,0)=x+N1*N2*N5*N4=x+(N/2)$，$W^{(N/2)}=-1$，故$W^{(0,0,1,0,0)*(k5,k4,1,0,0)}=-W^{(0,0,1,0,0)*(k5,k4,0,0,0)}$。

$$\begin{Bmatrix} F_3(l1,l2,0,k4,k5) \\ F_3(l1,l2,1,k4,k5) \end{Bmatrix}$$
$$= \begin{bmatrix} W^{(0,0,0,0,0)*(k5,k4,0,0,0)} & W^{(0,0,1,0,0)*(k5,k4,0,0,0)} \\ W^{(0,0,0,0,0)*(k5,k4,1,0,0)} & W^{(0,0,1,0,0)*(k5,k4,1,0,0)} \end{bmatrix} \begin{Bmatrix} F_2(l1,l2,0,k4,k5) \\ F_2(l1,l2,1,k4,k5) \end{Bmatrix}$$
$$= \begin{bmatrix} 1 & W^x \\ 1 & -W^x \end{bmatrix} \begin{Bmatrix} F_2(l1,l2,0,k4,k5) \\ F_2(l1,l2,1,k4,k5) \end{Bmatrix}$$

省時版對暫時運作空間之需求與實用版相同：即若N為2之整數次方時可不用暫時運作空間，否則呼叫程式須提供此空間。對計算效率之改進考慮下列二項：(1)對於偶數基底，增加採用4之基底，並利用扭轉因數(twiddle factor)，可再減少實用版之1/4乘算時間；(2)對於奇數基底，亦利用扭轉因數及共軛特性，可比前二版減少1/4至3/4之乘算時間。首先說明扭轉因數：仍以NOW=$N3$為例，式(17.65)可分成下列二式：

$$F_2'(l1,l2,l3,k4,k5) = F_2(l1,l2,l3,k4,k5) * W^{(0,0,l3,0,0)*(k5,k4,0,0,0)} \tag{17.67}$$

$$F_3(l1,l2,k3,k4,k5) = \sum_{l3=0}^{N3-1} F_2'(l1,l2,l3,k4,k5) * W^{(0,0,l3,0,0)*(0,0,k3,0,0)} \tag{17.68}$$

式(17.67)之$W^{(0,0,l3,0,0)*(k5,k4,0,0,0)}$稱扭轉因數。用乘過扭轉因數之$F_2'$可使式(17.68)之轉換矩陣$[W]$之型式較簡，茲針對基底4與奇數基底分述如下：

(1)基底為4時，$W^{(0,0,l3,0,0)*(0,0,k3,0,0)} = W^{l3*k3*N/4}$，因此可得：

$$[W] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

可知式(17.68)不必做任何乘算(乘i僅需將實部與虛部對調再做適當變號即可)，而扭轉因數之乘算數目為$3N/4$，因其中對應$l3=0$之$N/4$個扭轉因數為1。注意實用版做二次基底2之乘算數目為$2*N/2=N$。理論上採用基底8可再減少乘算數量，但改進效果不大，故未考慮於程式中。

(2)基底為奇數時，以NOW=$N3$=5為例，$W^{(0,0,l3,0,0)*(0,0,k3,0,0)} = W^{l3*k3*N/5} = U^{l3*k3}$，式中$U = e^{-2\pi i/5}$，因此可得：

$$[W] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & U^1 & U^2 & U^3 & U^4 \\ 1 & U^2 & U^4 & U^1 & U^3 \\ 1 & U^3 & U^1 & U^4 & U^2 \\ 1 & U^4 & U^3 & U^2 & U^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & U^1 & U^2 & \bar{U}^2 & \bar{U}^1 \\ 1 & U^2 & U^4 & \bar{U}^4 & \bar{U}^2 \\ 1 & \bar{U}^2 & \bar{U}^4 & U^4 & U^2 \\ 1 & \bar{U}^1 & \bar{U}^2 & U^2 & U^1 \end{bmatrix}$$

式中U與$\bar{U}$之次方x相同者如做為一組合併計算如式(17.69)所示(式中U代表U^x)，即可經如式(17.70)之安排，先算Xt, Yt，再算Xo, Yo，則僅需4次實數乘算，等於1次複數乘。因共有$(N3-1)^2/4$組，故乘算數量為$N*(N3-1)^2/(4*N3)$，再加上扭轉因數之乘算數量$N*(N3-1)/N3$，即為此階段之乘算總數量。若用簡潔版則此數量為$N*(N3-1)^2/N3$。當$N3 = (3,5,7,11,13,17)$時二者之比為$\frac{1}{4}+\frac{1}{N3-1} = (0.75, 0.50, 0.42, 0.35, 0.33, 0.31)$。

$$\begin{Bmatrix} Xo \\ Yo \end{Bmatrix} = \begin{bmatrix} U & \bar{U} \\ \bar{U} & U \end{bmatrix} \begin{Bmatrix} Xi \\ Yi \end{Bmatrix} \tag{17.69}$$

$$\begin{aligned} Xt = (Xo+Yo)/2 &= U_r\,(Xi+Yi)\,, \quad Xo = Xt+Yt \\ Yt = (Xo-Yo)/2 &= i*U_i\,(Xi-Yi)\,, \quad Yo = Xt-Yt \end{aligned} \tag{17.70}$$

17.13 實數及共軛複數之轉換

實際應用f_l常為實數，即虛部全為零。而轉換後之元素F_k具有$F_{-k} = F_{N-k} = \bar{F}_k$之關係。其中必然浪費了許多對零之多餘運算。以下介紹二種較省時之方法：(1)將二個相同個數(均為N)之實數h_l與g_l一起做轉換之方式；(2)若實數f_l之資料個數為偶數(設為$2N$)，可將其視為N個複數做轉換之方式。

方式(1)使用上較不方便，但有關計算式會用於方式(2)故先行介紹：將h_l當做實數部分，g_l當做虛數部分，則可求得$h_l + ig_l$之富利葉轉換為$R_k + iI_k$。其與h_l之轉換$Hr_k + iHi_k$及g_l之轉換$Gr_k + iGi_k$之關係為$R_k + iI_k = (Hr_k + iHi_k) + i(Gr_k + iGi_k)$，因此得式(17.71)，再利用實數之轉換值$H_{N-k} = \bar{H}_k$，即$Hr_{N-k} = Hr_k$與$Hi_{N-k} = -Hi_k$之關係，可得式(17.72)。由式(17.71)與式(17.72)即可解得式(17.73)與式(17.74)。

$$R_k = Hr_k - Gi_k\,, \quad I_k = Hi_k + Gr_k \tag{17.71}$$

$$R_{N-k} = Hr_k + Gi_k\,, \quad I_{N-k} = -Hi_k + Gr_k \tag{17.72}$$

$$Hr_k = \frac{1}{2}(R_k + R_{N-k})\,, \quad Hi_k = \frac{1}{2}(I_k - I_{N-k}) \tag{17.73}$$

$$Gi_k = -\frac{1}{2}(R_k - R_{N-k})\,, \quad Gr_k = \frac{1}{2}(I_k + I_{N-k}) \tag{17.74}$$

方式(2)係令$h_l = f_{2l}$當做實數部分，$g_l = f_{2l+1}$當做虛數部分，則由方式(1)可求得H_k與G_k如上二式所示。再由下列關係即可計算F_k。

$$\begin{aligned} F_k &= \sum_{j=0}^{2N-1} f_j e^{-2kj\pi i/2N} = \sum_{l=0}^{N-1} f_{2l} e^{-2kl\pi i/N} + \sum_{l=0}^{N-1} f_{2l+1} e^{-k(2l+1)\pi i/N} \\ &= H_k + G_k e^{-k\pi i/N} = H_k + iG'_k \\ Fr_k &= Hr_k - Gi_k \sin(\frac{-k\pi}{N}) + Gr_k \cos(\frac{-k\pi}{N}) = Hr_k - Gi'_k \qquad (17.75) \\ Fr_{N-k} &= Hr_k + Gi_k \sin(\frac{-k\pi}{N}) - Gr_k \cos(\frac{-k\pi}{N}) = Hr_k + Gi'_k \qquad (17.76) \\ Fi_k &= Hi_k + Gr_k \sin(\frac{-k\pi}{N}) + Gi_k \cos(\frac{-k\pi}{N}) = Hi_k + Gr'_k \qquad (17.77) \\ Fi_{N-k} &= -Hi_k + Gr_k \sin(\frac{-k\pi}{N}) + Gi_k \cos(\frac{-k\pi}{N}) = -Hi_k + Gr'_k \qquad (17.78) \end{aligned}$$

對於共軛複數值之富利葉轉換可由前述步驟做逆向運算：參考上列各式令$Gi'_k = Gi_k \sin(\frac{-k\pi}{N}) - Gr_k \cos(\frac{-k\pi}{N})$，$Gr'_k = Gr_k \sin(\frac{-k\pi}{N}) + Gi_k \cos(\frac{-k\pi}{N})$，

則由已知之F_k按上列四式可求得H_k與G'_k，其計算式類似式(17.73)，再由$Gi_k = Gi'_k \sin(\frac{-k\pi}{N}) + Gr'_k \cos(\frac{-k\pi}{N})$，$Gr_k = Gr'_k \sin(\frac{-k\pi}{N}) - Gi'_k \cos(\frac{-k\pi}{N})$，算得$G_k$後，代入式(17.71)與式(17.72)，可求得R_k與I_k，最後由此N個複數值做富利葉逆轉換即可得$2N$個實數f_l。注意式(17.75)至式(17.78)與式(17.71)至式(17.72)完全相似，可知逆算與順算過程一致，計算式中僅G_k與G'_k互換公式內之cos項之係數符號不同。詳副程式$FFTR$與$FFTC$。

17.14 正弦及餘弦轉換

實數之富利葉轉換之虛數部分變號後等於下列之正弦(sin)轉換：

$$F_k = \sum_{l=0}^{N-1} f_l \sin(\frac{2kl\pi}{N}) \tag{17.79}$$

實數之富利葉轉換之實數部分則等於下列之餘弦(cos)轉換：

$$F_k = \sum_{l=0}^{N-1} f_l \cos(\frac{2kl\pi}{N}) \tag{17.80}$$

若將$2N$個實數f_l寫成反對稱項$a_l = (f_l - f_{2N-l})/2$與對稱項$b_l = (f_l + f_{2N-l})/2$之和，即$f_l = a_l + b_l$。則a_l之富利葉轉換為反對稱之純虛數，與f_l之富利葉轉換之虛數部分相同；而b_l之富利葉轉換為對稱之純實數，與f_l之富利葉轉換之實數部分相同。因此若用a_l之前N個值做半正弦轉換(詳次節)，則其二倍值即為$2N$個f_l值之正弦轉換之前N個值，其餘之值可由反對稱求得。同理若用b_l之前$N+1$個值做半餘弦轉換(詳次節)，其二倍值即為$2N$個f_l值之餘弦轉換之前$N+1$個值，其餘之值可由對稱求得。詳副程式$FFTCS$。

17.15 半正弦及半餘弦轉換

有些應用須做實數之半正弦(half-sin)轉換，即(式中$k = 0, \ldots, N-1$)

$$F_k = \sum_{l=0}^{N-1} f_l \sin(\frac{kl\pi}{N}) \tag{17.81}$$

$$f_l = \frac{2}{N} \sum_{l=0}^{N-1} F_k \sin(\frac{kl\pi}{N}) \tag{17.82}$$

或半餘弦(half-cos)轉換，即(式中$k=0,\ldots,N$，$\sum''$表示首末二項須乘$\frac{1}{2}$)

$$F_k = \sum_{l=0}^{N}{}'' f_l \cos(\frac{kl\pi}{N}) \tag{17.83}$$

$$f_l = \frac{2}{N}\sum_{l=0}^{N}{}'' F_k \cos(\frac{kl\pi}{N}) \tag{17.84}$$

如令$f_{2N-l}=-f_l$，$f_o=f_N=0$，使成為反對稱之$2N$個資料，再用富利葉轉換計算之，則所得之轉換為純虛數，該值除以$2i$即為半正弦轉換。如令$f_{2N-l}=f_l$，使成為對稱之$2N$個資料，再用富利葉轉換計算之，則所得之轉換為純實數，該值除以2即為半餘弦轉換。但以上做法除需二倍計算空間外亦較費時。以下介紹較省時之做法：

(1)半正弦轉換先計算下列輔助數列：

$$\begin{aligned} y_o &= 0 \\ y_l &= \frac{1}{2}(f_l - f_{N-l}) + \sin(\frac{l\pi}{N})(f_l + f_{N-l}), \quad l=1,\ldots,N-1 \end{aligned} \tag{17.85}$$

計算其富利葉轉換(可用實係數之程式$FFTR$)，得$R_k + i\,I_k$，其與F_k之關係如下，由此即可計算F_k。詳副程式$FFTSC$。

$$F_{2k} = I_k, \quad F_{2k+1} = F_{2k-1} + R_k, \quad k=0,\ldots,(N/2-1) \tag{17.86}$$

$$\text{由上式令 } k=0 \text{ 及因 } F_1 = -F_{-1} \text{ 可得} \qquad F_1 = \frac{1}{2}R_o \tag{17.87}$$

(2)半餘弦轉換先計算下列輔助數列：

$$y_l = \frac{1}{2}(f_l + f_{N-l}) - \sin(\frac{l\pi}{N})(f_l - f_{N-l}), \quad l=0,\ldots,N-1 \tag{17.88}$$

計算其富利葉轉換，得$R_k + i\,I_k$，其與F_k之關係如下，由此即可計算F_k。詳副程式$FFTSC$。

$$F_{2k} = R_k, \qquad F_{2k+1} = F_{2k-1} + I_k, \qquad k=0,\ldots,(N/2-1) \tag{17.89}$$

$$\text{由式(17.83)令 } k=1 \text{ 可得 } F_1 = \frac{1}{2}(f_o - f_N) + \sum_{l=1}^{N-1} f_l \cos(\frac{l\pi}{N}) \tag{17.90}$$

(3)半餘弦(half-cos)轉換亦有用如下之定義者，此式僅用N個函數值(式(17.83)需$N+1$個值)，即(式中$k=0,\ldots,N-1$，$\sum'$表示首項須乘$\frac{1}{2}$)

$$F_k = \sum_{l=0}^{N-1} f_l \cos\frac{k(l+\frac{1}{2})\pi}{N} \tag{17.91}$$

$$f_l = \frac{2}{N}\sum_{l=0}^{N-1}{}' F_k \cos\frac{k(l+\frac{1}{2})\pi}{N} \tag{17.92}$$

式(17.91)之轉換可先計算下列輔助數列：

$$y_l = \frac{1}{2}\,(f_l + f_{N-1-l}) - \sin\frac{(l+\frac{1}{2})\pi}{N}\,(f_l - f_{N-1-l})\,, \quad l = 0,\ldots,N-1 \tag{17.93}$$

計算其富利葉轉換，得 $R_k + i\,I_k$ ，其與 F_k 之關係如下，由此即可按 $k = N/2-1$ 遞減至 $k=0$ 之順序計算 $F_{2k},\ F_{2k+1}$ 。詳副程式 $FFTCH$ 。

$$F_{2k} = R_k\cos(\frac{k\pi}{N}) - I_k\sin(\frac{k\pi}{N}) \tag{17.94}$$

$$F_{2k+1} - F_{2k-1} = R_k\sin(\frac{k\pi}{N}) + I_k\cos(\frac{k\pi}{N})\,, \quad k = (N/2-1),\ldots,0 \tag{17.95}$$

由上式令 $k = N/2$ 及因 $F_{N-1} = -F_{N+1}$ 可得 $\qquad F_{N-1} = -\frac{1}{2}\,R_{N/2}$ (17.96)

式(17.92)之轉換可照上述做法逆向反求：即由 F_k 按下式計算 $R_k,\ I_k$ ，由 $FFTC$ 求 y_j ，再由式(17.99)計算 f_j 。詳副程式 $FFTCH$ 。

$$R_k = (F_{2k+1} - F_{2k-1})\sin(\frac{k\pi}{N}) + F_{2k}\cos(\frac{k\pi}{N}) \tag{17.97}$$

$$I_k = (F_{2k+1} - F_{2k-1})\cos(\frac{k\pi}{N}) - F_{2k}\sin(\frac{k\pi}{N})\,, \quad k = (N/2-1),\ldots,0 \tag{17.98}$$

$$f_l = \frac{1}{2}\,(y_l + y_{N-1-l}) - (y_l - y_{N-1-l})/4\sin\frac{(l+\frac{1}{2})\pi}{N}\,, \quad l = 0,\ldots,N-1 \tag{17.99}$$

以下証明式(17.86)之關係

$$\begin{aligned} R_k &= \sum_{l=0}^{N-1} y_l\,\cos(\frac{2kl\pi}{N}) = \sum_{l=0}^{N-1}(f_l + f_{N-l})\sin(\frac{l\pi}{N})\cos(\frac{2kl\pi}{N}) \\ &= \sum_{l=0}^{N-1} 2f_l\,\sin(\frac{l\pi}{N})\cos(\frac{2kl\pi}{N}) \\ &= \sum_{l=0}^{N-1} f_l\,\left[\sin\frac{(2k+1)l\pi}{N} - \sin\frac{(2k-1)l\pi}{N}\right] = F_{2k+1} - F_{2k-1} \end{aligned} \tag{17.100}$$

$$\begin{aligned} I_k &= \sum_{l=0}^{N-1} y_l\,\sin(\frac{2kl\pi}{N}) = \sum_{l=0}^{N-1}\frac{1}{2}\,(f_l - f_{N-l})\sin(\frac{2kl\pi}{N}) \\ &= \sum_{l=0}^{N-1} f_l\,\sin(\frac{2kl\pi}{N}) = F_{2k} \end{aligned} \tag{17.101}$$

以下証明式(17.89)之關係

$$R_k = \sum_{l=0}^{N-1} y_l\,\cos(\frac{2kl\pi}{N}) = \sum_{l=0}^{N-1}\frac{1}{2}\,(f_l + f_{N-l})\cos(\frac{2kl\pi}{N})$$

$$= \sum_{l=0}^{N-1} f_l \cos(\frac{2kl\pi}{N}) = F_{2k} \tag{17.102}$$

$$\begin{aligned} I_k &= \sum_{l=0}^{N-1} y_l \sin(\frac{2kl\pi}{N}) = \sum_{l=0}^{N-1} -(f_l - f_{N-l}) \sin(\frac{l\pi}{N}) \sin(\frac{2kl\pi}{N}) \\ &= \sum_{l=0}^{N-1} -2f_l \sin(\frac{l\pi}{N}) \sin(\frac{2kl\pi}{N}) \\ &= \sum_{l=0}^{N-1} f_l \left[\cos\frac{(2k+1)l\pi}{N} - \cos\frac{(2k-1)l\pi}{N}\right] = F_{2k+1} - F_{2k-1} \end{aligned} \tag{17.103}$$

以下証明式(17.94)與式(17.95)之關係：令$R'_k = R_k \cos(\frac{k\pi}{N}) - I_k \sin(\frac{k\pi}{N})$，$I'_k = R_k \sin(\frac{k\pi}{N}) + I_k \cos(\frac{k\pi}{N})$，由下列二式即可得式(17.104)與式(17.105)。

$$\begin{aligned} \cos(\frac{2kl\pi}{N}) \cos(\frac{k\pi}{N}) - \sin(\frac{2kl\pi}{N}) \sin(\frac{k\pi}{N}) &= \cos\frac{2k(l+\frac{1}{2})\pi}{N} \\ \cos(\frac{2kl\pi}{N}) \sin(\frac{k\pi}{N}) + \sin(\frac{2kl\pi}{N}) \cos(\frac{k\pi}{N}) &= \sin\frac{2k(l+\frac{1}{2})\pi}{N} \end{aligned}$$

$$\begin{aligned} R'_k &= \sum_{l=0}^{N-1} y_l \cos\frac{2k(l+\frac{1}{2})\pi}{N} = \sum_{l=0}^{N-1} \frac{1}{2}(f_l + f_{N-1-l}) \cos\frac{2k(l+\frac{1}{2})\pi}{N} \\ &= \sum_{l=0}^{N-1} f_l \cos\frac{2k(l+\frac{1}{2})\pi}{N} = F_{2k} \end{aligned} \tag{17.104}$$

$$\begin{aligned} I'_k &= \sum_{l=0}^{N-1} y_l \sin\frac{2k(l+\frac{1}{2})\pi}{N} = \sum_{l=0}^{N-1} -(f_l - f_{N-1-l}) \sin\frac{(l+\frac{1}{2})\pi}{N} \sin\frac{2k(l+\frac{1}{2})\pi}{N} \\ &= \sum_{l=0}^{N-1} -2f_l \sin\frac{(l+\frac{1}{2})\pi}{N} \sin\frac{2k(l+\frac{1}{2})\pi}{N} \\ &= \sum_{l=0}^{N-1} f_l \left[\cos\frac{(2k+1)(l+\frac{1}{2})\pi}{N} - \cos\frac{(2k-1)(l+\frac{1}{2})\pi}{N}\right] \\ &= F_{2k+1} - F_{2k-1} \end{aligned} \tag{17.105}$$

17.16 二維轉換

為了方便使用，文內同時附上可以直接用於二維轉換之副程式$FFT2D$、與$FFT2SC$。其他二維或二維以上之轉換可仿該二副程式寫成。

17.17 其他注意事項

最後應注意有些轉換須除以某些常數如N或$N/2$，應於呼叫所附副程式之前或之後自行計算之。

富利葉級數與富利葉積分均可表示成離散富利葉轉換而能利用快速富利葉轉換之方法計算之。富利葉轉換之應用非常廣泛，文內以結構動力系統之反應分析為例，說明富利葉轉換不但可求得穩態解，亦可求得任意初始條件之暫態解，因此使快速之富利葉轉換能夠用以解決各種動力反應之問題。

因富利葉轉換獨特具有週期循環之特性，而能發展出快速轉換之計算方法，使其能夠更有效的被運用於各種領域之分析。惟快速富利葉轉換因系屬一種數值計算，難免有誤差存在，而使其與富利葉級數及富利葉積分之性質有些差異，使用時應予詳細了解，方能正確地使用這項計算利器。

[表二(a)] 快速富利葉轉換副程式(簡潔版)

```
************************************************************************
      SUBROUTINE FFTA(F,A,N,INV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 F(0:N-1),A(0:N-1)
C**   ============================================================    **
C**   F(N) = Input (FI) and Output (FO) data array of arbitrary N     **
C**   A(N) = Working array (F1,F2,F3,...)                             **
C**   ============================================================    **
      IF(N.LE.1) RETURN
      IFA=1
      NBF=N
      NAF=1
C**   +-----------------------------------------------------------+    **
C**   | NOW = 2,..,3,..,5,..,7,..,11,..,13,..,17,..               |    **
C**   +-----------------------------------------------------------+    **
      NOW=2
   20 IF((NBF/NOW)*NOW.NE.NBF) THEN
        NOW=NOW+1
        GO TO 20
      ENDIF
      NBF=NBF/NOW
      IF(IFA.GT.0) CALL FFTBNA(F,A,NBF,NOW,NAF,INV)
      IF(IFA.LT.0) CALL FFTBNA(A,F,NBF,NOW,NAF,INV)
      IFA=-IFA
      NAF=NAF*NOW
      IF(NBF.GT.1) GOTO 20
C**   +-----------------------------------------------------------+    **
C**   | Return transformed values in F(0:N-1)                     |    **
C**   +-----------------------------------------------------------+    **
      IF(IFA.GT.0) RETURN
      DO 50 I=0,N-1
```

```
   50 F(I)=A(I)
      RETURN
      END
*************************************************************************
      SUBROUTINE FFTBNA(FI,FO,NBF,NOW,NAF,INV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 FI(0:NBF-1,0:NOW-1,0:NAF-1),FO(0:NBF-1,0:NAF-1,0:NOW-1)
      COMPLEX*16 OMG,AR1,FF
C**   ============================================================    **
      ANG=-6.2831853071795864D0/ISIGN(NOW*NAF,INV)
      OMG=DCMPLX(DCOS(ANG),DSIN(ANG))
C**   ------------------------------------------------------------    **
      AR1=(1.D0,0.D0)
      DO 90 IM=0,NOW-1
      DO 80 IA=0,NAF-1
      DO 70 IB=0,NBF-1
      FF=FI(IB,NOW-1,IA)
      DO 60 IN=NOW-2,0,-1
   60 FF=FF*AR1+FI(IB,IN,IA)
   70 FO(IB,IA,IM)=FF
   80 AR1=AR1*OMG
   90 CONTINUE
      RETURN
      END
*************************************************************************
```

[表二(b)] 快速富利葉轉換副程式(實用版)

```
*************************************************************************
      SUBROUTINE FFTA(F,A,N,INV)
C ===================================================================== *
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 F(0:N-1),A(0:N-1)
C ===================================================================== *
C F(N) = Input (FI) and Output (FO) data array of arbitrary N           *
C A(N) = Working array (F1,F2,F3,...) reqd for N .NE. 2**Integer only   *
C --------------------------------------------------------------------- *
C FO(K5,K4,K3,K2,K1)=FI(L1,L2,L3,L4,L5)W^(L1,L2,L3,L4,L5)(K5,K4,K3,K2,K1)
C       where     W = EXP(i*(-2*PI*INV/N))                              *
C ===================================================================== *
C For NOW = 2  (N5=N4=N3=2)  :  by DO 290                               *
C --------------------------------------------------------------------- *
C F1(L1,L2,L3,L4,J5)=FI(L1,L2,L3,L4,I5)W^( 0, 0, 0, 0,I5)(J5, 0, 0, 0, 0)
C F2(L1,L2,L3,J4,K5)=F1(L1,L2,L3,I4,K5)W^( 0, 0, 0,I4, 0)(K5,J4, 0, 0, 0)
C F3(L1,L2,J3,K4,K5)=F2(L1,L2,I3,K4,K5)W^( 0, 0,I3, 0, 0)(K5,K4,J3, 0, 0)
C --------------------------------------------------------------------- *
C Bit-reversal       FR(L1,L2,K5,K4,K3)=F3(L1,L2,K3,K4,K5) : by DO 590  *
C --------------------------------------------------------------------- *
C For NOW > 2  (N2>2, N1>2)  :  by DO 90                                *
C --------------------------------------------------------------------- *
C F4(L1,K5,K4,K3,J2)=FR(L1,I2,K5,K4,K3)W^( 0,I2, 0, 0, 0)(K5,K4,K3,J2, 0)
C FO(K5,K4,K3,K2,J1)=F4(I1,K5,K4,K3,K2)W^(I1, 0, 0, 0, 0)(K5,K4,K3,K2,J1)
C ===================================================================== *
C W^(0,0,I3,0,0)(K5,K4,J3,0,0) = W^(0,0,1,0,0)(K5,K4,J3,0,0)I3 = AR1^I3 *
C OMG = W^(0,0,1,0,0) = EXP(i*(-2*PI*INV/N)*NBF)                        *
C AR1 = W^(0,0,1,0,0)(K5,K4,J3,0,0) = 1*OMG*OMG*OMG*...                 *
C --------------------------------------------------------------------- *
C NBFore =  N1*N2*N3*N4   N1*N2*N3    N1*N2       N1           1         *
C NOW    =       N5          N4         N3        N2           N1        *
```

```
C NAFter =       1           N5       N5*N4    N5*N4*N3   N5*N4*N3*N2   *
C ------------------------------------------------------------------- *
C IB     = (L1,L2,L3,L4) (L1,L2,L3)  (L1,L2)     (L1)          0       *
C IN     =      (I5)        (I4)       (I3)      (I2)         (I1)      *
C IA     =       0          (K5)     (K5,K4)  (K5,K4,K3) (K5,K4,K3,K2)  *
C IR     =       0          (K5)     (K4,K5)  (K3,K4,K5) (K2,K3,K4,K5)  *
C IM     =      (J5)        (J4)       (J3)      (J2)         (J1)      *
C < for NOW=N2=5 > : ------------------------------------------------- *
C       / FO(IB,IA,0) \   / W00 W01 W02 W03 W04 \ / FI(IB,0,IA) \       *
C       | FO(IB,IA,1) |   | W10 W11 W12 W13 W14 | | FI(IB,1,IA) |       *
C       | FO(IB,IA,2) | = | W20 W21 W22 W23 W24 | | FI(IB,2,IA) |       *
C       | FO(IB,IA,3) |   | W30 W31 W32 W33 W34 | | FI(IB,3,IA) |       *
C       \ FO(IB,IA,4) /   \ W40 W41 W42 W43 W44 / \ FI(IB,4,IA) /       *
C ------------------------------------------------------------------- *
C       where : Wij = W^ij,   W1j = W^j,   W1j = AR1 = OMG^(IA,j)       *
C ------------------------------------------------------------------- *
C FO($#j)=FI($0#)+(FI($1#)+(FI($2#)+(FI($3#)+FI($4#)*W1j)*W1j)*W1j)*W1j *
C =================================================================== *
      IF(N.LE.1) RETURN
      IFA=1
      NBF=N
      NAF=1
C**   +--------------------------------------------------------+    **
C**   | NOW = 2,..,3,..,5,..,7,..,11,..,13,..,17,..            |    **
C**   +--------------------------------------------------------+    **
      NOW=2
   20 IF((NBF/NOW)*NOW.NE.NBF) THEN
        NOW=NOW+1
        GO TO 20
      ENDIF
      NBF=NBF/NOW
      IF(IFA.GT.0) CALL FFTBNA(F,A,NBF,NOW,NAF,INV)
      IF(IFA.LT.0) CALL FFTBNA(A,F,NBF,NOW,NAF,INV)
      IF(NOW.NE.2) IFA=-IFA
      NAF=NAF*NOW
      IF(NBF.GT.1) GOTO 20
C**   +--------------------------------------------------------+    **
C**   | Return transformed values in F(0:N-1)                  |    **
C**   +--------------------------------------------------------+    **
      IF(IFA.GT.0) RETURN
      DO 50 I=0,N-1
   50 F(I)=A(I)
      RETURN
      END
**********************************************************************
      SUBROUTINE FFTBNA(FI,FO,NBF,NOW,NAF,INV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 FI(0:NBF-1,0:NOW-1,0:NAF-1),FO(0:NBF-1,0:NAF-1,0:NOW-1)
      COMPLEX*16 OMG,AR1,FF
C**   ============================================================    **
      ANG=-6.2831853071795864D0/ISIGN(NOW*NAF,INV)
      OMG=DCMPLX(DCOS(ANG),DSIN(ANG))
C**   ------------------------------------------------------------    **
      IF(NOW.EQ.2) GOTO 200
C**   ------------------------------------------------------------    **
      AR1=(1.D0,0.D0)
      DO 90 IM=0,NOW-1
      DO 80 IA=0,NAF-1
      DO 70 IB=0,NBF-1
```

```
      FF=FI(IB,NOW-1,IA)
      DO 60 IN=NOW-2,0,-1
   60 FF=FF*AR1+FI(IB,IN,IA)
   70 FO(IB,IA,IM)=FF
   80 AR1=AR1*OMG
   90 CONTINUE
      RETURN
C**   +-----------------------------------------------------------+    **
C**   | NOW=2 : Use dual nodes and in-place computation           |    **
C**   +-----------------------------------------------------------+    **
  200 IR=0
      AR1=(1.D0,0.D0)
      DO 290 IA=0,NAF-1
C**     +-------------------------------------------------------+    **
C**     | / FI(IB,0,IR) \   / 1  AR1 \ / FI(IB,0,IR) \          |    **
C**     | \ FI(IB,1,IR) / = \ 1 -AR1 / \ FI(IB,1,IR) /          |    **
C**     +-------------------------------------------------------+    **
        DO 280 IB=0,NBF-1
          FF=FI(IB,1,IR)*AR1
          FI(IB,1,IR)=FI(IB,0,IR)-FF
          FI(IB,0,IR)=FI(IB,0,IR)+FF
  280   CONTINUE
        IF(IA.EQ.NAF-1) GO TO 290
        IS=NAF/2
  285   IF(IS.LE.IR) THEN
          IR=IR-IS
          IS=IS/2
          GOTO 285
        ENDIF
        IR=IR+IS
        AR1=AR1*OMG
  290 CONTINUE
      IF((NBF/2)*2.EQ.NBF) RETURN
C**   +-----------------------------------------------------------+    **
C**   | Bit-reversal operation after all base 2 transformations   |    **
C**   +-----------------------------------------------------------+    **
      IR=0
      DO 590 IA=0,2*NAF-2
        IF(IA.LT.IR) THEN
          DO 540 IB=0,NBF-1
            FF=FI(IB,IR,0)
            FI(IB,IR,0)=FI(IB,IA,0)
            FI(IB,IA,0)=FF
  540     CONTINUE
        ENDIF
        IS=NAF
  550   IF(IS.LE.IR) THEN
          IR=IR-IS
          IS=IS/2
          GOTO 550
        ENDIF
        IR=IR+IS
  590 CONTINUE
      RETURN
      END
*************************************************************************
```

[表二(c)] 快速富利葉轉換副程式(省時版)

```
************************************************************************
      SUBROUTINE FFTA(F,A,N,INV)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 F(0:N-1),A(0:N-1)
C**   ================================================================    **
C**   F(N) = Input (FI) and Output (FO) data array of arbitrary N     **
C**   A(N) = Working array (F1,F2,F3,...)  for N .NE. 2**M only       **
C**   ================================================================    **
      IF(N.LE.1) RETURN
      IFA=1
      NBF=N
      NAF=1
C**   +--------------------------------------------------------------+    **
C**   | NOW = 4,..,2,3,..,5,..,7,..,11,..,13,..,17,..                |    **
C**   +--------------------------------------------------------------+    **
      NOW=4
   20 IF((NBF/NOW)*NOW.NE.NBF) THEN
        NOW=NOW+1
        IF(NOW.EQ.5) NOW=2
        IF(NOW.EQ.4) NOW=5
        GO TO 20
      ENDIF
      NBF=NBF/NOW
      IF(IFA.GT.0) CALL FFTBNA(F,A,NBF,NOW,NAF,INV)
      IF(IFA.LT.0) CALL FFTBNA(A,F,NBF,NOW,NAF,INV)
      IF(NOW.NE.2.AND.NOW.NE.4) IFA=-IFA
      NAF=NAF*NOW
      IF(NBF.GT.1) GOTO 20
C**   +--------------------------------------------------------------+    **
C**   | Return transformed values in F(0:N-1)                        |    **
C**   +--------------------------------------------------------------+    **
      IF(IFA.GT.0) RETURN
      DO 50 I=0,N-1
   50 F(I)=A(I)
      RETURN
      END
************************************************************************
      SUBROUTINE FFTBNA(FI,FO,NBF,NOW,NAF,INV)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 FI(0:NBF-1,0:NOW-1,0:NAF-1),FO(0:NBF-1,0:NAF-1,0:NOW-1)
      COMPLEX*16 OMG,AR1,AR2,AR3,FF,V0,V1,V2,V3
      COMPLEX*16 XXT,YYT,XXI,YYI,UNM,UN1,U11
      DIMENSION  XT(0:1),YT(0:1),XI(0:1),YI(0:1),U(0:1)
      EQUIVALENCE (XXT,XT),(YYT,YT),(XXI,XI),(YYI,YI),(UNM,U)
C**   ================================================================    **
      ANG=-6.283185307179586D0/ISIGN(NOW*NAF,INV)
      OMG=DCMPLX(DCOS(ANG),DSIN(ANG))
C**   ----------------------------------------------------------------    **
      IF(NOW.EQ.4) GOTO 400
      IF(NOW.EQ.2) GOTO 200
C**   +--------------------------------------------------------------+    **
C**   | NOW=Odd : use twiddle factor computation                     |    **
C**   +--------------------------------------------------------------+    **
      ANG=-6.283185307179586D0/ISIGN(NOW,INV)
      U11=DCMPLX(DCOS(ANG),DSIN(ANG))
      AR1=(1.D0,0.D0)
```

```
      DO 190 IA=0,NAF-1
C**     +-------------------------------------------------------+    **
C**     | For IA > 0 : Multiplicate twiddle factors             |    **
C**     +-------------------------------------------------------+    **
        IF(IA.GT.0) THEN
          AR1=AR1*OMG
          AR2=(1.D0,0.D0)
          DO 120 IN=1,NOW-1
          AR2=AR2*AR1
          DO 110 IB=0,NBF-1
          FI(IB,IN,IA)=FI(IB,IN,IA)*AR2
  110     CONTINUE
  120     CONTINUE
        ENDIF
C**     +-------------------------------------------------------+     **
C**     / FO(IB,IA,0) \   / U00 U01 U02 U03 U04 \ / FI(IB,0,IA) \     **
C**     | FO(IB,IA,1) |   | U10                 | | FI(IB,1,IA) |     **
C**     | FO(IB,IA,2) | = | U20                 | | FI(IB,2,IA) |     **
C**     | FO(IB,IA,3) |   | U30                 | | FI(IB,3,IA) |     **
C**     \ FO(IB,IA,4) /   \ U40                 / \ FI(IB,4,IA) /     **
C**     +-------------------------------------------------------+     **
        DO 140 IB=0,NBF-1
          FO(IB,IA,0)=FI(IB,0,IA)
          DO 130 IN=1,NOW-1
          FO(IB,IA,0)=FO(IB,IA,0)+FI(IB,IN,IA)
          FO(IB,IA,IN)=FI(IB,0,IA)
  130     CONTINUE
  140   CONTINUE
C**     +-------------------------------------------------------+     **
C**     / FO(IB,IA,0) \   /                     \ / FI(IB,0,IA) \     **
C**  XO | FO(IB,IA,1) |   |     U11 U12 U13 U14 | | FI(IB,1,IA) |     **
C**     | FO(IB,IA,2) | = |     U21 U22 U23 U24 | | FI(IB,2,IA) | XI  **
C**     | FO(IB,IA,3) |   |     U31 U32 U33 U34 | | FI(IB,3,IA) | YI  **
C**  YO \ FO(IB,IA,4) /   \     U41 U42 U43 U44 / \ FI(IB,4,IA) /     **
C**     +-------------------------------------------------------+     **
        UN1=(1.D0,0.D0)
        DO 180 IN=1,NOW/2
        JN=NOW-IN
        UN1=UN1*U11
        UNM=(1.D0,0.D0)
        DO 170 IM=1,NOW/2
        JM=NOW-IM
        UNM=UNM*UN1
C**     +-------------------------------------------------------+    **
C**     | XXO =  XXT+YYT = (U(0)+i*U(1))*XXI + (U(0)-i*U(1))*YYI |    **
C**     | YYO =  XXT-YYT = (U(0)-i*U(1))*XXI + (U(0)+i*U(1))*YYI |    **
C**     | XXT = (XXO+YYO)/2 =   U(0) * (XXI+YYI)                |    **
C**     | YYT = (XXO-YYO)/2 = i*U(1) * (XXI-YYI)                |    **
C**     +-------------------------------------------------------+    **
        DO 160 IB=0,NBF-1
          XXI=FI(IB,IN,IA)
          YYI=FI(IB,JN,IA)
          XT(0)=(XI(0)+YI(0))*U(0)
          XT(1)=(XI(1)+YI(1))*U(0)
          YT(0)=(YI(1)-XI(1))*U(1)
          YT(1)=(XI(0)-YI(0))*U(1)
          FO(IB,IA,IM)=FO(IB,IA,IM)+XXT+YYT
          FO(IB,IA,JM)=FO(IB,IA,JM)+XXT-YYT
  160   CONTINUE
  170   CONTINUE
```

```
  180   CONTINUE
  190 CONTINUE
      RETURN
C**   +--------------------------------------------------------------+    **
C**   | NOW=2 : Use dual nodes and in-place computation              |    **
C**   +--------------------------------------------------------------+    **
  200 IR=0
      AR1=(1.D0,0.D0)
      DO 290 IA=0,NAF-1
C**     +----------------------------------------------------------+    **
C**     | / FI(IB,0,IR) \   / 1  AR1 \ / FI(IB,0,IR) \             |    **
C**     | \ FI(IB,1,IR) / = \ 1 -AR1 / \ FI(IB,1,IR) /             |    **
C**     +----------------------------------------------------------+    **
        DO 280 IB=0,NBF-1
          FF=FI(IB,1,IR)*AR1
          FI(IB,1,IR)=FI(IB,0,IR)-FF
          FI(IB,0,IR)=FI(IB,0,IR)+FF
  280   CONTINUE
        IF(IA.EQ.NAF-1) GO TO 290
        IS=NAF/2
  285   IF(IS.LE.IR) THEN
          IR=IR-IS
          IS=IS/2
          GOTO 285
        ENDIF
        IR=IR+IS
        AR1=AR1*OMG
  290 CONTINUE
      GOTO 500
C**   +--------------------------------------------------------------+    **
C**   | NOW=4 : Use twiddle factor and in-place computation          |    **
C**   +--------------------------------------------------------------+    **
  400 IR=0
      AR1=(1.D0,0.D0)
      DO 490 IA=0,NAF-1
C**     +---------------------< for INV>=1 >---------------------+     **
C**     | / FO(IB,0,0,IR) \   / 1  1  1  1 \ / FI(IB,0,0,IR) \  |     **
C**     | | FO(IB,0,1,IR) |   | 1 -i -1  i | | FI(IB,1,0,IR) |  |     **
C**     | | FO(IB,1,0,IR) | = | 1 -1  1 -1 | | FI(IB,0,1,IR) |  |     **
C**     | \ FO(IB,1,1,IR) /   \ 1  i -1 -i / \ FI(IB,1,1,IR) /  |     **
C**     +-----------------------------------------------------------+     **
        DO 480 IB=0,NBF-1
          IF(IA.GT.0) THEN
            FI(IB,1,IR)=FI(IB,1,IR)*AR1
            FI(IB,2,IR)=FI(IB,2,IR)*AR2
            FI(IB,3,IR)=FI(IB,3,IR)*AR3
          ENDIF
          V0=FI(IB,0,IR)+FI(IB,2,IR)
          V1=FI(IB,1,IR)+FI(IB,3,IR)
          V2=FI(IB,0,IR)-FI(IB,2,IR)
          V3=FI(IB,1,IR)-FI(IB,3,IR)
          IF(INV.GT.0) THEN
            V3=CMPLX(AIMAG(V3),-REAL(V3))
          ELSE
            V3=CMPLX(-AIMAG(V3),REAL(V3))
          ENDIF
          FI(IB,0,IR)=V0+V1
          FI(IB,2,IR)=V2+V3
          FI(IB,1,IR)=V0-V1
          FI(IB,3,IR)=V2-V3
```

```
  480   CONTINUE
        IF(IA.EQ.NAF-1) GO TO 490
        IS=NAF/2
  485   IF(IS.LE.IR) THEN
          IR=IR-IS
          IS=IS/2
          GOTO 485
        ENDIF
        IR=IR+IS
        AR1=AR1*OMG
        AR2=AR1*AR1
        AR3=AR2*AR1
  490 CONTINUE
  500 IF((NBF/2)*2.EQ.NBF) RETURN
C**   +--------------------------------------------------------+    **
C**   | Bit-reversal operation after all base 4 + 2 transforms |    **
C**   +--------------------------------------------------------+    **
      IR=0
      DO 590 IA=0,NOW*NAF-2
        IF(IA.LT.IR) THEN
          DO 540 IB=0,NBF-1
            FF=FI(IB,IR,0)
            FI(IB,IR,0)=FI(IB,IA,0)
            FI(IB,IA,0)=FF
  540     CONTINUE
        ENDIF
        IS=NOW*NAF/2
  550   IF(IS.LE.IR) THEN
          IR=IR-IS
          IS=IS/2
          GOTO 550
        ENDIF
        IR=IR+IS
  590 CONTINUE
      RETURN
      END
*****************************************************************************
```

［表三］ 實數/共軛複數富利葉轉換程式；正/餘弦轉換程式

```
*****************************************************************************
      SUBROUTINE FFTR(A,B,N,INV)
C**   ==========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(0:N-1),B(0:N-1)
C**   ==========================================================    **
C**   FFT of real data for any even N                               **
C**   ----------------------------------------------------------    **
C**   f(0:1,0:N-1) ===> F(0:1,0:N-1) : f(1,I)=0, F(1,0)=F(1,N/2)=0    **
C**                F(0,N-I)=F(0,I), F(1,N-I)=-F(1,I),  for I=1,N/2    **
C**   ==========================================================    **
C**   For INV=+-1 : DIMENSION A(0:2*N-1)                            **
C**   Input  : A(2*I)=f(0,I), for I=0,N-1  (others unused)          **
C**   Output : A(2*I)=F(0,I), A(2*I+1)=F(1,I), for I=0,N-1          **
C**   ----------------------------------------------------------    **
C**   For INV=+-2 : DIMENSION A(0:N+1)                              **
C**   Input  : A(I) = f(0,I), for I=0,N-1                           **
C**   Output : A(2*I)=F(0,I), A(2*I+1)=F(1,I), for I=0,N/2          **
C**   ----------------------------------------------------------    **
C**   For INV=+-3 : DIMENSION A(0:N-1)                              **
```

```
C**   Input  : A(I) = f(0,I), for I=0,N-1                              **
C**   Output : A(2*I)=F(0,I), A(2*I+1)=F(1,I), for I=1,N/2-1           **
C**            A(0) = F(0,0), A(1)  =  F(0,N/2)                       **
C**   ============================================================    **
C**   Working : B(0:N-1) required for N .NE. 2**M only                **
C**   ============================================================    **
      IF(IABS(INV).NE.1) GOTO 40
      DO 20 I=0,N-1
   20 A(I)=A(2*I)
C**   ------------------------------------------------------------    **
   40 CALL FFTA(A,B,N/2,ISIGN(1,INV))
C**   ------------------------------------------------------------    **
      AN0R=A(0)-A(1)
      A(0)=A(0)+A(1)
      A(1)=0.0
C**   ------------------------------------------------------------    **
      ANG=-3.1415926535897932D0*ISIGN(2,INV)/N
      W0=DCOS(ANG)
      W1=DSIN(ANG)
      U0=W0
      U1=W1
      J=N-2
      DO 50 I=2,N/2,2
      H0=(A(I  )+A(J  ))*0.5
      G1=(A(J  )-A(I  ))*0.5
      G0=(A(I+1)+A(J+1))*0.5
      H1=(A(I+1)-A(J+1))*0.5
      P0=G0*U1+G1*U0
      P1=G1*U1-G0*U0
      A(I  )=H0-P1
      A(J  )=H0+P1
      A(I+1)=P0+H1
      A(J+1)=P0-H1
      T0=U0
      U0=T0*W0-U1*W1
      U1=T0*W1+U1*W0
   50 J=J-2
C**   ------------------------------------------------------------    **
      GOTO (60,90,55),IABS(INV)
   55 A(1)=AN0R
      RETURN
   60 J=2*N-2
      DO 80 I=2,N-1,2
      A(J)=A(I)
      A(J+1)=-A(I+1)
   80 J=J-2
   90 A(N)=AN0R
      A(N+1)=0.0
      RETURN
      END
*************************************************************************
      SUBROUTINE FFTC(A,B,N,INV)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(0:N-1),B(0:N-1)
C**   ============================================================    **
C**   FFT of conjugate data for any even N                            **
C**   ------------------------------------------------------------    **
C**   f(0:1,0:N-1) <=== F(0:1,0:N-1) : f(1,I)=0, F(1,0)=F(1,N/2)=0    **
C**                F(0,N-I)=F(0,I), F(1,N-I)=-F(1,I),  for I=1,N/2    **
```

```
C**   ============================================================    **
C**   For INV=+-1 : DIMENSION A(0:2*N-1)                              **
C**   Input  : A(2*I)=F(0,I), A(2*I+1)=F(1,I), for I=0,N/2            **
C**            note that  :   A(1)=0,  A(N+1)=0,  others unused       **
C**   Output : A(2*I)=f(0,I), A(2*I+1)=f(1,I)=0, for I=0,N-1          **
C**   ------------------------------------------------------------    **
C**   For INV=+-2 : DIMENSION A(0:N+1)                                **
C**   Input  : A(2*I)=F(0,I), A(2*I+1)=F(1,I), for I=0,N/2            **
C**            note that  :   A(1)=0,  A(N+1)=0                       **
C**   Output : A(I) = f(0,I), for I=0,N-1  (f(1,I)=0 not given)       **
C**   ------------------------------------------------------------    **
C**   For INV=+-3 : DIMENSION A(0:N-1)                                **
C**   Input  : A(2*I)=F(0,I), A(2*I+1)=F(1,I), for I=1,N/2-1          **
C**            A(0) = F(0,0), A(1)  =  F(0,N/2)                       **
C**   Output : A(I) = f(0,I), for I=0,N-1  (f(1,I)=0 not given)       **
C**   ============================================================    **
C**   Working : B(0:N-1) required for N .NE. 2**M only                **
C**   ============================================================    **
      AN0R=A(1)
      IF(IABS(INV).LT.3) AN0R=A(N)
      A(1)=A(0)-AN0R
      A(0)=A(0)+AN0R
C**   ------------------------------------------------------------    **
      ANG=3.1415926535897932D0*ISIGN(2,INV)/N
      W0=DCOS(ANG)
      W1=DSIN(ANG)
      U0=W0
      U1=W1
      J=N-2
      DO 50 I=2,N/2,2
      H0=A(I  )+A(J  )    !   Reverse from:    H0=(A(I  )+A(J  ))*0.5
      P1=A(J  )-A(I  )    !                    G1=(A(J  )-A(I  ))*0.5
      P0=A(I+1)+A(J+1)    !  1 <           /   G0=(A(I+1)+A(J+1))*0.5
      H1=A(I+1)-A(J+1)    !      \       /     H1=(A(I+1)-A(J+1))*0.5
      G0=P0*U1-P1*U0      !        \   /       P0=G0*U1+G1*U0
      G1=P1*U1+P0*U0      !  2 <-----X------   P1=G1*U1-G0*U0
      A(I  )=H0-G1        !        /   \       A(I  )=H0-P1
      A(J  )=H0+G1        !      /       \     A(J  )=H0+P1
      A(I+1)=G0+H1        !  3 <           \   A(I+1)=P0+H1
      A(J+1)=G0-H1        !                    A(J+1)=P0-H1
      T0=U0               !
      U0=T0*W0-U1*W1      !   Except *0.5 to be done outside FFTC
      U1=T0*W1+U1*W0
   50 J=J-2
C**   ------------------------------------------------------------    **
      CALL FFTA(A,B,N/2,ISIGN(1,INV))
C**   ------------------------------------------------------------    **
      IF(IABS(INV).NE.1) RETURN
      DO 80 I=N-1,0,-1
      A(2*I+1)=0.0
   80 A(2*I)=A(I)
      RETURN
      END
**********************************************************************
      SUBROUTINE FFTSC(A,B,N,ICS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(0:N-1),B(0:N-1)
C**   ============================================================    **
C**   Half-SIN or Half-COS transform of real data for even N          **
```

```
C**   ------------------------------------------------------------    **
C**   Input  : A(0:N-1)/A(0:N)/A(0:N-1) = AI() = real dat             **
C**   Output : A(0:N-1)/A(0:N)/A(0:N-1) = Asin()/Acos()/Ac()          **
C**   ------------------------------------------------------------    **
C**   ICS=+-1: Asin(K) = \sum_{J=0}^{N-1} AI(J)*SIN(K*J*PI/N)         **
C**   ICS=+-2: Acos(K) = \sum_{J=1}^{N-1} AI(J)*COS(K*J*PI/N)         **
C**                            + (AI(0)+AI(N)*COS(K*PI))*0.5          **
C**   ICS=+3 : Ac(K)   = \sum_{J=0}^{N-1} AI(J)*COS(K*(J+.5)*PI/N)    **
C**   ICS=-3 : Ac(K)   = \sum_{J=1}^{N-1} AI(J)*COS((K+.5)*J*PI/N)    **
C**                              + 0.5*AI(0)*COS((K+.5)*0*PI/N)       **
C**   ------------------------------------------------------------    **
C**   Working : B(0:N-1) required for N .NE. 2**M only                **
C**   ============================================================    **
      IF(IABS(ICS).GE.3) THEN
         CALL FFTCH(A,B,N,ICS)
         RETURN
       ENDIF
C**   ------------------------------------------------------------    **
      ANG=3.1415926535897932D0/N
      W0=DCOS(ANG)
      W1=DSIN(ANG)
C**   ------------------------------------------------------------    **
      U0=W0
      U1=W1
      IF(IABS(ICS).EQ.1) THEN
        A(0)=0.0
      ELSE
        A1  =(A(0)-A(N))*0.5
        A(0)=(A(0)+A(N))*0.5
      ENDIF
      DO 50 I=1,N/2
      IF(IABS(ICS).EQ.1) THEN
        R0=(A(I)-A(N-I))*0.5
        R1=(A(I)+A(N-I))*U1
        A(I)  =R1+R0
        A(N-I)=R1-R0
      ELSE
        R0=(A(I)+A(N-I))*0.5
        R1=(A(I)-A(N-I))*U1
        A1=(A(I)-A(N-I))*U0+A1
        A(I)  =R0-R1
        A(N-I)=R0+R1
      ENDIF
      T0=U0
      U0=T0*W0-U1*W1
      U1=T0*W1+U1*W0
   50 CONTINUE
C**   ------------------------------------------------------------    **
      CALL FFTR(A,B,N,-3)
      IF(IABS(ICS).NE.1) A(N)=A(1)
      A(1)=0.0
C**   ------------------------------------------------------------    **
      IF(IABS(ICS).EQ.1) THEN
        A1=-0.5*A(0)
        DO 60 I=0,N-2,2
        A1=A1+A(I)
        A(I)=A(I+1)
        A(I+1)=A1
   60   CONTINUE
      ELSE
```

```
        DO 70 I=1,N-1,2
        A1=A1+A(I)
        A(I)=A1
   70   CONTINUE
      ENDIF
      RETURN
      END
*********************************************************************
      SUBROUTINE FFTCH(A,B,N,ICS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(0:N-1),B(0:N-1)
C**   ============================================================    **
C**   Half-SIN or Half-COS transform of real data for even N          **
C**   ------------------------------------------------------------    **
C**   Input  : A(0:N-1) = AI(0:N-1) = real data                       **
C**   Output : A(0:N-1) = AC(0:N-1)                                   **
C**   ------------------------------------------------------------    **
C**   ICS=+3 : AC(K)   = \sum_{J=0}^{N-1} AI(J)*COS(K*(J+.5)*PI/N)    **
C**   ICS=-3 : AC(K)   = \sum_{J=1}^{N-1} AI(J)*COS((K+.5)*J*PI/N)    **
C**                                 + 0.5*AI(0)*COS((K+.5)*0*PI/N)    **
C**   ------------------------------------------------------------    **
C**   Working : B(0:N-1) required for N .NE. 2**M only                **
C**   ============================================================    **
      ANG=3.1415926535897932D0/N
      W0=DCOS(ANG)
      W1=DSIN(ANG)
C**   ------------------------------------------------------------    **
      IF(ICS.GT.0) THEN
      U0=DCOS(0.5*ANG)
      U1=DSIN(0.5*ANG)
      DO 20 I=0,N/2-1
      R0=(A(I)+A(N-1-I))*0.5
      R1=(A(I)-A(N-1-I))*U1
      A(I)    =R0-R1
      A(N-1-I)=R0+R1
      T0=U0
      U0=T0*W0-U1*W1
      U1=T0*W1+U1*W0
   20 CONTINUE
C**   ------------------------------------------------------------    **
      CALL FFTR(A,B,N,-3)
      AN=A(1)
      A(1)=0.0
C**   ------------------------------------------------------------    **
      AJ=-0.5*AN
      U0=W0
      U1=W1
      DO 30 I=N-2,0,-2
      DJ    =A(I)*U0+A(I+1)*U1
      A(I)  =A(I)*U1-A(I+1)*U0
      A(I+1)=AJ
      AJ=AJ-DJ
      T0=U0
      U0=T0*W0-U1*W1
      U1=T0*W1+U1*W0
   30 CONTINUE
      RETURN
C**   ------------------------------------------------------------    **
      ELSE
```

```
      AN=A(N-1)
      U0=W0
      U1=W1
      DO 70 I=N-2,2,-2
      DJ    =A(I+1)-A(I-1)
      A(I+1)=DJ*U1-A(I)*U0
      A(I)  =DJ*U0+A(I)*U1
      T0=U0
      U0=T0*W0-U1*W1
      U1=T0*W1+U1*W0
   70 CONTINUE
C**   ------------------------------------------------------------    **
      A(1)=-2.0*AN
      CALL FFTC(A,B,N,3)
C**   ------------------------------------------------------------    **
      U0=DCOS(0.5*ANG)
      U1=DSIN(0.5*ANG)
      DO 80 I=0,N/2-1
      R0=(A(I)+A(N-1-I))*0.25
      R1=(A(I)-A(N-1-I))*0.125/U1
      A(I)    =R0-R1
      A(N-1-I)=R0+R1
      T0=U0
      U0=T0*W0-U1*W1
      U1=T0*W1+U1*W0
   80 CONTINUE
      RETURN
      ENDIF
      END
*****************************************************************************
      SUBROUTINE FFTCS(A,B,N,ICS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(0:N-1),B(0:N-1)
C**   ============================================================    **
C**   SIN or COS transform of real data for even N                    **
C**   ------------------------------------------------------------    **
C**   Input  : A(0:N-1) = AI(0:N-1) = real data                       **
C**   Output : A(0:N-1) = Asin(0:N-1) or Acos(0:N-1)                  **
C**   ------------------------------------------------------------    **
C**   ICS=+-1: Asin(K) = \sum_{J=0}^{N-1} AI(J)*SIN(K*J*2PI/N)        **
C**   ICS=+-2: Acos(K) = \sum_{J=0}^{N-1} AI(J)*COS(K*J*2PI/N)        **
C**   ------------------------------------------------------------    **
C**   Working : B(0:N/2-1) required for N .NE. 2**M only              **
C**   ============================================================    **
      IF(IABS(ICS).EQ.1) THEN
        A(0)=0.0
        DO 20 I=1,N/2-1
          A(I)=A(I)-A(N-I)
   20   CONTINUE
      ELSE
        A(0)=A(0)+A(0)
        DO 30 I=1,N/2-1
          A(I)=A(I)+A(N-I)
   30   CONTINUE
        A(N/2)=A(N/2)+A(N/2)
      ENDIF
C**   ------------------------------------------------------------    **
      CALL FFTSC(A,B,N/2,ICS)
C**   ------------------------------------------------------------    **
```

```
      IF(IABS(ICS).EQ.1) THEN
        A(N/2)=0.0
        DO 40 I=1,N/2-1
          A(N-I)=-A(I)
   40   CONTINUE
      ELSE
        DO 60 I=1,N/2-1
          A(N-I)=A(I)
   60   CONTINUE
      ENDIF
      RETURN
      END
*======================================================================*
```

［表四］ 二維轉換副程式之範例

```
*======================================================================*
      SUBROUTINE FFT2D(A,N1,N2,INV1,INV2,W,NRA)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 A(0:NRA-1,0:N2-1),W(0:*)
C**   ================================================================    **
C**   INV1,INV2 = +1 : Fourior transform                                   **
C**   INV1,INV2 = -1 : Inverse transform                                   **
C**   ----------------------------------------------------------------    **
C**   Input & output : A(0:N1-1,0:N2-1) = complex data                     **
C**   ----------------------------------------------------------------    **
C**   W(0:N-1) : Working array                                             **
C**          N .GE. MAX(N1,N2+N2) for any N1  and any N2                   **
C**          N .GE. MAX(N1,   N2) for any N1  and N2=2**K                  **
C**          N .GE.        N2+N2  for N1=2**M and any N2                   **
C**          N .GE.           N2  for N1=2**M and N2=2**K                  **
C**   ----------------------------------------------------------------    **
      IF(N2.GT.1) THEN
        DO 40 I=0,N1-1
          DO 20 J=0,N2-1
            W(J)=A(I,J)
   20     CONTINUE
          CALL FFTA(W,W(N2),N2,INV2)
          DO 30 J=0,N2-1
            A(I,J)=W(J)
   30     CONTINUE
   40   CONTINUE
      ENDIF
C**   ----------------------------------------------------------------    **
      IF(N1.GT.1) THEN
        DO 50 J=0,N2-1
          CALL FFTA(A(0,J),W,N1,INV1)
   50   CONTINUE
      ENDIF
C**   ----------------------------------------------------------------    **
      RETURN
      END
*======================================================================*
      SUBROUTINE FFT2SC(A,N1,N2,ICS1,ICS2,W,NRA)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(0:NRA-1,0:N2-1),W(0:*)
C**   ================================================================    **
C**   2-D Half-SIN/COS transform                                           **
```

```
C**   ------------------------------------------------------------    **
C**   ICS1=+-1  Half-SIN transform for any N1 (NR=N1  )               **
C**   ICS1=+-2  Half-COS transform for any N1 (NR=N1+1)               **
C**   ICS1=+-3  Half-COS trans/inv for any N1 (NR=N1  )               **
C**   ICS2=+-1  Half-SIN transform for any N2 (NC=N2  )               **
C**   ICS2=+-2  Half-COS transform for any N2 (NC=N2+1)               **
C**   ICS2=+-3  Half-COS trans/inv for any N2 (NC=N2  )               **
C**   ------------------------------------------------------------    **
C**   Input & output : A(0:NR-1,0:NC-1) = real data                   **
C**   ------------------------------------------------------------    **
C**   W(0:N-1) : Working array                                        **
C**          N .GE. MAX(N1,N2+NC) for any N1  and any N2              **
C**          N .GE. MAX(N1,   NC) for any N1  and N2=2**J             **
C**          N .GE.        N2+NC  for N1=2**I and any N2              **
C**          N .GE.           NC  for N1=2**I and N2=2**J             **
C**   ============================================================    **
      NR=N1
      NC=N2
      IF(IABS(ICS1).EQ.2.AND.N1.GT.1) NR=N1+1
      IF(IABS(ICS2).EQ.2.AND.N2.GT.1) NC=N2+1
C**   ------------------------------------------------------------    **
      IF(N2.GT.1) THEN
        DO 40 I=0,NR-1
          DO 20 J=0,NC-1
            W(J)=A(I,J)
   20     CONTINUE
          CALL FFTSC(W,W(NC),N2,ICS2)
          DO 30 J=0,NC-1
            A(I,J)=W(J)
   30     CONTINUE
   40   CONTINUE
      ENDIF
C**   ------------------------------------------------------------    **
      IF(N1.GT.1) THEN
        DO 50 J=0,NC-1
          CALL FFTSC(A(0,J),W,N1,ICS1)
   50   CONTINUE
      ENDIF
C**   ------------------------------------------------------------    **
      RETURN
      END
**************************************************************************
```

［表五］ 以富利葉轉換解動力反應之副程式

```
**************************************************************************
      SUBROUTINE FFTTRR(W,E,YST,Y,DT,N,U,U0,V0)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(0:N-1),U(0:N+1)
C**   ============================================================    **
C**   This subroutine computes transient response of the equation     **
C**   Rel.A(I) + 2*E*W*V(I) + W*W*U(I) = W*W*YST*Y(I)                 **
C**   with initial conditions :   U(0)=U0,   V(0)=V0                  **
C**   and N=2**M by Fast Fourier Transform.  (V=dU/dT, A=dV/dT)       **
C**   ------------------------------------------------------------    **
C*I   W    = Natural frequence of the system                          **
C*I   E    = Damping ratio of the system                              **
C*I   YST  = Scale constant                                           **
C**        > 0 : for transient response                               **
```

```
C**         < 0 : for steady-state response                              **
C*I    Y(I) = Input force divided by W*W*YST at T=I*DT                   **
C*I    DT   = Time interval                                              **
C*I    N    = Number of data points                                      **
C*O    U(I) = Output response at T=I*DT                                  **
C*I    U0   = Initial displacement at T=0                                **
C*I    V0   = Initial velocity at T=0                                    **
C**    ------------------------------------------------------------      **
C**    Array U(N+2) may be equivalence to Y(N).                          **
C**    If N .NE. 2**Integer then array U() must has length 2*N+2         **
C**    ============================================================      **
       DO 10 I=0,N-1
    10 U(I)=Y(I)*YST*W*W
C**    +----------------------------------------------------------+      **
C**    | Perform Fourier Transform by FFT(Real)                   |      **
C**    +----------------------------------------------------------+      **
       CALL FFTR(U,U(N+2),N,2)
C**    +----------------------------------------------------------+      **
C**    | Multiple by transfer function H(I*DW) and divided by N   |      **
C**    +----------------------------------------------------------+      **
       DW=6.2831853071795864D0/(DT*N)
       DO 30 I=0,N/2
       H1= W**2-(I*DW)**2
       H2=-2.*E*W*DW*I
       HH=(H1*H1+H2*H2)*N
       U1=U(2*I)
       U(2*I  )=(U1*H1-U(2*I+1)*H2)/HH
    30 U(2*I+1)=(U1*H2+U(2*I+1)*H1)/HH
       U(N+1)=0.0
C**    +----------------------------------------------------------+      **
C**    | For YST<0 : Compute steady-state response by Inverse FFT |      **
C**    +----------------------------------------------------------+      **
       IF(YST.LT.0) THEN
         CALL FFTC(U,U(N+2),N,-2)
         RETURN
       ENDIF
C**    +----------------------------------------------------------+      **
C**    | For YST>0 : Get initial value of steady-state response   |      **
C**    +----------------------------------------------------------+      **
       UO=0.0
       VO=0.0
       DO 40 I=1,N/2-1
       UO=UO+U(2*I)
    40 VO=VO-U(2*I+1)*DW*I
       UO=UO+UO+U(0)+U(N)
       VO=VO+VO
C**    +----------------------------------------------------------+      **
C**    | Then : Compute the steady-state response by Inverse FFT  |      **
C**    +----------------------------------------------------------+      **
       CALL FFTC(U,U(N+2),N,-2)
C**    +----------------------------------------------------------+      **
C**    | And : Adjust I.C. to U0 and V0 for transient response    |      **
C**    +----------------------------------------------------------+      **
       UU=U0-UO
       VV=(V0-VO)/W
       D=DSQRT(1.0-E*E)
       O=DATAN(E/D)
       EDT=1.0/D
       WDDT=W*D*DT
       EWDT=DEXP(-E*W*DT)
```

```
C**   ------------------------------------------------------------    **
      DO 50 I=0,N-1
      WDT=WDDT*I
      U(I)=U(I)+EDT*(UU*DCOS(WDT-0)+VV*DSIN(WDT))
   50 EDT=EDT*EWDT
      RETURN
      END
*************************************************************************
```

習題

1. 試証式(??)可分解成下列五個子轉換階段及基底反向易位之處理：

$$F_1(l1,l2,l3,l4,k5) = \sum_{l5=0}^{N5-1} f\ (l1,l2,l3,l4,l5) * W^{(l1,l2,l3,l4,l5)*(k5,0,0,0,0)}$$

$$F_2(l1,l2,l3,k4,k5) = \sum_{l4=0}^{N4-1} F_1(l1,l2,l3,l4,k5) * W^{(l1,l2,l3,l4,0)*(0,k4,0,0,0)}$$

$$F_3(l1,l2,k3,k4,k5) = \sum_{l3=0}^{N3-1} F_2(l1,l2,l3,k4,k5) * W^{(l1,l2,l3,0,0)*(0,0,k3,0,0)}$$

$$F_4(l1,k2,k3,k4,k5) = \sum_{l2=0}^{N2-1} F_3(l1,l2,k3,k4,k5) * W^{(l1,l2,0,0,0)*(0,0,0,k2,0)}$$

$$F_5(k1,k2,k3,k4,k5) = \sum_{l1=0}^{N1-1} F_4(l1,k2,k3,k4,k5) * W^{(l1,0,0,0,0)*(0,0,0,0,k1)}$$

$$F\ (k5,k4,k3,k2,k1) = F_5(k1,k2,k3,k4,k5)$$

或分解成下列五個子轉換階段，以免除基底反向易位之處理。

$$F_1(l1,l2,l3,l4,k5) = \sum_{l5=0}^{N5-1} f\ (l1,l2,l3,l4,l5) * W^{(l1,l2,l3,l4,l5)*(k5,0,0,0,0)}$$

$$F_2(l1,l2,l3,k5,k4) = \sum_{l4=0}^{N4-1} F_1(l1,l2,l3,l4,k5) * W^{(l1,l2,l3,l4,0)*(0,k4,0,0,0)}$$

$$F_3(l1,l2,k5,k4,k3) = \sum_{l3=0}^{N3-1} F_2(l1,l2,l3,k5,k4) * W^{(l1,l2,l3,0,0)*(0,0,k3,0,0)}$$

$$F_4(l1,k5,k4,k3,k2) = \sum_{l2=0}^{N2-1} F_3(l1,l2,k5,k4,k3) * W^{(l1,l2,0,0,0)*(0,0,0,k2,0)}$$

$$F\ (k5,k4,k3,k2,k1) = \sum_{l1=0}^{N1-1} F_4(l1,k5,k4,k3,k2) * W^{(l1,0,0,0,0)*(0,0,0,0,k1)}$$

2. 試按上題之分段方式改寫副程式$FFTA$以計算富利葉轉換。

3. 式(17.65)與習題一中之基底反向易位均於子轉換完成後再處理，但亦可以先做基底反向易位之後再做子轉換。試寫出對應之轉換式。

4. 試按上題之方式改寫副程式$FFTA$以計算富利葉轉換。

5. 試以富利葉轉換法求第十二章習題1或習題2之常微分方程式之數值解。並與該章之結果(如有的話)相互比較。

參考文獻

1. Cooley, J. W. and Tukey, J. W., "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol.19, No.90, 1965.

2. Lin, T. W. and Wang, S. J., "The Expansion and the Numerical Evaluation of Duhamel's Integral", 4th Intl. Conf. on Applied Numerical Modelling, Taiwan, Tainan, 1984.

3. Biggs, J. M., *Introduction to Structural Dynamic,* McGraw-Hill Book Co., New York, 1964.

4. Brigham, E. O., *The Fast Fourier Transform,* Englewood Cliffs, New Jersey, Prentice-Hall, 1974.

5. 林聰悟，"快速富利葉轉換在動力分析上之應用及其計算原理"，結構工程，第三卷，第四期，1988。

第十八章

資料排序法

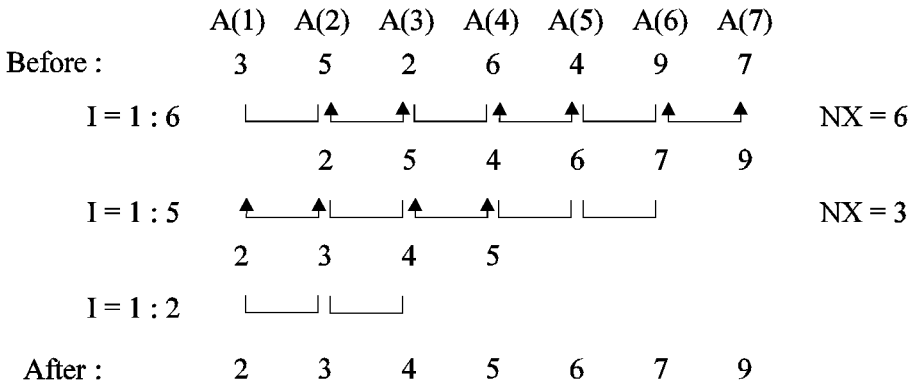
18.1 前言

排序雖然在工程分析上的應用不是很多，但是在商業上或一般性的應用卻是一項非常有用的工具。排序的方法很多，有些方法之運算量與排序關鍵值之數目(設為N)之平方成正比，程式比較簡短，適用於N較小之情形；有些則與$N\log_2 N$成正比，程式略為複雜，當N值較大時則須使用這些方法。本章將介紹四種排序法：氣泡排序法 (Bubble sort)，震動排序法 (Shaker sort)，堆積排序法 (Heapsort)及快速排序法 (Quicksort)。前二者為慢的方法；後二者為快的方法。本章所有排序說明或程式均以將關鍵值$(A(I), I = 1, N)$，簡寫成$A(1{:}N)$，排成由小至 大之順序為例。本章亦提供：指標排序、穩定排序及與關鍵值之資料類型無關 之排序副程式。並對堆積排序法與快速排序法之運算量做一比較。

18.2 氣泡排序法

氣泡排序法為依$I = 1, 2, \ldots, N-1$之次序，比較相鄰二關鍵值$A(I)$與$A(I+1)$，當二者順序不對時則予對調。最後$A(N)$必然為最大值。第二回再依$I = 1, 2, \ldots, N-2$之次序，比較相鄰二關鍵值，但注意比較範圍已改為$I = 1,\ N-2$。依此類推並逐回減少比較範圍，即可將關鍵值按順序排好(詳圖一)。注意每一回合之比較範圍可不止減一，而可減至前回比較時最後一次對調的地方之前。例如：若前回合最後一次對調的位置為NX，即$A(NX)$與$A(NX+1)$為最後一次對調，則$A(NX+1)$至$A(N)$

之順序已正確，且這些關鍵值均比$A(1)$至$A(NX)$大，因此下一回合僅須對$A(1)$至$A(NX)$之部分排序即可，故比較範圍改為$I = 1, NX - 1$。詳如副程式$BUBBLE$所示。



圖一 氣泡排序法之比較過程

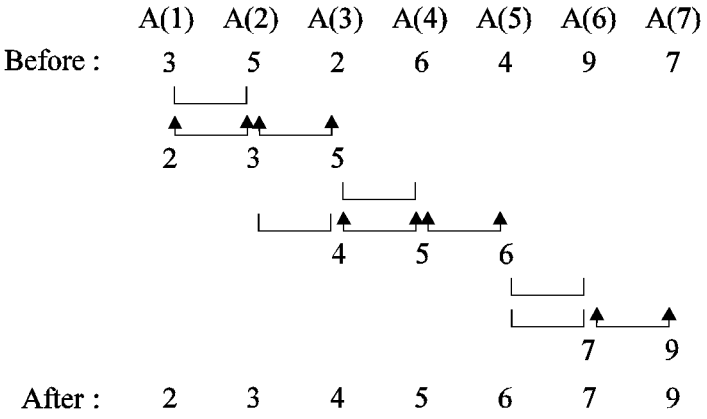
氣泡排序法在最壞的情況下(原始資料反向排列)需比較並對調$N*(N-1)/2$次；但在最好的情況下(原始資料按序排列)則僅需比較$(N-1)$次。

［表一］ 氣泡排序法副程式

```
**************************************************************************
      SUBROUTINE BUBBLE(A,N)
C**   ========================================================    **
      DIMENSION A(N)
C**   ========================================================    **
C**   Bubble sort : After sorting  A(I).LE.A(I+1)                 **
C**   ========================================================    **
      NX=N
   20 M=NX-1
      DO 50 I=1,M
      IF(A(I).LE.A(I+1)) GO TO 50
      AT=A(I)
      A(I)=A(I+1)
      A(I+1)=AT
      NX=I
   50 CONTINUE
C**   +------------------------------------------------------+    **
C**   | Return if no exchange during do loop 50 (NX=M+1)     |    **
C**   +------------------------------------------------------+    **
      IF(NX.LE.M) GO TO 20
      RETURN
      END
**************************************************************************
```

18.3 震動排序法

震動排序法為當氣泡排序法逐次往後比較相鄰二關鍵值時，若須對調二關鍵值，則於對調後繼續往前比較相鄰二關鍵值並對調順序不對者，直至順序正確不必對調為止，相當於往前也做氣泡排序。例如圖二中，比較$A(4)=6$與$A(5)=4$後，須對調為$A(4)=4$與$A(5)=6$，因此繼續往前



圖二 震動排序法之比較過程

比較$A(3)=5$與$A(4)=4$，對調為$A(3)=4$與$A(4)=5$，再比較$A(2)=3$與$A(3)=4$，其順序正確不必對調而停止往前的比較，並回復往後之$A(5)$與$A(6)$的比較。注意往前的比較及對調使前段(指已做過往後比較之範圍)關鍵值完全按順序排好，例如在比較$A(5)$與$A(6)$時，$A(1:4)=(2,3,4,5)$已照順序排好，故僅須做一回往後之比較即可完成排序工作。詳副程式$SHAKER$。因資料時而往後移動時而往前移動，故稱為震動排序法。

［表二］ 震動排序法副程式

```
************************************************************************
      SUBROUTINE SHAKER(A,N)
C**   ============================================================    **
      DIMENSION A(N)
C**   ============================================================    **
C**   Shaker sort : After sorting  A(I).LE.A(I+1)                     **
C**   ============================================================    **
      DO 50 I=2,N
      IF(A(I-1).LE.A(I)) GO TO 50
      K=I
      AX=A(K)
   30 A(K)=A(K-1)
      K=K-1
```

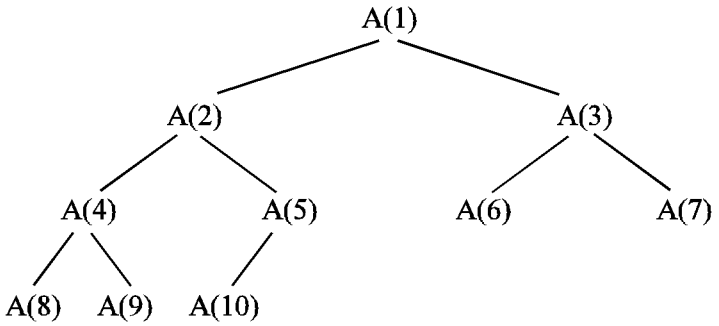
```
      IF(K.LE.1) GO TO 40
      IF(A(K-1).GT.AX) GO TO 30
   40 A(K)=AX
   50 CONTINUE
      RETURN
      END
**************************************************************************
```

18.4 堆積排序法

若有連續$M-K+1$個關鍵值$A(K:M)$，滿足下列關係(參考圖三)：

$$A(I) \geq A(2*I) \qquad \text{且} \qquad A(I) \geq A(2*I+1) \tag{18.1}$$

則稱$A(K:M)$為堆積。當$K=1$時堆積僅一組含所有M個值。當$K=2,\ M=7$時堆積分為二組，各含3個值。當$K=2,\ M=10$時堆積亦分為二組，一組含3個值$(A(3), A(6), A(7))$，另一組含其餘6個值。



圖三 堆積內連線二值須滿足式(18.1)

堆積排序法的一個主要基本運算為：加入一個關鍵值AX到$A(K+1:M)$之堆積中，使形成新堆積$A(K:M)$。詳細步驟如下(見圖四)：

步驟一：先令$I=K$。續做步驟二。

步驟二：看是否AX可置於$A(I)$？可以的話則令$A(I)=AX$而成新堆積，其條件為：

(1)當$2*I>M$時。

(2)當$2*I=M$時：$AX \geq A(2*I)$。

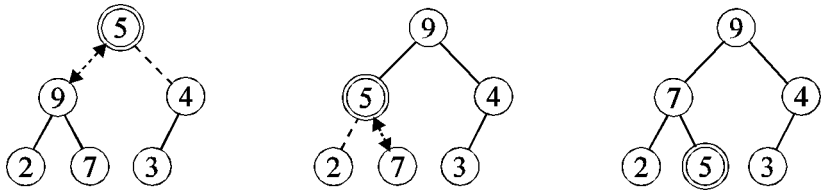
(3)當$2*I<M$時：$AX \geq \max(A(2*I), A(2*I+1))$。

上述運算可先令$J=2*I$，而當$A(2*I)<A(2*I+1)$時則改令$J=2*I+1$。

再做$AX \geq A(J)$之比較，成立的話則令$A(I) = AX$，即可形成新堆積而結束運算；否則續做步驟三。

步驟三:令$A(I) = A(J),\ I = J$回步驟二。亦即將$A(I)$之下層中較大的$A(J)$上移一層，AX的可能存放位置I則下移一層至J，再回步驟二繼續比較AX是否可置於$A(J)$。

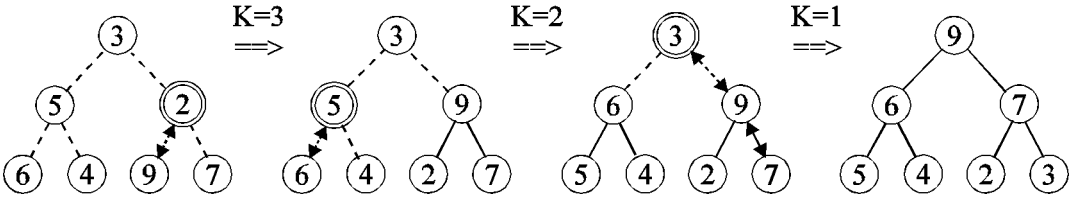
以上過程詳圖四與副程式$HEAP$內之$SHIFT$。



圖四 加值至堆積之推移過程

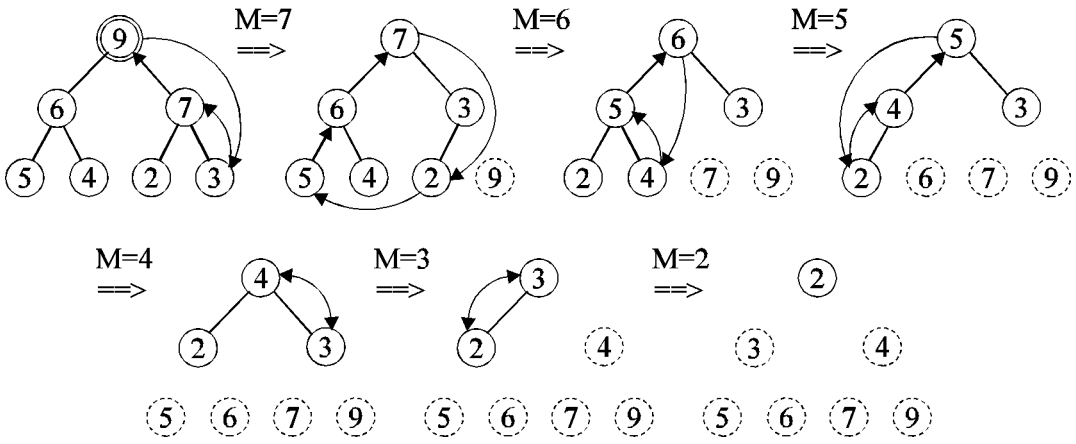
以堆積做排序可分為下列二階段：

(1)階段一：為聯合$(A(I), I = 1, N)$使成堆積。做法為：若已有堆積$(A(I), I = K + 1, N)$，可利用前段所述方法加入AX，以形成新堆積$(A(I), I = K, N)$，令$K = N/2, 1, -1$重複此過程即可完成堆積$(A(I), I = 1, N)$。此處K自$N/2$開始係因$(A(I), I = N/2 + 1, N)$均無下層資料，即$2 * I > N$，故已形成堆積。注意圖五中實線相連之部分已形成堆積。



圖五 階段一形成堆積之過程

(2)階段二：為自堆積$(A(I), I = 1, M)$中，取其最大值$A(1)$置於排序後之正確位置M，$A(M)$則予取出令為AX，亦按前段所述方法加到堆積$(A(I), I = 2, M - 1)$中形成新堆積$(A(I), I = 1, M - 1)$，令$M = N, 2, -1$重複上述過程即可完成排序工作。注意圖六中虛圓之部分已排好順序。



圖六 階段二堆積排序之過程

［表三］ 堆積排序法副程式

```
************************************************************************
      SUBROUTINE HEAP(A,N)
C**   ================================================================    **
      DIMENSION A(N)
C**   ================================================================    **
C**   Heap sort : After sorting  A(I).LE.A(I+1)                           **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Form heap (A(I),I=K,N) for K=N/2 ... 1                       |    **
C**   +--------------------------------------------------------------+    **
      DO 10 K=N/2,1,-1
      AX=A(K)
      CALL SHIFT(A,K,N,AX)
   10 CONTINUE
C**   +--------------------------------------------------------------+    **
C**   | Put max of A(1:M) in the correct position M                   |    **
C**   | Form heap (A(I),I=1,M-1) for M=N ... 2                       |    **
C**   +--------------------------------------------------------------+    **
      DO 20 M=N,2,-1
      AX=A(M)
      A(M)=A(1)
      CALL SHIFT(A,1,M-1,AX)
   20 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE SHIFT(A,K,M,AX)
C**   ================================================================    **
      DIMENSION A(M)
C**   ================================================================    **
C**   Given AX and heap A(K+1:M)  where  K .LE. M                         **
C**   Shift AX down to form new heap A(K:M)                               **
C**   A: A(I).GE.A(2*I) .AND. A(I).GE.A(2*I+1)                            **
C**   ================================================================    **
      I=K
   60 J=I+I
      IF(J.LE.M) THEN
        IF(J.LT.M) THEN
```

```
          IF(A(J+1).GT.A(J)) J=J+1
        ENDIF
        IF(A(J).GT.AX) THEN
C**       +-------------------------------------------------+    **
C**       | Shift AX down to position J                     |    **
C**       +-------------------------------------------------+    **
          A(I)=A(J)
          I=J
          GO TO 60
        ENDIF
      ENDIF
C**   +-----------------------------------------------------+    **
C**   | No more shift down : I is the correct position for AX    **
C**   +-----------------------------------------------------+    **
      A(I)=AX
      RETURN
      END
************************************************************************
      SUBROUTINE HEAPB(A,B,N)
C**   =============================================================    **
      DIMENSION A(N),B(N)
C**   =============================================================    **
C**   Heap sort : After sorting  A(I).LE.A(I+1)                        **
C**   =============================================================    **
C**   +-----------------------------------------------------+    **
C**   | Form heap (A(I),I=K,N) for K=N/2 ... 1              |    **
C**   +-----------------------------------------------------+    **
      DO 10 K=N/2,1,-1
      AX=A(K)
      BX=B(K)
      CALL SHIFTB(A,B,K,N,AX,BX)
   10 CONTINUE
C**   +-----------------------------------------------------+    **
C**   | Put max of A(1:M) in the correct position M          |    **
C**   | Form heap (A(I),I=1,M-1) for M=N ... 2              |    **
C**   +-----------------------------------------------------+    **
      DO 20 M=N,2,-1
      AX=A(M)
      BX=B(M)
      A(M)=A(1)
      B(M)=B(1)
      CALL SHIFTB(A,B,1,M-1,AX,BX)
   20 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE SHIFTB(A,B,K,M,AX,BX)
C**   =============================================================    **
      DIMENSION A(M),B(M)
C**   =============================================================    **
C**   Given AX and heap A(K+1:M)  where  K .LE. M                      **
C**   Shift AX down to form new heap A(K:M)                            **
C**   A: A(I).GE.A(2*I) .AND. A(I).GE.A(2*I+1)                         **
C**   =============================================================    **
      I=K
   60 J=I+I
      IF(J.LE.M) THEN
        IF(J.LT.M) THEN
          IF(A(J+1).GT.A(J)) J=J+1
        ENDIF
```

```
        IF(A(J).GT.AX) THEN
C**       +-------------------------------------------------------+    **
C**       | Shift AX down to position J                           |    **
C**       +-------------------------------------------------------+    **
          A(I)=A(J)
          B(I)=B(J)
          I=J
          GO TO 60
        ENDIF
      ENDIF
C**   +-----------------------------------------------------------+    **
C**   | No more shift down : I is the correct position for AX     |    **
C**   +-----------------------------------------------------------+    **
      A(I)=AX
      B(I)=BX
      RETURN
      END
*************************************************************************
```

18.5 快速排序法

快速排序法是目前公認最快的排序法。該法之原理為：將所有關鍵值分成前後二段，使前段之關鍵值均不大於後段之關鍵值，如此前後段即可單獨做排序運算，當二段均排序完成則所有關鍵值即排序完成。至於每段之排序又可用同樣的方法再分成前後二段單獨排序。依此類推，即可完成排序。因此快速排序法之主要運算為將$A(I)$至$A(M)$間之關鍵值$A(I:M)$分成前段$A(I:K)$及後段$A(J:M)$。做法為自$A(I:M)$間任選一關鍵值作為比較值AX，前段自$A(I)$起逐一往後與AX比較(亦即令$J=I,I+1,\ldots$，比較$A(J)$與AX)，後段亦自$A(M)$起逐一往前與AX比較(亦即令$K=M,M-1,\ldots$，比較$A(K)$與AX)，前段小於AX者或後段大於AX者均留在原處，否則即將該二值對調。重複此過程一直到前段關鍵值在後段關鍵值之後(即$J>K$)為止即完成分段之工作。前述比較用前段小於AX而不用小於等於AX可以免掉$J>M$之比較；同理，後段可免掉$K<I$之比較；但會增加等於AX之關鍵值之搬移運算。另注意$A(K)$至$A(J)$間亦可能有等於AX的關鍵值存在，不過此中間段不必再經排列，仍然只須再做前後二段之排序即可。詳見副程式$QUICK$中標注為$PARTITION(A,I,M,J,K,AX)$之部分程式。

因分段後不能二段同時做排序，必須保留其中一段，繼續做另一段之排序，當然又須保留一分段，繼續做另一分段，如此重複下去必須保留許多分段。因此須考慮減少保留之分段數。注意只要每次均保留大的

分段則保留段數不大於$\log_2 N$；但如保留小的分段，而不幸分段長均為一則保留段數等於$N-1$。因此分段後須做二分段長之比較以決定保留那一個較長分段。

另外在副程式$QUICK$中之指令80做了下列二項判斷：

(1)當欲排序之段長小於11時就改用震動排序法而不再繼續分段。因為對於少量關鍵值之排序，分段排序反而效率不好。至於其數量(此處之11)之決定可用試誤方式為之，該數可能會因使用不同之電腦或編譯器而有不同之較佳值。

(2)當$I=1$時即使段長小於11亦不用震動排序法。因為此處之震動排序法，當往前比較二關鍵值以決定是否繼續往前移動時，使用了『哨兵』的技巧。使得環路95只用了必須做的$A(K-1)>AX$之比較，而免掉$K\le I$之比較，因此可減少環路的運算時間。在分段排序的過程中，除最前面的一個分段(即$I=1$的分段)以外，每一分段的前面均至少有一個關鍵值(在該分段之前一分段內)小於或等於AX，該值稱為哨兵，因該值可以使$A(K-1)>AX$之條件不滿足而終止環路95(等於被哨兵擋住去路)，故即使免掉$K\le I$之比較亦不致使K繼續往前減小而越過I值。

注意指令10以後十數行指令係用以將$A(I), A((I+M)/2), A(M)$三個關鍵值按序排列，並取其中間值做為AX。其目的在使AX值更接近分段之中間關鍵值，亦即使二分段長接近相等，以使分段排序更有效。因為分段排序最不利的情況為：當每次分段後之小分段長為一時，比較次數與N平方成正比。另注意首尾二個關鍵值與AX之順序已正確，因此環路40及50之比較並非從首尾關鍵值開始。

前四節介紹之四種排序法在電腦執行的時間比例，以$N=8192$為例，按氣泡排序法，震動排序法，堆積排序法，快速排序法之順序約為385:165:1.75:1。

［表四］ 快速排序法副程式

```
**********************************************************************
      SUBROUTINE QUICK(A,N)
C**   ============================================================    **
      DIMENSION A(N),IL(20),IU(20)
C**   ============================================================    **
C**   Quick sort : after sorting  A(I).LE.A(I+1)                      **
C**   IL(NN),IU(NN) permit sorting up to N=2**(NN+1)-1                **
C**   ============================================================    **
      I=1
      M=N
      NN=1
C**   ------------------------------------------------------------    **
```

```
   10 IF(I.GE.M) GO TO 70
C**   +-------------------------------------------------------+    **
C**   | Sorting A(I),A(IX),A(M)                               |    **
C**   | And get AX=MID(A(I),A(IX),A(M))                       |    **
C**   +-------------------------------------------------------+    **
      IX=(I+M)/2
      AX=A(IX)
      IF(A(I).GT.AX) THEN
        A(IX)=A(I)
        A(I)=AX
        AX=A(IX)
      ENDIF
      IF(AX.GT.A(M)) THEN
        A(IX)=A(M)
        A(M)=AX
        AX=A(IX)
        IF(A(I).GT.AX) THEN
          A(IX)=A(I)
          A(I)=AX
          AX=A(IX)
        ENDIF
      ENDIF
C**   ============================================================    **
C**   SUBROUTINE PARTITION(A,I,M,J,K,AX)
C**   ============================================================    **
C**   Sorting A(I:M) by partition into A(I:K) and A(J:M)              **
C**   ============================================================    **
C**   Such that  I <= K < J <= M  provided I < M                      **
C**   A(x) <= AX  for  I <= x <= K                                    **
C**   A(x)  = AX  for  K <  x <  J                                    **
C**   A(x) >= AX  for  J <= x <= M                                    **
C**   ============================================================    **
C**   +-------------------------------------------------------+    **
C**   | Test right part for A(K)>AX & left part for A(J)<AX    |    **
C**   +-------------------------------------------------------+    **
      J=I
      K=M
   40 K=K-1
      IF(A(K).GT.AX) GO TO 40
   50 J=J+1
      IF(AX.GT.A(J)) GO TO 50
C**   ------------------------------------------------------------    **
      IF(J.LE.K) THEN
        AT=A(J)
        A(J)=A(K)
        A(K)=AT
        GO TO 40
      ENDIF
C**   +-------------------------------------------------------+    **
C**   | End of partitioning : Find longer partition to save   |    **
C**   +-------------------------------------------------------+    **
      IF(K-I.GT.M-J) THEN
C**     +-----------------------------------------------------+    **
C**     | Save  left partition A(I:K) then sort A(J:M)        |    **
C**     +-----------------------------------------------------+    **
        IL(NN)=I
        IU(NN)=K
        NN=NN+1
        I=J
      ELSE
```

```
C**     +-------------------------------------------------------+    **
C**     | Save right partition A(J:M) then sort A(I:K)          |    **
C**     +-------------------------------------------------------+    **
        IL(NN)=J
        IU(NN)=M
        NN=NN+1
        M=K
      ENDIF
      GO TO 80
C**   +-------------------------------------------------------+    **
C**   | Get a saved partition A(I:M) for sorting              |    **
C**   +-------------------------------------------------------+    **
   70 NN=NN-1
      IF(NN.EQ.0) RETURN
      I=IL(NN)
      M=IU(NN)
C**   +-------------------------------------------------------+    **
C**   | Use quick sort for long key array                     |    **
C**   +-------------------------------------------------------+    **
   80 IF(M-I.GE.11.OR.I.EQ.1) GO TO 10
C**   +-------------------------------------------------------+    **
C**   | Use shaker sort for short key array                   |    **
C**   +-------------------------------------------------------+    **
   90 I=I+1
      IF(I.GT.M) GO TO 70
      IF(A(I-1).LE.A(I)) GO TO 90
      K=I
      AX=A(K)
   95 A(K)=A(K-1)
      K=K-1
      IF(A(K-1).GT.AX) GO TO 95
C**   ---------------------------------------------------------    **
      A(K)=AX
      GO TO 90
      END
*************************************************************************
```

18.6 指標排序

前面所介紹的排序程式均直接將原來關鍵值搬動位置。當關鍵值移動位置時，其相關資料必須跟著搬動。例如按學號排序時，其相關資料如姓名、地址等亦須跟著關鍵值搬動。如以副程式$HEAP$為例，可更改如$HEAPB$所示。但這樣做除了使程式略為繁複外，跟隨移動的資料亦做了許多不必要的搬動。因為，如果已經知道每個資料的最後排列順序時，每個資料最多只需搬動一次即可按順序排好。但排序過程中，每個資料將不只搬動一次才會到達其最後正確位置。欲避免這些缺點，可改用指標排序。

所謂指標排序為排序時不更動關鍵值之位置，但排序之結果為產生一

組指標$(L(I), I = 1, N)$，使$L(I)$表示排序後第I個關鍵值為原來之第$L(I)$個關鍵值。亦即原關鍵值按順序排列應為：$A(L(1)), A(L(2)), \ldots, A(L(N))$。利用指標排序可以很方便地按數個不同關鍵值做排序而分別產生不同之順序指標。這些指標可以同時存在以備不同用途之使用。

前面所介紹的排序副程式均可改為指標排序，方法如下：在未排序前先令$L(I) = I,\ I = 1, N$。排序中，序列之第I個關鍵值須改為$A(L(I))$，比較第I個與第J個關鍵值則改為比較$A(L(I))$與$A(L(J))$。當二個關鍵值順序不對而須要對調時，只須將$L(I)$與$L(J)$對調即可。因$L(I)$與$L(J)$對調以後，新的$A(L(I))$與$A(L(J))$的順序就對了。

副程式$SHAKEL$與$HEAPL$為將$SHAKER$與$HEAP$分別改為指標排序之結果，其他副程式則保留作為習題，均可仿此做類似更改。

指標排序後，如果需要將各項資料按關鍵值之順序重新排列，可對每項資料分別呼叫副程式$SEQUNS$，即可照排序後所得之指標排好。當然亦可在副程式$SEQUNS$中直接增列所有各項資料之搬移運算。注意副程式$SEQUNS$不必使用暫時位置，但效率較差，約需二倍時間。但如利用增列搬移運算之方式，則效率反而會較好，因每增列一項僅增加不到四分之一的運算時間。

指標排序副程式較非指標排序副程式的運算時間約增加百分之25。

［表五］ 指標排序副程式

```
************************************************************************
      SUBROUTINE SHAKEL(A,L,N)
C**   ============================================================    **
      DIMENSION A(N),L(N)
C**   ============================================================    **
C**   Shaker sort : After sorting  A(L(I)).LE.A(L(I+1))               **
C**   ============================================================    **
      DO 10 I=1,N
   10 L(I)=I
C**   ------------------------------------------------------------    **
      DO 50 I=2,N
      IF(A(L(I-1)).LE.A(L(I))) GO TO 50
      K=I
      LX=L(K)
   30 L(K)=L(K-1)
      K=K-1
      IF(K.LE.1) GO TO 40
      IF(A(L(K-1)).GT.A(LX)) GO TO 30
   40 L(K)=LX
   50 CONTINUE
      RETURN
      END
************************************************************************
```

```
      SUBROUTINE HEAPL(A,L,N)
C**   ================================================================    **
      DIMENSION A(N),L(N)
C**   ================================================================    **
C**   Heap sort : After sorting  A(L(I)).LE.A(L(I+1))                     **
C**   ================================================================    **
      DO 5 I=1,N
    5 L(I)=I
C**   +--------------------------------------------------------------+    **
C**   | Form heap (A(L(I)),I=K,N) for K=N/2 ... 1                     |    **
C**   +--------------------------------------------------------------+    **
      DO 10 K=N/2,1,-1
      LX=L(K)
      CALL SHIFTL(A,L,K,N,LX)
   10 CONTINUE
C**   +--------------------------------------------------------------+    **
C**   | Put max of A(L(1:M)) in the correct position M                |    **
C**   | Form heap (A(L(I)),I=1,M-1) for M=N ... 2                     |    **
C**   +--------------------------------------------------------------+    **
      DO 20 M=N,2,-1
      LX=L(M)
      L(M)=L(1)
      CALL SHIFTL(A,L,1,M-1,LX)
   20 CONTINUE
      RETURN
      END
*************************************************************************
      SUBROUTINE SHIFTL(A,L,K,M,LX)
C**   ================================================================    **
      DIMENSION A(M),L(M)
C**   ================================================================    **
C**   Given LX and heap A(L(K+1:M))  where  K .LE. M                      **
C**   Shift A(LX) down to form new heap A(L(K:M))                         **
C**   A: A(L(I)).GE.A(L(2*I)) .AND. A(L(I)).GE.A(L(2*I+1))                **
C**   ================================================================    **
      I=K
   60 J=I+I
      IF(J.LE.M) THEN
        IF(J.LT.M) THEN
          IF(A(L(J+1)).GT.A(L(J))) J=J+1
        ENDIF
        IF(A(L(J)).GT.A(LX)) THEN
C**       +--------------------------------------------------------+    **
C**       | Shift A(LX) down to position J                         |    **
C**       +--------------------------------------------------------+    **
          L(I)=L(J)
          I=J
          GO TO 60
        ENDIF
      ENDIF
C**   +--------------------------------------------------------------+    **
C**   | No more shift down : I is the correct position for A(LX)     |    **
C**   +--------------------------------------------------------------+    **
      L(I)=LX
      RETURN
      END
*************************************************************************
```

18.7 與關鍵值之資料類型無關的排序副程式

前面所介紹的副程式只能適用於關鍵值為實數之情形。對於其他資料類型之關鍵值，程式需略加修改。事實上，只要將關鍵值之比較及搬移改為呼叫外在函數與副程式，即可使一個排序程式適用於各種資料類型之關鍵值及指標排序。詳細做法請直接參閱副程式$HEAPG$及$QUICKG$，並與副程式$HEAP$及$QUICK$對照即可明白，在此不予贅述。其他排序副程式亦可仿此寫成。

[表七(a)]及[表七(b)]之程式分別為適用於實數關鍵值及其指標排序之外在函數與副程式之範例。其他資料類型(如字串等)之函數與副程式，均可仿此寫成。當然這些副程式具有運算效率較差(運算時間增加之百分比對非指標排序約為65，對指標排序約為50)及資料須經由共用區塊(COMMON BLOCK)傳遞之缺點。

注意外在函數之傳回值為邏輯(.TRUE., .FALSE.)而不是整數值(-1,0,+1)可以增加頗多的效率。

[表六] 與資料類型無關之排序副程式

```
**************************************************************************
      SUBROUTINE HEAPG(N)
C**   ============================================================    **
C**   LOGICAL ICMPIJ,ICMPIX,ICMPXJ                                   **
C**   ============================================================    **
C**   Heap  ascending  sort (for decending : change .LE. to .GE.)     **
C**   =================< key sorting >==+===< index sorting >=====    **
C**   DIMENSION             A(N)        |       A(N),L(N)            **
C**   After sorting A : A(I).LE.A(I+1)  |  A(L(I)).LE.A(L(I+1))      **
C**                 D : A(I).GE.A(I+1)  |  A(L(I)).GE.A(L(I+1))      **
C**   ----------------------------------+-------------------------    **
C**   ICMPIJ(I,J) = A : A(I).LE.A(J)    |  A(L(I)).LE.A(L(J))        **
C**   ICMPIX(I)   = A : A(I).LE. AX     |  A(L(I)).LE. A(LX)         **
C**   ICMPXJ(J)   = A :  AX .LE.A(J)    |   A(LX) .LE.A(L(J))        **
C**   ----------------------------------+-------------------------    **
C**   MOVEIJ(I,J) :      A(I)=A(J)      |      L(I)=L(J)             **
C**   MOVEIX(I)   :      A(I)=AX        |      L(I)=LX               **
C**   MOVEXJ(I)   :      AX=A(J)        |      LX=L(J)               **
C**   ----------------------------------+-------------------------    **
C**   SWAPIJ(I,J) :   AT=A(I)           |   LT=L(I)                  **
C**                :      A(I)=A(J)     |      L(I)=L(J)             **
C**                :           A(J)=AT  |           L(J)=LT          **
C**   ----------------------------------+-------------------------    **
C**   ROTXIJ(I,J) : (AX=A(I)) A(I)=A(J) | (LX=L(I)) L(I)=L(J)        **
C**                :  A(J)=AX & AX=A(I)  |  L(J)=LX & LX=L(I)         **
C**   ============================================================    **
C**   INITIL(N)   : Set L(I)=I for I=1,N for index sorting            **
C**   ============================================================    **
      CALL INITIL(N)
C**   +-----------------------------------------------------------+    **
```

```
C**   | Form heap (A(I),I=K,N) for K=N/2 ... 1                    |    **
C**   +-----------------------------------------------------------+    **
      DO 10 K=N/2,1,-1
      CALL MOVEXJ(K)
      CALL SHIFTG(K,N)
   10 CONTINUE
C**   +-----------------------------------------------------------+    **
C**   | Put max of A(1:M) in the correct position M               |    **
C**   | Form heap (A(I),I=1,M-1) for M=N ... 2                    |    **
C**   +-----------------------------------------------------------+    **
      DO 20 M=N,2,-1
      CALL MOVEXJ(M)
      CALL MOVEIJ(M,1)
      CALL SHIFTG(1,M-1)
   20 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE SHIFTG(K,M)
C**   ============================================================    **
      LOGICAL ICMPIJ,ICMPIX,ICMPXJ
C**   ============================================================    **
C**   Given AX and heap A(K+1:M)  where  K .LE. M                     **
C**   Shift AX down to form new heap A(K:M)                           **
C**   A: A(I).GE.A(2*I) .AND. A(I).GE.A(2*I+1)                        **
C**   ============================================================    **
      I=K
   60 J=I+I
      IF(J.LE.M) THEN
        IF(J.LT.M) THEN
          IF(.NOT.ICMPIJ(J+1,J)) J=J+1
        ENDIF
        IF(.NOT.ICMPIX(J)) THEN
C**       +-----------------------------------------------------+    **
C**       | Shift AX down to position J                         |    **
C**       +-----------------------------------------------------+    **
          CALL MOVEIJ(I,J)
          I=J
          GO TO 60
        ENDIF
      ENDIF
C**   +-----------------------------------------------------------+    **
C**   | No more shift down : I is the correct position for AX     |    **
C**   +-----------------------------------------------------------+    **
      CALL MOVEIX(I)
      RETURN
      END
************************************************************************
      SUBROUTINE QUICKG(N)
C**   ============================================================    **
      LOGICAL ICMPIJ,ICMPIX,ICMPXJ
      DIMENSION IL(20),IU(20)
C**   IL(NN),IU(NN) permit sorting up to N=2**(NN+1)-1                **
C**   ============================================================    **
C**   Quick  ascending sort (for decending : change .LE. to .GE.)     **
C**   =================< key sorting >==+===< index sorting >=====    **
C**   DIMENSION             A(N)        |       A(N),L(N)           **
C**   After sorting A : A(I).LE.A(I+1)  |  A(L(I)).LE.A(L(I+1))     **
C**                 D : A(I).GE.A(I+1)  |  A(L(I)).GE.A(L(I+1))     **
C**   ----------------------------------+-------------------------    **
```

```
C**   ICMPIJ(I,J) = A : A(I).LE.A(J)     |  A(L(I)).LE.A(L(J))         **
C**   ICMPIX(I)   = A : A(I).LE. AX      |  A(L(I)).LE. A(LX)          **
C**   ICMPXJ(J)   = A :  AX .LE.A(J)     |   A(LX) .LE.A(L(J))         **
C**   -----------------------------------+-------------------------    **
C**   MOVEIJ(I,J) :      A(I)=A(J)       |      L(I)=L(J)              **
C**   MOVEIX(I)   :      A(I)=AX         |      L(I)=LX                **
C**   MOVEXJ(I)   :      AX=A(J)         |      LX=L(J)                **
C**   -----------------------------------+-------------------------    **
C**   SWAPIJ(I,J) :   AT=A(I)            |   LT=L(I)                   **
C**                :      A(I)=A(J)      |      L(I)=L(J)              **
C**                :           A(J)=AT   |           L(J)=LT           **
C**   -----------------------------------+-------------------------    **
C**   ROTXIJ(I,J) : (AX=A(I)) A(I)=A(J) | (LX=L(I)) L(I)=L(J)         **
C**                :  A(J)=AX & AX=A(I)  |  L(J)=LX & LX=L(I)          **
C**   ================================================================    **
C**   INITIL(N)   : Set L(I)=I for I=1,N for index sorting            **
C**   ================================================================    **
      CALL INITIL(N)
C**   ----------------------------------------------------------------    **
      I=1
      M=N
      NN=1
C**   ----------------------------------------------------------------    **
   10 IF(I.GE.M) GO TO 70
C**   +--------------------------------------------------------------+    **
C**   | Sorting A(I),A(IX),A(M)                                      |    **
C**   | And get AX=MID(A(I),A(IX),A(M))                              |    **
C**   +--------------------------------------------------------------+    **
      IX=(I+M)/2
      CALL MOVEXJ(IX)
      IF(.NOT.ICMPIX(I)) THEN
        CALL ROTXIJ(IX,I)
      ENDIF
      IF(.NOT.ICMPXJ(M)) THEN
        CALL ROTXIJ(IX,M)
        IF(.NOT.ICMPIX(I)) THEN
          CALL ROTXIJ(IX,I)
        ENDIF
      ENDIF
C**   ================================================================    **
C**   SUBROUTINE PARTITION(A,I,M,J,K,AX)
C**   ================================================================    **
C**   Sorting A(I:M) by partition into A(I:K) and A(J:M)              **
C**   ================================================================    **
C**   Such that  I <= K < J <= M  provided I < M                      **
C**   A(x) <= AX  for  I <= x <= K                                    **
C**   A(x)  = AX  for  K <  x <  J                                    **
C**   A(x) >= AX  for  J <= x <= M                                    **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Test right part for A(K)>AX & left part for A(J)<AX          |    **
C**   +--------------------------------------------------------------+    **
      J=I
      K=M
   40 K=K-1
      IF(.NOT.ICMPIX(K)) GO TO 40
   50 J=J+1
      IF(.NOT.ICMPXJ(J)) GO TO 50
C**   ----------------------------------------------------------------    **
      IF(J.LE.K) THEN
```

```
        CALL SWAPIJ(J,K)
        GO TO 40
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | End of partitioning : Find longer partition to save    |    **
C**   +--------------------------------------------------------+    **
      IF(K-I.GT.M-J) THEN
C**     +------------------------------------------------------+    **
C**     | Save  left partition A(I:K) then sort A(J:M)         |    **
C**     +------------------------------------------------------+    **
        IL(NN)=I
        IU(NN)=K
        NN=NN+1
        I=J
      ELSE
C**     +------------------------------------------------------+    **
C**     | Save right partition A(J:M) then sort A(I:K)         |    **
C**     +------------------------------------------------------+    **
        IL(NN)=J
        IU(NN)=M
        NN=NN+1
        M=K
      ENDIF
      GO TO 80
C**   +--------------------------------------------------------+    **
C**   | Get a saved partition A(I:M) for sorting               |    **
C**   +--------------------------------------------------------+    **
   70 NN=NN-1
      IF(NN.EQ.0) RETURN
      I=IL(NN)
      M=IU(NN)
C**   +--------------------------------------------------------+    **
C**   | Use quick sort for long key array                      |    **
C**   +--------------------------------------------------------+    **
   80 IF(M-I.GE.11.OR.I.EQ.1) GO TO 10
C**   +--------------------------------------------------------+    **
C**   | Use shaker sort for short key array                    |    **
C**   +--------------------------------------------------------+    **
   90 I=I+1
      IF(I.GT.M) GO TO 70
      IF(ICMPIJ(I-1,I)) GO TO 90
      K=I
      CALL MOVEXJ(K)
   95 CALL MOVEIJ(K,K-1)
      K=K-1
      IF(.NOT.ICMPIX(K-1)) GO TO 95
C**   ----------------------------------------------------------    **
      CALL MOVEIX(K)
      GO TO 90
      END
*************************************************************************
```

［表七(a)］ 與資料類型無關之外在函數與副程式

```
**************************************************************************
      LOGICAL FUNCTION ICMPIJ(I,J)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   +----------------------------------------------------------+    **
C**   | For decending key sorting                                |    **
C**   +----------------------------------------------------------+    **
C**   ICMPIJ=A(I).GE.A(J)
C**   +----------------------------------------------------------+    **
C**   | For ascending key sorting                                |    **
C**   +----------------------------------------------------------+    **
      ICMPIJ=A(I).LE.A(J)
      RETURN
      END
**************************************************************************
      LOGICAL FUNCTION ICMPIX(I)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   +----------------------------------------------------------+    **
C**   | For decending key sorting                                |    **
C**   +----------------------------------------------------------+    **
C**   ICMPIX=A(I).GE.AX
C**   +----------------------------------------------------------+    **
C**   | For ascending key sorting                                |    **
C**   +----------------------------------------------------------+    **
      ICMPIX=A(I).LE.AX
      RETURN
      END
**************************************************************************
      LOGICAL FUNCTION ICMPXJ(J)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   +----------------------------------------------------------+    **
C**   | For decending key sorting                                |    **
C**   +----------------------------------------------------------+    **
C**   ICMPXJ=AX.GE.A(J)
C**   +----------------------------------------------------------+    **
C**   | For ascending key sorting                                |    **
C**   +----------------------------------------------------------+    **
      ICMPXJ=AX.LE.A(J)
      RETURN
      END
**************************************************************************
      SUBROUTINE MOVEIJ(I,J)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   ============================================================    **
C**   For key sorting
C**   ============================================================    **
      A(I)=A(J)
      RETURN
      END
**************************************************************************
      SUBROUTINE MOVEIX(I)
```

```
C**   ================================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   ================================================================    **
C**   For key sorting
C**   ================================================================    **
      A(I)=AX
      RETURN
      END
*************************************************************************
      SUBROUTINE MOVEXJ(J)
C**   ================================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   ================================================================    **
C**   For key sorting
C**   ================================================================    **
      AX=A(J)
      RETURN
      END
*************************************************************************
      SUBROUTINE SWAPIJ(I,J)
C**   ================================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   ================================================================    **
C**   For key sorting
C**   ================================================================    **
      AT=A(I)
      A(I)=A(J)
      A(J)=AT
      RETURN
      END
*************************************************************************
      SUBROUTINE ROTXIJ(I,J)
C**   ================================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   ================================================================    **
C**   For key sorting
C**   ================================================================    **
      A(I)=A(J)
      A(J)=AX
      AX=A(I)
      RETURN
      END
*************************************************************************
      SUBROUTINE INITIL(N)
C**   ================================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/INTINK/ AX
C**   ================================================================    **
C**   For key sorting
C**   ================================================================    **
      RETURN
      END
*************************************************************************
```

[表七(b)] 與資料類型無關之外在函數與副程式-指標排序

```
************************************************************************
      LOGICAL FUNCTION ICMPIJ(I,J)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   ============================================================    **
C**   For stable sorting by HEAPG or QUICKG                           **
C**       Change Cstable to blanks                                    **
C**   ============================================================    **
Cstable IF(A(L(I)).NE.A(L(J))) THEN
C**       +-------------------------------------------------+    **
C**       | For decending index sorting                     |    **
C**       +-------------------------------------------------+    **
C**       ICMPIJ=A(L(I)).GE.A(L(J))
C**       +-------------------------------------------------+    **
C**       | For ascending index sorting                     |    **
C**       +-------------------------------------------------+    **
          ICMPIJ=A(L(I)).LE.A(L(J))
Cstable ELSE
Cstable   ICMPIJ=L(I).LE.L(J)
Cstable ENDIF
      RETURN
C**   +-------------------------------------------------------+    **
C**   | Following statements may be more efficient            |    **
C**   | For some computers or compilers                       |    **
C**   +-------------------------------------------------------+    **
C**   | For decending stable index sorting                    |    **
C**   +-------------------------------------------------------+    **
C**   IF(A(L(I))-A(L(J))) 3,2,1
C**   +-------------------------------------------------------+    **
C**   | For ascending stable index sorting                    |    **
C**   +-------------------------------------------------------+    **
C**   IF(A(L(I))-A(L(J))) 1,2,3
C** 1 ICMPIJ=.TRUE.
C**   RETURN
C** 2 ICMPIJ=L(I).LE.L(J)
C**   RETURN
C** 3 ICMPIJ=.FALSE.
C**   RETURN
      END
************************************************************************
      LOGICAL FUNCTION ICMPIX(I)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   ============================================================    **
C**   For stable sorting by HEAPG or QUICKG                           **
C**       Change Cstable to blanks                                    **
C**   ============================================================    **
Cstable IF(A(L(I)).NE.A(LX)) THEN
C**       +-------------------------------------------------+    **
C**       | For decending index sorting                     |    **
C**       +-------------------------------------------------+    **
C**       ICMPIX=A(L(I)).GE.A(LX)
C**       +-------------------------------------------------+    **
C**       | For ascending index sorting                     |    **
```

```
C**       +-------------------------------------------------+    **
          ICMPIX=A(L(I)).LE.A(LX)
Cstable ELSE
Cstable   ICMPIX=L(I).LE.LX
Cstable ENDIF
      RETURN
      END
*************************************************************************
      LOGICAL FUNCTION ICMPXJ(J)
C**   =============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   =============================================================    **
C**   For stable sorting by HEAPG or QUICKG                            **
C**       Change Cstable to blanks                                     **
C**   =============================================================    **
Cstable IF(A(LX).NE.A(L(J))) THEN
C**       +-------------------------------------------------+    **
C**       | For decending index sorting                     |    **
C**       +-------------------------------------------------+    **
C**       ICMPXJ=A(LX).GE.A(L(J))
C**       +-------------------------------------------------+    **
C**       | For ascending index sorting                     |    **
C**       +-------------------------------------------------+    **
          ICMPXJ=A(LX).LE.A(L(J))
Cstable ELSE
Cstable   ICMPXJ=LX.LE.L(J)
Cstable ENDIF
      RETURN
      END
*************************************************************************
      SUBROUTINE MOVEIJ(I,J)
C**   =============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   =============================================================    **
C**   For index sorting                                                **
C**   =============================================================    **
      L(I)=L(J)
      RETURN
      END
*************************************************************************
      SUBROUTINE MOVEIX(I)
C**   =============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   =============================================================    **
C**   For index sorting                                                **
C**   =============================================================    **
      L(I)=LX
      RETURN
      END
*************************************************************************
      SUBROUTINE MOVEXJ(J)
C**   =============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
```

```
      COMMON/INTINK/ LX
C**   ============================================================    **
C**   For index sorting                                               **
C**   ============================================================    **
      LX=L(J)
      RETURN
      END
**********************************************************************
      SUBROUTINE SWAPIJ(I,J)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   ============================================================    **
C**   For index sorting                                               **
C**   ============================================================    **
      LT=L(I)
      L(I)=L(J)
      L(J)=LT
      RETURN
      END
**********************************************************************
      SUBROUTINE ROTXIJ(I,J)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   ============================================================    **
C**   For index sorting                                               **
C**   ============================================================    **
      L(I)=L(J)
      L(J)=LX
      LX=L(I)
      RETURN
      END
**********************************************************************
      SUBROUTINE INITIL(N)
C**   ============================================================    **
      COMMON/EXTKEY/ A(1)
      COMMON/EXTIND/ L(1)
      COMMON/INTINK/ LX
C**   ============================================================    **
C**   For index sorting                                               **
C**   ============================================================    **
      DO 10 I=1,N
   10 L(I)=I
      RETURN
      END
**********************************************************************
```

18.8 穩定排序

所謂穩定排序 (Stable sorting) 為排序後相同大小之關鍵值仍保持其原來的前後相關順序。前面所介紹的氣泡排序法及震動排序法屬穩定排序；而堆積排序法及快速排序法則不是穩定排序法。

非穩定排序法亦可做穩定排序，只須於比較關鍵值之大小時，如果遇到關鍵值相等則再予比較其原來位置指標的大小以決定其順序。這樣做除了須多做一些比較外，如何知道其原來位置指標則是一個問題。對於指標排序，因關鍵值之位置不改變故不成問題；對於非指標排序，則可能需要利用一個N元整數陣列來儲存原來位置指標，如此則倒不如利用指標排序。因之在此只提供指標排序之做法。如[表七(b)]中將起始字元為$Cstable$者改為空白，亦即加入那幾行指令，即可使$HEAPG$及$QUICKG$做穩定排序。注意$BUBBLG$及$SHAKEG$(見習題)本來就是穩定排序，不要更改[表七(b)]，以免浪費運算時間。[表七(b)]改為穩定排序後運算時間約增加百分之33(改變程式的寫法，如將第一個邏輯IF改為算術IF，對不同電腦或編譯器可能會改變運算時間之增加百分比)。副程式$HEAPL$及$QUICKL$亦可仿此改為穩定排序。

18.9 快速排序法與堆積排序法運算量之比較

如經詳細分析或在副程式中加入一些計數的指令，即可獲得各指令之大約運算量如[表三](僅列出與$N*\log_2 N$成正比者)。表中$a=\log_2 N,\ b=\log_2(N/2),\ c=\log_2(N/8)$，當$N$很大時可視為$a=b=c$。

表中備註欄內符號相同者為二個副程式之運算量大致可以相抵的部分(雖然$QUICK$者總略少於$HEAP$者)。因此可明顯看出$QUICK$之運算量確實少很多。

［表三］ 快速排序法與堆積排序法之運算量

副程式	指令型式	運算量	備註
$QUICK$	$AT = A(J)\ ,\ \ A(J) = A(I)\ ,\ \ A(I) = AT$	$cN3/4$	$=$
	$K = K - 1\ ,\ \ J = J + 1$	cN	$+$
	$IF(A(K).GT.AX)\ ,\ \ IF(A(J).LT.AX)$	cN	$>$
	$IF(J.LE.K)$	$cN/4$	$<$
$HEAP$	$J = I + I$	aN	$+$
	$IF(J - M)$	aN	$<$
	$IF(A(J).LT.A(J + 1))$	bN	
	$J = J + 1$	$bN/2$	
	$IF(A(J).LE.AX)$	bN	$>$
	$A(I) = A(J)$	bN	$=$
	$I = J$	bN	

習題

1. 試將副程式$BUBBLE$及$QUICK$改為指標排序副程式$BUBBLL$及$QUICKL$。

2. 試將副程式$BUBBLE$及$SHAKER$改為與關鍵值資料類型無關的排序副程式$BUBBLG$及$SHAKEG$。

3. 試將[表七(a)]或[表七(b)]之程式改為適用於關鍵值為字串之外在函數與副程式。

4. 雪氏排序法(Shell's sort)為震動排序法的一種改良排序法。先令$M = N/2$，然後依次減半(化為整數)，直至$M = 1$為止。對於每個M值，做如下之排序：將$A(I), A(I+M), A(I+2M), A(I+3M), \ldots$視為一組，因此總共有$M$組，分別利用震動排序法排列之。事實上，只要做最後一次$M = 1$時之排序即可排序完成。對於$M > 1$時之排序雖然是多餘的，但其可將關鍵值做較大步的移動，故其效率頗佳，運算時間約與$N^{3/2}$成正比。試寫該法之副程式$SHELL$。

參考文獻

1. Knuth, D.E., *Sorting and Searching,* Vol.3 of *The Art of Computer Programming,* Addison-Wesley, 1973.

2. Williams, J. W. J., "Algorithm 232, Heapsort," Comm. ACM 7(June 1964), p.347.

3. Floyd, R. W., "Algorithm 245, Treesort 3 [M1]," Comm. ACM 7(Dec. 1964), p.701.

4. Hoare, C. A. R., "Algorithm 63, Partition, and 64, Quicksort," Comm. ACM 4(July 1961), p.321.

5. Singleton, R. C., "Algorithm 347, An efficient algorithm for sorting with minimal storage [M1]," Comm. ACM, 12(March 1969), p.185-187.

6. Shell, D. L., "Algorithm 201, A high-speed sorting procedure," Comm. ACM, 2(July 1959), p.30-32.

第十九章

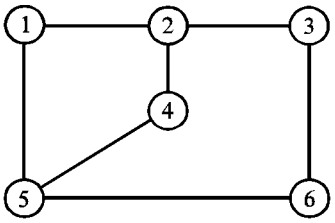
減少矩陣帶寬之結點排序法

19.1 前言

結構勁度矩陣分析或有限元素分析時，未知數之號碼常按照結點號碼之順序編排。而未知數之編排順序將影響勁度矩陣之帶寬。但求解矩陣所需之時間，以常用之直接解法而言，約與帶寬之平方成正比。因此如何編排結點之號碼以減少矩陣之帶寬，實為有限元素分析中一個重要之課題。目前有限元素分析常以程式自動產生結點，此時結點號碼之自動編排也就成為必備的工具。採用結點重排資料時，如考慮下列各點，即可不影響原來有限元素分析程式之架構與資料結構：(1)僅採用結點重排資料做為未知數編碼之用，而分析程式之輸入與輸出甚至內部資料之儲存，仍然採用原來之結點號碼。如此方可免去各處理程式引用結點重排資料之運算與該資料之保留與傳遞。(2)結點重排資料，僅一次用於未知數編碼之時，此時只需改按新的結點順序編排未知數之號碼，即可達到結點重排減少帶寬之目的。(3)若結點重排資料係由自動編碼程式置於檔案中，則有限元素分析程式之未知數編碼副程式，僅需多預留（或借用）與結點等數目之整數陣列以讀進結點重排資料。於未知數編碼後，由於(1)項之考量，其他程式即可不再使用結點重排資料，甚至可將此項資料捨去，因此而可不影響原程式之其他資料結構，而最重要的好處是不必更改其他副程式。

19.2 結點號碼與矩陣元素排列之關係

通常一個結點之未知數不只一個，但大部分為相等，故其個數對結點重排之影響不大，因此以後之說明即假設一個結點對應一個未知數。有限元素分析中之勁度矩陣係由個別元素之勁度矩陣所組成，因此矩陣元素之排列取決於結點間之元素。以後之討論僅考慮每一元素僅與二個結點相連之情形，此元素即為圖形理論之邊(Edge)，結點即為圖形理論之點(Node)。如元素與三個結點相連，可用三邊取代；與m個結點相連，可用$m(m-1)/2$個邊取代，即其中任意二結點都需有一個邊相連之。因此組成有限元素之元素與結點，即可由(1)點之集合：$\{x_1, x_2, \ldots, x_n\}$，稱點集$\{X\}$，與(2)邊之集合：$\{(x_1, x_2), (x_5, x_3), \ldots\}$，稱邊集$\{E\}$，兩者所成之圖形$G = (\{X\}, \{E\})$來表示。以圖一之圖形為例：$\{X\} = \{1,2,3,4,5,6\}$，$\{E\} = \{(1,2),(2,3),(1,5),(2,4),(3,6),(5,6),(4,5)\}$，其對應矩陣非零元素(即*元素)如圖二(a)。若結點順序改為：$\{X\} = \{3,6,2,5,1,4\}$，則其對應矩陣如圖二(b)。 由圖二(a)轉為圖二(b)之結點重排資料可用一重排指標$LVSN(n)$表示：$LVSN(1)$=3, $LVSN(2)$=6, $LVSN(3)$=2, $LVSN(4)$=5, $LVSN(5)$=1, $LVSN(6)$=4，或$LVSN(1{:}6)$=$\{3,6,2,5,1,4\}$。$LVSN(2) = 6$意為新2號結點為原6號結點，或原6號結點重排為新2號結點。



圖一 圖形之點與邊

	1	2	3	4	5	6
1	*	*			*	
2	*	*	*	*		
3		*	*			*
4		*		*	*	
5	*			*	*	*
6			*		*	*

圖二(a)

	3	6	2	5	1	4
3	*	*	*			
6	*	*		*		
2	*		*		*	*
5		*		*	*	*
1			*	*	*	
4			*	*		*

圖二(b)

19.3 圖形之資料表示法

將圖形之邊集$\{E\}$用一個二維陣列$JOE(2,Ne)$儲存Ne個邊之二端結點號碼，此項資料即已能表示一圖形之完整資料。但該資料之儲存方式，對於結點重排之運算極為不便，因此需將圖形資料改用下述較便於結點重排運算之資料，稱“結點之相鄰諸結點”。因各結點之相鄰結點之數目，稱度數，並不相等，因此不方便以一個二維陣列儲存。如圖一之圖形，各結點之相鄰諸結點依序為$(2,5),(1,3,4),(2,6),(2,5),(1,4,6),(3,5)$，各結點之度數依序為$2,3,2,2,3,2$。若將各結點之相鄰諸結點依序儲存在一個一維陣列中，即$ADJN(1{:}14)=\{2,5,1,3,4,2,6,2,5,1,4,6,3,5\}$。再用一組位置指標指明各結點之相鄰諸結點之起始位置，即$XADJ(1{:}7)=\{1,3,6,8,10,13,15\}$。如此即可方便取得各結點之相鄰諸結點：即結點$J$之相鄰諸結點為$ADJN(XADJ(J))$至$ADJN(XADJ(J{+}1){-}1)$。而結點$J$之度數等於$XADJ(J{+}1){-}XADJ(J)$。注意陣列$XADJ$之資料個數比結點數$Nn$多一，以方便上述二項運算。

$ADJN(K)$	2	5	1	3	4	2	6	2	5	1	4	6	3	5	
K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$XADJ(J)$	1	3	6	8	10	13	15
J	1	2	3	4	5	6	7

欲建立上述之“結點之相鄰諸結點”之一維陣列資料，應先有結點之度數以建立位置指標$XADJ(Nn+1)$。但欲求得結點之度數，則須建立結點之相鄰諸結點之資料（直接計算某結點出現在邊集內之次數做為該結點之度數，不一定會正確，因與某二結點相連之邊可以重複而不只一個，如有限元素之圖形即是）。因此直接建立上述資料確有困難。以下再介紹一種以鏈串 (Link) 方式儲存之資料結構。以此方式建立“結點之相鄰諸結點”之資料，不必先有結點之度數，因此沒有上述困難。但此方式之儲存空間需前者之二倍，且用於結點重排之運算時效率較差。因此僅打算以此方式求得各結點之度數：即以鏈串方式建立“結點之相鄰諸結點”後，僅保留結點之度數以建立位置指標$XADJ(Nn+1)$，再用以建立一維陣列資料$ADJN()$。既然以鏈串方式建立資料之目的僅為了獲得結點之度數，因此如改為建立“結點之相鄰諸上結點”，即可僅用一半之儲存空間，而可

借用與其儲存空間大小相等之$ADJN$陣列之位置。所謂某結點之相鄰諸“上結點”指相鄰結點其號碼比某結點大者，亦即不包含比某結點之號碼為小之相鄰結點。以下即為圖一之“結點之相鄰諸上結點”之鏈串資料。以結點$J=2$為例其取得相鄰結點之方式如下：由$K_1=HEAD(J)=2$，得其相鄰上結點$NODE(K_1)=3$；再由$K_2=LINK(K_1)=4$，得另一相鄰上結點$NODE(K_2)=4$；最後由$K_3=LINK(K_2)=-2$，為負值而知已無相鄰上結點。注意結點2之相鄰結點1，因$1<2$，而未列入鏈串中。

$LINK(K)$	3	4	-1	-2	-3	-5	-4
$NODE(K)$	2	3	5	4	6	6	5
K	1	2	3	4	5	6	7

$HEAD(J)$	1	2	5	7	6	-6
J	1	2	3	4	5	6

如圖形中想加入一新邊$(1,4)$時，可按如下步驟於結點1加入一新的相鄰上結點4：(1)先確定結點4尚未列在結點$J=1$之相鄰上結點中：由$K_1=HEAD(J)=1$，得其相鄰上結點$NODE(K_1)=2$，不等於4；再由$K_2=LINK(K_1)=3$，得另一相鄰上結點$NODE(K_2)=5$，亦不等於4；最後由$K_3=LINK(K_2)=-1$，為負值而可確定4尚未列在相鄰上結點中。(2)加入結點4：令$LINK(Next)=-1$，$NODE(Next)=4$，$LINK(K_2)=Next$，$Next=Next+1$。其中$Next$為下一個可用儲存位置，結點4加入之前為8，加入之後為9。

$LINK(K)$	3	4	8	-2	-3	-5	-4	-1
$NODE(K)$	2	3	5	4	6	6	5	4
K	1	2	3	4	5	6	7	8

$HEAD(J)$	1	2	5	7	6	-6
J	1	2	3	4	5	6

副程式$GRAPHS$即是利用上述做法求結點度數，存於$JAJT$，再建立$XADJ$與$ADJN$，分別存於$JAJP$與$JAJS$。於建立鏈串資料時$HEAD$存於$JAJP$，$NODE(K)$存於$JAJS(2*K-1)$，$LINK(K)$存於$JAJS(2*K)$。

19.4 矩陣之帶寬與包圍量

在第三章討論過：矩陣$[A]$分解成下三角矩陣$[L]$與上三角矩陣$[U]$之乘積，可保住矩陣之帶狀特性。$[L]$僅需儲存每列由左第一個非零元素至對角線間之元素，其個數（含非零元素但不計對角元素）稱該列之左半帶寬。$[U]$僅需儲存每行由上第一個非零元素至對角線間之元素，其個數（含非零元素但不計對角元素）稱該行之上半帶寬。矩陣$[L]$之包圍量(Envelope或Profile)為各列之左半帶寬之和。矩陣$[U]$之包圍量為各行之上半帶寬之和。本章僅考慮$[A]$為對稱矩陣，故$[L]=[U]^T$，矩陣$[L]$或$[U]$之包圍量亦稱為矩陣$[A]$之包圍量。而各列之左半帶寬之最大者，或各行之上半帶寬之最大者，即為矩陣$[A]$之半帶寬。而最大左半帶寬與最大上半帶寬之和加一，即為矩陣$[A]$之帶寬。

另外，本章會討論到將矩陣對應之未知變數反向排列之情形：即矩陣之第一列與最後一列對調，第二列與倒數第二列對調，餘類推。矩陣各行亦做類似對調。如欲由原矩陣求對調後之矩陣之包圍量，則需計算(1)每列由右第一個非零元素至對角線間之元素個數，即該列之右半帶寬；或(2)每行由下第一個非零元素至對角線間之元素個數，即該行之下半帶寬。則對調後矩陣之包圍量等於各列之右半帶寬之和或各行之下半帶寬之和。因某列j之右半帶寬為對調後對應列$(Nn-j+1)$之左半帶寬；某行之下半帶寬為對調後對應行之上半帶寬。注意矩陣對調前後之包圍量不一定會相等。

19.5 *RCM(Reverse Cuthill–McKee)*法

Nn個結點之排列總數高達$Nn!$之多，對於較大之Nn，即使運算速度再快，亦幾乎無法從所有可能排列中找出符合某種要求條件之最佳排列。這就是結點重排問題困難之處。但是如果不要求最佳之結果，則因可能之合適之結果也必然為數甚多，因此也必存在許多方法可尋求各種合適之結果。本節所要介紹的RCM法為目前最通用的方法，其運算效率甚佳且所得結果也不差。RCM法基本上係將CM法所得之結點順序，反向排列而得。因為由CM所得之結點排列經反向後，矩陣之帶寬不變，但矩陣之包圍量，只會減少不會增多，此點將於介紹CM法後再加說明。

以下將CM法分為二部分說明：(1)固定根結點後，結點之排序法；

(2)根結點之決定。結點排列中之第一個結點，稱為根結點(Root)。結點之排序法則一經確定，則由選定之根結點即已決定了結點之排列順序。因此根結點之選取相當重要。但大部分的運算工作為結點之排序過程，因此先介紹(1)再介紹(2)。

以下為結點排序之編號原則（自根結點開始編為新1號）：

(1)與已編號結點相連之結點應比未相連之結點先編號。

(2)先編號結點之相鄰結點比後編號結點之相鄰結點先編號。

(3)對某一結點之相鄰結點編號時，結點之度數少者比度數多者先編號。

$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\left[\begin{array}{cccccc}
\cdot & \cdot & \cdot & & & \\
\cdot & x & \cdot & * & & * \\
\cdot & \cdot & \cdot & & & \\
 & * & & y & & \\
 & & & & \cdot & \\
 & * & & & & y
\end{array}\right]
\end{array}
$$

圖三

原則(1)主要目的在減少帶寬，因若某結點x之相鄰結點y之編號每延後一號，該y結點對應之左半(或上半)帶寬即多一。例如圖三中當考慮對已編為新2號之結點x之某相鄰結點y編號時：若y編為新4號，則第4列之左半帶寬為2=4−2；若y編為新6號，則第6列之左半帶寬為4=6−2。原則(2)目的亦在減少帶寬，因若後編號結點之相鄰結點先編號時，將使先編號結點之相鄰結點延後編號而增加其帶寬。原則(3)目的在將較可能引致較大帶寬之度數多之相鄰結點儘量延後編號，亦希望因此能減少帶寬。注意這些原則基本上雖以減少帶寬為考量，但均僅從某結點的主觀因素考慮局部之效應，並未顧及與其他結點的互動影響，故仍然不能保証會求得最小之帶寬。

根據原則(3)，若將某結點之相鄰結點預先按照其度數由小至大之順序排列，則在對某結點之相鄰結點編號時，即可依結點取得之順序編號。前節提過以$LVSN(1{:}Nn)$表示結點重排資料，現假設結點編排過程中，已編號結點數為$NSIZ$，即新號已編至$NSIZ$，而$LVSN(1), LVSN(2), \ldots, LVSN(NSIZ)$依序為新1號至新$NSIZ$號之結點原號碼。結點若已編新號

碼，可令$XADJ(I)$為正值記錄之（未編號前先變為負值），以避免重覆編碼。以下為結點之編號過程：

(1)令$NSIZ=1$，$LVSN(NSIZ)=ROOT$，$XADJ(ROOT)=-XADJ(ROOT)$，$ROOT$為根結點原號碼。

(2)令i自1起至Nn為止，做下列運算：

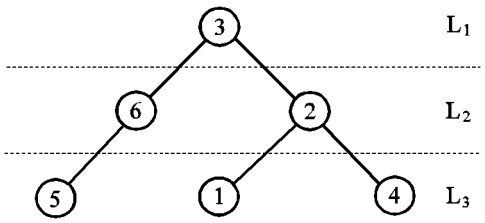
(2a)由$k=LVSN(i)$，知新i號結點為原k號結點。

(2b)令j為自$ADJN(|XADJ(k)|)$起至$ADJN(|XADJ(k+1)|-1)$為止之相鄰結點，若j尚未編新號碼，即$XADJ(j)<0$，則將其編碼為新$NSIZ$+1號，即令$NSIZ=NSIZ$+1，$LVSN(NSIZ)=j$，$XADJ(j)=-XADJ(j)$。

以下說明CM法中根結點之決定方法。該法係以下列定義之層次結構之特性（如層深及底層之結點）為依據做運算以決定根結點：

(1)第1層為根結點所組成之點集合$\{L_1\}$。

(2)第$i+1$層為第i層所屬結點之所有相鄰結點中經去除第i層與第$i-1$層之結點後之點集合$\{L_{i+1}\}$。



圖四 圖一圖形之層次結構

一層次結構之總層數稱為層深(Depth)。第1層又稱頂層；最後一層又稱底層。以圖一為例，當根結點為$ROOT$=3時，$\{L_1\}=\{3\}$，$\{L_2\}=\{6,2\}$，$\{L_3\}=\{5,1,4\}$，層深為3，如圖四所示。由編號過程中，亦可順便獲得層次結構，即各層之點集合。例如第i+1層之結點為編排第i層結點之相鄰結點時所有新編碼之結點。以下即為CM法決定根結點之過程：

(1)任選一結點為根結點。

(2)以該根結點編新號碼並求得層次結構。記錄其層深。

(3)由底層結點中選度數最少者為新的根結點。

(4)反覆步驟(2)(3)，比較層深大小，至層深不再增加為止。

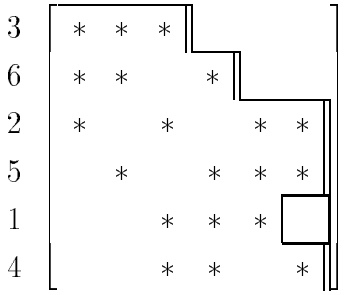
通常層深愈大帶寬會愈小，但並非絕對的關係。因此本章提供程式多考

慮下述二項改進之做法：(1)在編號過程中順便算出帶寬，並多比較帶寬大小，至帶寬亦不再減小為止。(2)在編號過程中亦順便算出包圍量，並多比較包圍量大小，至包圍量亦不再減小為止。上述過程(3)選根結點的原則之主要理由略述如次：為了增加層深，根結點最好選自圖形之外圍周邊結點，而外圍周邊結點較可能出現在底層之點集合中，且外圍周邊結點之度數通常會比內部結點者為小，故根結點選底層度數最少之結點。以下說明另一種做法供參考：根結點取自底層結點中度數小於某一限值之結點，如此即有更多的編碼結果可供比較選擇，因而可能獲得較理想的編碼。但其所增之編碼運算量為數可觀，並不一定值得。

19.6 反向排列之道理與包圍量之求法

本節將先說明CM法排列所對應之矩陣之特性：就矩陣之上三角元素而言，令第i列之最右邊之$*$元素之行號為$C(i)$，令$B(i)=\max_{k=1}^{i} C(k)$，即$B(i)$為$C(1)$至$C(i)$中之最大值。若$m(i)=C(i)-B(i-1)>0$，則第i列自第$B(i-1)+1$行起會有連續$m(i)$個$*$元素。此特性可由CM法之編號原則(1)與(2)看出：考慮目前正針對新i號結點之相鄰結點編號。注意在此項編號前，對其先前$i-1$個新號結點之相鄰結點已編至第$B(i-1)$號。若其所有相鄰結點均已編過號，則$B(i)=B(i-1)$，$C(i)\le B(i)$；若有相鄰結點尚未編號，設有$m(i)$個，則按照原則(2)，其編號必須起自$B(i-1)+1$，且連號持續編至$C(i)=B(i-1)+m(i)$，即將所有尚未編號之相鄰結點全部編完為止，且按照原則(1)其間不能插編未相鄰之結點。因此第i列在第$B(i-1)$行之右邊會有連續$m(i)$個$*$元素。

若在各列之第$B(i)$行與第$B(i)+1$行間劃一垂直線，稱滴水線；另在前述連續$m(i)$個$*$元素上方劃一水平線做為屋頂面；則水滴沿滴水線滴下即會落在屋頂面而只能順著屋頂面向右流至下一個滴水線。各行由屋頂面至矩陣對角線間所包圍之元素數，即為該行之上半帶寬，各行上半帶寬之和，即為CM法排列所得矩陣之包圍量$\mathrm{Env(CM)}$。注意該包圍量亦等於：各列由滴水線至矩陣對角線間所包圍之元素數相加之和，即$\mathrm{Env(CM)}=\sum_{i=1}^{Nn}(B(i)-i)$。現考慮將$CM$法之結點排列反向，即$RCM$法之結點排列，則其所對應之矩陣之包圍量等於各列右半帶寬之和，即$\mathrm{Env(RCM)}=\sum_{i=1}^{Nn}(C(i)-i)$。但因$C(i)\le B(i)$，故$\mathrm{Env(RCM)}\le\mathrm{Env(CM)}$。至於二種結點排列之半帶寬則均等於$\max_{i=1}^{Nn}(B(i)-i)$。



圖五(a)

	$C(i)$	$B(i)$	$LVSN(i)$	i
	3	3	3	1
	4	4	6	2
	6	6	2	3
	6	6	5	4
	5	6	1	5
	6	6	4	6

圖五(b)

前述 $B(i) = \max_{k=1}^{i} C(i)$ 之定義可寫成 $B(i) = \max(B(i-1), C(i))$。即若 $m(i) = C(i) - B(i-1) > 0$，則 $B(i) = C(i) > B(i-1)$；但若 $m(i) = C(i) - B(i-1) \le 0$，則 $B(i) = B(i-1) \ge C(i)$。注意 $B(i)$ 可以直接由結點之編號過程中獲得：在新 i 號之相鄰結點已全部編完時之 $NSIZ$ 即等於 $B(i)$。但 $C(i)$ 之值，除了當 $B(i) > B(i-1)$ 時，$C(i) = B(i)$ 之情形外，需要另尋途徑求取。以下介紹二種求 $C(i)$ 之方式：

方式一：在前述之結點編號過程中之(2b)後加做下列二步驟：

(2c)若 $B(i) > B(i-1)$，則 $C(i) = B(i)$；否則令 $C(i) = B(i-1)$ 續做(2d)。

(2d)若 $C(i) = i$，則結束；否則，做下列運算：

令 $k = LVSN(C(i))$，比較 k 是否為新 i 號結點之相鄰結點。

若是，則 $C(i)$ 已求得而結束；若不是，令 $C(i) = C(i)-1$ 續做(2d)。

方式二：將結點編號過程改為：

(0)令 $LISN(1{:}Nn) = 0$。

(1)令 $NSIZ = 1$，$LVSN(NSIZ) = ROOT$，$LISN(ROOT) = NSIZ$。

(2)令 i 自1起至 Nn 為止，做下列運算：

(2a)由 $k = LVSN(i)$，知新 i 號結點為原 k 號結點。

(2b)令 j 為自 $ADJN(XADJ(k))$ 起至 $ADJN(XADJ(k+1)-1)$ 為止之相鄰結點，若 j 尚未編新號碼，即 $LISN(j) = 0$，則將其編碼為新 $NSIZ$+1號，即令 $NSIZ = NSIZ$+1，$LVSN(NSIZ) = j$，$LISN(j) = NSIZ$。

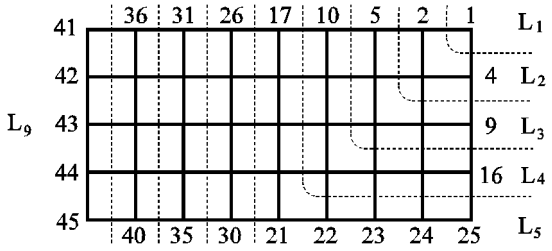
(2c)若 $B(i) > B(i-1)$，則 $C(i) = B(i)$；否則，令 $C(i) = i$ 續做(2d)。

(2d)令 j 為自 $ADJN(XADJ(k))$ 起至 $ADJN(XADJ(k+1)-1)$ 為止之相鄰結點，計算 $C(i) = \max(C(i), LISN(j))$。

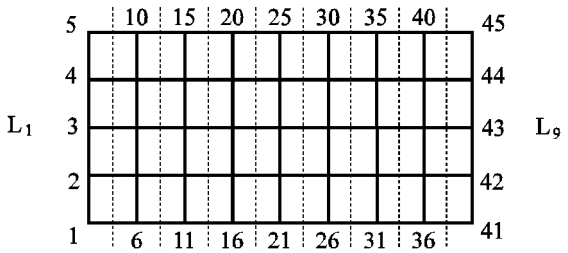
本章所提供程式係採用方式一。方式二則做為習題，該法需增加陣列 $LISN(1{:}Nn)$，以方便(2d)之運算，附帶可用其做為是否已編號之指標，以取代$XADJ()$之正值標記，因此而更改(2b)如上，其他對$XADJ()$取絕對值之指令亦應配合更改。另外注意，前面所用之$C(i)$, $B(i)$, $m(i)$並不需要用陣列儲存，僅為說明方便而設。在本章程式中，$NSIZ$用以存$B(i)$，$NSAV$先暫存$B(i{-}1)$，而於$B(i) \le B(i{-}1)$時，$NSAV$即用以存$C(i)$，$NDIF$為計算包圍量用之$B(i){-}i$或$C(i){-}i$。

19.7 多根頂層之考慮

通常在頂層附近之結點順序會較零亂，帶寬較參差不齊，而至底層會較規律而平順。此現象主要是因根結點只有一個之故。因此考慮將底層較規律之結點同時放在頂層而形成多根頂層之層次結構。頂層內之結點順序可採用前次排列反向編號後之順序，此順序因已較平順，應可誘導後繼結點之規律性。比較圖六(a)與圖六(b)可知其對規則性有限元素之功效。不過因圖形各具特性，任何做法均很難對大部分情況都有效。因此上述做法，在程式中僅當做一種替代方案：即在最後試用一次多根頂層，但其包圍量或帶寬必須比RCM小才採用。



圖六(a) 單根頂層之層次結構



圖六(b) 多根頂層之層次結構

19.8 有限元素網之結點排序法

有限元素網中每一元素之結點大都超過二個，不像圖形理論討論之邊僅有二個結點。雖然前面提過m個點的元素可用$m(m-1)/2$個邊替代，但由此所得之邊為數可觀，需要較大的儲存陣列。因此考慮以有限元素網之下述二組資料取代“結點之相鄰諸結點”：

(1) 以$JOES(M),\ JOEP(Ne+1)$儲存“元素之諸結點”：即將位在第I號元素之諸結點存放在$JOES(JOEP(I))$至$JOES(JOEP(I{+}1)-1)$中。此項資料為有限元素網之基本資料，通常為已知。

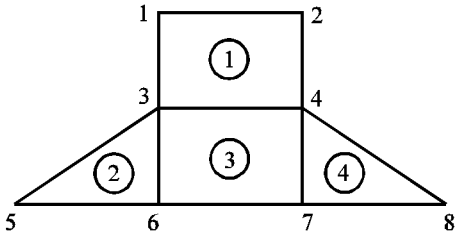
(2) 以$EOJS(M),\ EOJP(Nn+1)$儲存“結點之諸元素”：即將含有第J號結點之諸元素存放在$EOJS(EOJP(J))$至$EOJS(EOJP(J{+}1)-1)$中。此項資料可很容易地由(1)項資料求得。其好處為$EOJS(M)$之資料數M與$JOES(M)$者相等，且遠比“結點之相鄰諸結點”者為少，又因個數已知而容易預留其空間。以下為圖七之有限元素網之對應資料：

$JOES(K)$	1	3	4	2	3	5	6	3	6	7	4	4	7	8
K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$JOEP(I)$	1	5	8	12	15
I	1	2	3	4	5

$EOJS(K)$	1	1	1	2	3	1	3	4	2	2	3	3	4	4	
K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$EOJP(J)$	1	2	3	6	9	10	12	14	15
J	1	2	3	4	5	6	7	8	9



圖七 有限元素網

利用上二項資料在對某結點之相鄰結點編號時，須先透過$EOJS$取得相鄰元素，再由該相鄰元素透過$JOES$取得相鄰結點。由於相鄰結點未集中存放，無法先行按度數由小至大之順序排列。因此改在某一結點之相鄰結點全編完後，再根據結點度數由小至大之順序調整編號。不過程式$FEMRCM$係用結點之元素數取代結點度數以調整編號，因此與原CM法之編號原則(3)略為不同。如欲維持該原則，可照習題2建議方式處理。

19.9 副程式$GRAPHS$與$GENRCM$

[表一]副程式$GRAPHS$係利用有限元素網之資料建立“結點之相鄰諸結點”以做為呼叫副程式$GENRCM$之輸入資料。[表二]副程式$GENRCM$係利用RCM法重排結點，並考慮多項改進，即增加帶寬與包圍量之計算與比較，並與多根頂層之排序結果比較以擇優選用。

[表一] 有限元素網轉成結點之相鄰諸結點

```
************************************************************************
      SUBROUTINE GRAPHS(JOEP,JOES,JAJT,JAJP,JAJS,NELM,NODE,NEDG)
C**   ============================================================    **
      DIMENSION JOEP(1),JOES(1),JAJT(1),JAJP(1),JAJS(1)
C**   ============================================================    **
C**   Input  : JOEP(NELM+1),JOES(JOEP(NELM+1)-1),NELM,NODE            **
C**   Output : JAJP(NODE+1),JAJS(JAJP(NODE+1)-1),JAJT(NODE),NEDG      **
C**   ------------------------------------------------------------    **
C*I   JOES(JOEP(I)),...,JOES(JOEP(I+1)-1) = Nodes Of Element I        **
C*O   JAJS(JAJP(J)),...,JAJS(JAJP(J+1)-1) = Adjacency Nodes Of J      **
C*O   JAJT(J) = Degree of node J                                      **
C*I   NELM    = Number of Elements                                    **
C*I   NODE    = Number of Nodes                                       **
C*O   NEDG    = Number of Edges                                       **
C**   ============================================================    **
C**   +-----------------------------------------------------------+    **
C**   | Setup temporarily Lower adjacency Linked Lists            |    **
C**   | Adjacency Nodes of J are JAJS(JAJP(J)),                   |    **
C**   |                      JAJS(JAJS(JAJP(J)+1)),               |    **
C**   |                JAJS(JAJS(JAJS(JAJP(J)+1)+1)),...          |    **
C**   +-----------------------------------------------------------+    **
      NEXT=1
      DO 100 J=1,NODE
      JAJP(J)=-J
  100 JAJT(J)=0
      DO 180 IE=1,NELM
      NJA=JOEP(IE)
      NJB=JOEP(IE+1)-1
      IF(NJA.GT.NJB) GO TO 180
      DO 170 KX=NJA,NJB
      JX=JOES(KX)
      DO 160 KY=NJA,NJB
      JY=JOES(KY)
```

```
      IF(JX.GE.JY) GO TO 160
C**   +-------------------------------------------------------------+    **
C**   | Get the pointer of the first adjacency node                 |    **
C**   +-------------------------------------------------------------+    **
      JXP=JAJP(JX)
      IF(JXP.LE.0) THEN
        JAJP(JX)=NEXT
        GO TO 150
      ENDIF
C**   +-------------------------------------------------------------+    **
C**   | Test if node JY is already in adjacency lists of node JX    |    **
C**   | YES : CONTINUE                                              |    **
C**   | NO  : Get the pointer of the next adjacency node            |    **
C**   +-------------------------------------------------------------+    **
  120 IF(JAJS(JXP).EQ.JY) GO TO 160
      JXN=JAJS(JXP+1)
      IF(JXN.LE.0) THEN
        JAJS(JXP+1)=NEXT
        GO TO 150
      ENDIF
      JXP=JXN
      GO TO 120
C**   +-------------------------------------------------------------+    **
C**   | Put adjacency node at the end of List & Update pointer      |    **
C**   | Update degrees of nodes JX and JY : JAJT(JX) & JAJT(JY)     |    **
C**   +-------------------------------------------------------------+    **
  150 JAJS(NEXT)=JY
      JAJS(NEXT+1)=-JX
      NEXT=NEXT+2
      JAJT(JX)=JAJT(JX)+1
      JAJT(JY)=JAJT(JY)+1
  160 CONTINUE
  170 CONTINUE
  180 CONTINUE
C**   +-------------------------------------------------------------+    **
C**   | Setup adjacency structure for RCM method                    |    **
C**   | Given JAJT(J) as the degree of node J                       |    **
C**   +-------------------------------------------------------------+    **
      JAJP(1)=1
      DO 200 J=1,NODE
      JAJP(J+1)=JAJP(J)+JAJT(J)
  200 JAJT(J)=0
      DO 280 IE=1,NELM
      NJA=JOEP(IE)
      NJB=JOEP(IE+1)-1
      IF(NJA.GT.NJB) GO TO 280
      DO 270 KX=NJA,NJB
      JX=JOES(KX)
      JXT=JAJT(JX)
      DO 260 KY=NJA,NJB
      JY=JOES(KY)
      IF(JX.EQ.JY) GO TO 260
      JXP=JAJP(JX)
      IF(JXT.EQ.0) GO TO 250
      DO 240 JT=1,JXT
      IF(JAJS(JXP).EQ.JY) GO TO 260
  240 JXP=JXP+1
  250 JAJS(JXP)=JY
      JXT=JXT+1
  260 CONTINUE
```

```
  270 JAJT(JX)=JXT
  280 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Shaker sorting JAJS(NJX) ... JAJS(NJY)                |    **
C**   | in increasing order of degree for Adj nodes of each node |    **
C**   +-------------------------------------------------------+    **
      DO 380 J=1,NODE
      NJX=JAJP(J)
      NJY=JAJP(J+1)-1
      IF(NJX.GE.NJY) GO TO 380
      JYP=NJX
  330 JXP=JYP
      JYP=JYP+1
      JY=JAJS(JYP)
      JYT=JAJT(JY)
C**   -------------------------------------------------------    **
  340 JX=JAJS(JXP)
      JXT=JAJT(JX)
      IF(JXT.LE.JYT) GO TO 350
      JAJS(JXP+1)=JX
      JXP=JXP-1
      IF(JXP.GE.NJX) GO TO 340
C**   -------------------------------------------------------    **
  350 JAJS(JXP+1)=JY
      IF(JYP.LT.NJY) GO TO 330
  380 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Compute the numbers of the edges                      |    **
C**   +-------------------------------------------------------+    **
      NEDG=JAJP(NODE+1)-1
      RETURN
      END
*************************************************************************
```

［表二］ 一般圖形之結點重排程式

```
*************************************************************************
      SUBROUTINE GENRCM(NODE,XADJ,ADJN,LVSN)
C**   ===========================================================    **
      INTEGER XADJ(1),ADJN(1),LVSN(1)
C**   ===========================================================    **
C**   Perform the RCM ordering to get LVSN(NODE)                     **
C**   -----------------------------------------------------------    **
C**   Input  : XADJ(NODE+1),ADJN(XADJ(NODE+1)-1),NODE                **
C**   Output : LVSN(NODE),  ADJN : Sorted by degree in incr. order   **
C**   -----------------------------------------------------------    **
CIO   ADJN(XADJ(J)),...,ADJN(XADJ(J+1)-1) = Adjacency Nodes Of J     **
C*O   LVSN(I) = The original node number of the new I-th node        **
C*I   NODE    = Number of Nodes                                      **
C**   ===========================================================    **
C**   +-------------------------------------------------------+    **
C**   | Set LVSN(J) temporarily as the degree of node J       |    **
C**   +-------------------------------------------------------+    **
      DO 250 J=1,NODE
  250 LVSN(J)=XADJ(J+1)-XADJ(J)
C**   +-------------------------------------------------------+    **
C**   | Shaker sorting ADJN(NJX) ... ADJN(NJY)                |    **
C**   +-------------------------------------------------------+    **
      DO 380 J=1,NODE
      NJX=XADJ(J)
```

```
      NJY=XADJ(J+1)-1
      IF(NJX.GE.NJY) GO TO 380
      JYP=NJX
  330 JXP=JYP
      JYP=JYP+1
      JY=ADJN(JYP)
      JYT=LVSN(JY)
C**   -------------------------------------------------------------    **
  340 JX=ADJN(JXP)
      JXT=LVSN(JX)
      IF(JXT.LE.JYT) GO TO 350
      ADJN(JXP+1)=JX
      JXP=JXP-1
      IF(JXP.GE.NJX) GO TO 340
C**   -------------------------------------------------------------    **
  350 ADJN(JXP+1)=JY
      IF(JYP.LT.NJY) GO TO 330
  380 CONTINUE
C**   +-----------------------------------------------------------+    **
C**   | Set XADJ(J) < 0 to mark node J as an un-re-numbered node |    **
C**   +-----------------------------------------------------------+    **
      DO 500 J=1,NODE
  500 XADJ(J)=-XADJ(J)
C**   +-----------------------------------------------------------+    **
C**   | Call MRRCM to re-number node begin from ROOT = J         |    **
C**   +-----------------------------------------------------------+    **
      NBGN=1
      DO 600 J=1,NODE
      IF(XADJ(J).GT.0) GO TO 600
      CALL MRRCM(J,XADJ,ADJN,LVSN(NBGN),NSIZ)
      NBGN=NBGN+NSIZ
      IF(NBGN.GT.NODE) RETURN
  600 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE MRRCM(IROOT,XADJ,ADJN,LVSN,NSIZ)
C**   =============================================================    **
      INTEGER XADJ(1),ADJN(1),LVSN(1),LVBG(2),ROOT
      DATA KLVL/2/,MAXINT/32767/
C**   =============================================================    **
      ROOT=IROOT
      IROO=0
      NLVL=0
      NBND=MAXINT
      NENV=MAXINT
C**   +-----------------------------------------------------------+    **
C**   | Single Root begin re-number from here                    |    **
C**   +-----------------------------------------------------------+    **
  100 NSIZ=1
      LVSN(NSIZ)=ROOT
      XADJ(ROOT)=-XADJ(ROOT)
C**   +-----------------------------------------------------------+    **
C**   | Multiple Roots begin re-number from here                 |    **
C**   +-----------------------------------------------------------+    **
  110 NLVO=NLVL
      NBNO=NBND
      NENO=NENV
      NLVL=0
      NBND=0
```

```
      NENV=0
      LVEND=0
      DO 120 K=1,KLVL
  120 LVBG(K)=1
C**   +-------------------------------------------------------+    **
C**   | LVBGN & LVEND = Pointers to first & last of this level |    **
C**   +-------------------------------------------------------+    **
  200 NLVL=NLVL+1
      LVBGN=LVEND+1
      LVEND=NSIZ
      DO 210 K=2,KLVL
  210 LVBG(K-1)=LVBG(K)
      LVBG(KLVL)=LVBGN
C**   +-------------------------------------------------------+    **
C**   | Generate the new level by finding all the masked      |    **
C**   | un-re-numbered neighbors of nodes in this level       |    **
C**   +-------------------------------------------------------+    **
      DO 400 I=LVBGN,LVEND
      NOD=LVSN(I)
      JSTRT=IABS(XADJ(NOD))
      JSTOP=IABS(XADJ(NOD+1))-1
      IF(JSTRT.GT.JSTOP) GO TO 400
      NSAV=NSIZ
      DO 300 J=JSTRT,JSTOP
      NOD=ADJN(J)
      IF(XADJ(NOD).GT.0) GO TO 300
      XADJ(NOD)=-XADJ(NOD)
      NSIZ=NSIZ+1
      LVSN(NSIZ)=NOD
  300 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Compute the exact envelope of RCM ordering            |    **
C**   | by reducing the envelope of CM ordering               |    **
C**   +-------------------------------------------------------+    **
      NDIF=NSIZ-I
      IF(NSAV.LT.NSIZ) GO TO 380
  320 IF(NSAV.LE.I) GO TO 360
      NOD=LVSN(NSAV)
      DO 340 J=JSTRT,JSTOP
      IF(ADJN(J).EQ.NOD) GO TO 360
  340 CONTINUE
      NSAV=NSAV-1
      GO TO 320
  360 NDIF=NSAV-I
  380 NBND=MAX0(NBND,NDIF)
      NENV=NENV+NDIF
  400 CONTINUE
      IF(NSIZ.GT.LVEND) GO TO 200
C**   +-------------------------------------------------------+    **
C**   | No more node in the new level(All nodes are re-numbered) |    **
C**   | Compare the Depth(Band,Envelope) of two level structures |    **
C**   +-------------------------------------------------------+    **
  405 FORMAT(/' ** ROOT,NLVL,NBND,NENV/LVSN(*)=',4I8/(10I8))
      WRITE(*,405) ROOT,NLVL,NBND,NENV,(LVSN(I),I=1,NSIZ)
      IF(ROOT.EQ.IABS(IROO).OR.NLVL.EQ.NSIZ) GO TO 800
      IF(NLVL-NLVO) 700,420,450
  420 IF(NBND-NBNO) 450,430,700
  430 IF(NENV-NENO) 450,800,700
C**   +-------------------------------------------------------+    **
C**   | Find new ROOT from the last level with minimum degree |    **
```

```
C**   +-------------------------------------------------------+    **
  450 IROO=ROOT
      MINDEG=NSIZ
      DO 500 I=LVBGN,LVEND
      NOD=LVSN(I)
      NDEG=IABS(XADJ(NOD+1))-IABS(XADJ(NOD))
      IF(NDEG.GE.MINDEG) GO TO 500
      MINDEG=NDEG
      ROOT=NOD
  500 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Reset the mark to restart the re-numbering            |    **
C**   +-------------------------------------------------------+    **
  550 DO 600 I=1,NSIZ
      NOD=LVSN(I)
  600 XADJ(NOD)=-XADJ(NOD)
      GO TO 100
C**   +-------------------------------------------------------+    **
C**   | Reconstruct previous level structure                  |    **
C**   +-------------------------------------------------------+    **
  700 ROOT=IROO
      GO TO 550
C**   +-------------------------------------------------------+    **
C**   | Reverse the Cuthill-Mckee ordering                    |    **
C**   +-------------------------------------------------------+    **
  800 NS=NSIZ/2
      L=NSIZ
      DO 900 I=1,NS
      NOD=LVSN(I)
      LVSN(I)=LVSN(L)
      LVSN(L)=NOD
  900 L=L-1
      IF(NENV.GT.NENO) GO TO 550
      IF(IROO.LT.0) RETURN
      IROO=-ROOT
C**   +-------------------------------------------------------+    **
C**   | Use all nodes in the last KLVL levels as Multiple Roots |    **
C**   +-------------------------------------------------------+    **
      I1=LVEND-LVBG(1)+2
      DO 960 I=I1,NSIZ
      NOD=LVSN(I)
  960 XADJ(NOD)=-XADJ(NOD)
      NSIZ=I1-1
      GO TO 110
      END
************************************************************************
```

19.10 副程式$FEMRCM$

[表三]副程式$FEMRCM$直接利用有限元素網資料以RCM法重排結點。其內部自動建立“結點之諸元素”以取代“結點之相鄰諸結點”，前者之陣列長度遠比後者為小，且已先預知，使用較為方便。$FEMRCM$亦與$GENRCM$一樣考慮多項改進。[表四]為試用程式及輸入資料。其結果為$LVSN(1{:}9) = \{3,2,4,1,9,5,8,6,7\}$。

［表三］ 有限元素網之結點重排程式

```
************************************************************************
      SUBROUTINE FEMRCM(JOEP,JOES,LVSN,EOJP,EOJS,NELM,NODE)
C**   ================================================================    **
      INTEGER JOEP(1),JOES(1),LVSN(1),EOJP(1),EOJS(1)
C**   ================================================================    **
C**   Perform the RCM ordering to get LVSN(NODE)                          **
C**   ----------------------------------------------------------------    **
C**   Input  : JOEP(NELM+1),JOES(JOEP(NELM+1)-1),NELM,NODE                **
C**   Output : EOJP(NODE+1),EOJS(EOJP(NODE+1)-1),LVSN(NODE)               **
C**   ----------------------------------------------------------------    **
C*I   JOES(JOEP(I)),...,JOES(JOEP(I+1)-1) = Nodes Of Element I            **
C*O   EOJS(EOJP(J)),...,EOJS(EOJP(J+1)-1) = Elements of Node J            **
C*O   LVSN(I) = The original node number of the new I-th node             **
C*I   NELM    = Number of Elements                                        **
C*I   NODE    = Number of Nodes                                           **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Set LVSN(J) temporarily as the degree of node J            |    **
C**   +--------------------------------------------------------------+    **
      DO 100 NOD=1,NODE
  100 LVSN(NOD)=0
      DO 180 MEM=1,NELM
      JSTRT=JOEP(MEM)
      JSTOP=JOEP(MEM+1)-1
      IF(JSTRT.GT.JSTOP) GO TO 180
      DO 170 J=JSTRT,JSTOP
      NOD=JOES(J)
  170 LVSN(NOD)=LVSN(NOD)+1
  180 CONTINUE
C**   +--------------------------------------------------------------+    **
C**   | Setup a list of the element connected to each node         |    **
C**   | Given LVSN(J) as the degree of node J                      |    **
C**   +--------------------------------------------------------------+    **
      EOJP(1)=1
      DO 200 NOD=1,NODE
      EOJP(NOD+1)=EOJP(NOD)+LVSN(NOD)
  200 LVSN(NOD)=EOJP(NOD)
      DO 280 MEM=1,NELM
      JSTRT=JOEP(MEM)
      JSTOP=JOEP(MEM+1)-1
      IF(JSTRT.GT.JSTOP) GO TO 280
      DO 270 J=JSTRT,JSTOP
      NOD=JOES(J)
      K=LVSN(NOD)
      LVSN(NOD)=K+1
  270 EOJS(K)=MEM
  280 CONTINUE
C**   +--------------------------------------------------------------+    **
C**   | Set EOJP(J) < 0 to mark node J as an un-re-numbered node    |    **
C**   +--------------------------------------------------------------+    **
      DO 500 NOD=1,NODE
  500 EOJP(NOD)=-EOJP(NOD)
C**   +--------------------------------------------------------------+    **
C**   | Call EMRRCM to re-number node begin from ROOT = NOD         |    **
C**   +--------------------------------------------------------------+    **
      NBGN=1
      DO 600 NOD=1,NODE
      IF(EOJP(NOD).GT.0) GO TO 600
```

```
      CALL EMRRCM(NOD,JOEP,JOES,EOJP,EOJS,LVSN(NBGN),NELM,NSIZ)
      NBGN=NBGN+NSIZ
      IF(NBGN.GT.NODE) RETURN
  600 CONTINUE
      RETURN
      END
*********************************************************************
      SUBROUTINE EMRRCM(IROOT,JOEP,JOES,EOJP,EOJS,LVSN,NELM,NSIZ)
C**   ============================================================    **
      INTEGER JOEP(1),JOES(1),EOJP(1),EOJS(1),LVSN(1),LVBG(2),ROOT
      DATA KLVL/2/,MAXINT/32767/
C**   ============================================================    **
      ROOT=IROOT
      IROO=0
      NLVL=0
      NBND=MAXINT
      NENV=MAXINT
C**   +----------------------------------------------------------+    **
C**   | Single Root begin re-number from here                    |    **
C**   +----------------------------------------------------------+    **
  100 NSIZ=1
      LVSN(NSIZ)=ROOT
      EOJP(ROOT)=-EOJP(ROOT)
C**   +----------------------------------------------------------+    **
C**   | Multiple Roots begin re-number from here                 |    **
C**   +----------------------------------------------------------+    **
  110 NLVO=NLVL
      NBNO=NBND
      NENO=NENV
      NLVL=0
      NBND=0
      NENV=0
      LVEND=0
      DO 120 K=1,KLVL
  120 LVBG(K)=1
C**   +----------------------------------------------------------+    **
C**   | LVBGN & LVEND = Pointers to first & last of this level   |    **
C**   +----------------------------------------------------------+    **
  200 NLVL=NLVL+1
      LVBGN=LVEND+1
      LVEND=NSIZ
      DO 210 K=2,KLVL
  210 LVBG(K-1)=LVBG(K)
      LVBG(KLVL)=LVBGN
C**   +----------------------------------------------------------+    **
C**   | Generate the new level by finding all the masked         |    **
C**   | un-re-numbered neighbors of nodes in this level          |    **
C**   +----------------------------------------------------------+    **
      DO 400 I=LVBGN,LVEND
      NOD=LVSN(I)
      KSTRT=IABS(EOJP(NOD))
      KSTOP=IABS(EOJP(NOD+1))-1
      IF(KSTRT.GT.KSTOP) GO TO 400
      NSAV=NSIZ
      DO 240 K=KSTRT,KSTOP
      MEM=EOJS(K)
      IF(JOEP(MEM).LT.0) GO TO 240
      JOEP(MEM)=-JOEP(MEM)
      JSTRT=IABS(JOEP(MEM))
      JSTOP=IABS(JOEP(MEM+1))-1
```

```
C     IF(JSTRT.GT.JSTOP) GO TO 240
      DO 230 J=JSTRT,JSTOP
      NOD=JOES(J)
      IF(EOJP(NOD).GT.0) GO TO 230
      EOJP(NOD)=-EOJP(NOD)
      NSIZ=NSIZ+1
      LVSN(NSIZ)=NOD
  230 CONTINUE
  240 CONTINUE
C**
      IF(NSAV.GE.NSIZ) GO TO 320
      NDIF=NSIZ-I
      NJX=NSAV+1
      NJY=NSIZ
      IF(NJX.GE.NJY) GO TO 380
C**   +-------------------------------------------------------+    **
C**   | Shaker sorting LVSN(NJX) ... LVSN(NJY)                |    **
C**   +-------------------------------------------------------+    **
      JYP=NJX
  250 JXP=JYP
      JYP=JYP+1
      JY=LVSN(JYP)
      JYT=IABS(EOJP(JY+1))-IABS(EOJP(JY))
C**   -------------------------------------------------------    **
  260 JX=LVSN(JXP)
      JXT=IABS(EOJP(JX+1))-IABS(EOJP(JX))
      IF(JXT.LE.JYT) GO TO 270
      LVSN(JXP+1)=JX
      JXP=JXP-1
      IF(JXP.GE.NJX) GO TO 260
C**   -------------------------------------------------------    **
  270 LVSN(JXP+1)=JY
      IF(JYP.LT.NJY) GO TO 250
      GO TO 380
C**   +-------------------------------------------------------+    **
C**   | Compute the exact envelope of RCM ordering            |    **
C**   | by reducing the envelope of CM ordering               |    **
C**   +-------------------------------------------------------+    **
  320 IF(NSAV.LE.I) GO TO 360
      NOD=LVSN(NSAV)
      DO 340 K=KSTRT,KSTOP
      MEM=EOJS(K)
      JSTRT=IABS(JOEP(MEM))
      JSTOP=IABS(JOEP(MEM+1))-1
      DO 340 J=JSTRT,JSTOP
      IF(JOES(J).EQ.NOD) GO TO 360
  340 CONTINUE
      NSAV=NSAV-1
      GO TO 320
  360 NDIF=NSAV-I
  380 NBND=MAX0(NBND,NDIF)
      NENV=NENV+NDIF
  400 CONTINUE
      IF(NSIZ.GT.LVEND) GO TO 200
C**   +-------------------------------------------------------+    **
C**   | No more node in the new level(All nodes are re-numbered) |    **
C**   | Compare the Depth(Band,Envelope) of two level structures |    **
C**   +-------------------------------------------------------+    **
  405 FORMAT(/' ** ROOT,NLVL,NBND,NENV/LVSN(*)=',4I8/(10I8))
      WRITE(*,405) ROOT,NLVL,NBND,NENV,(LVSN(I),I=1,NSIZ)
```

```
      IF(ROOT.EQ.IABS(IROO).OR.NLVL.EQ.NSIZ) GO TO 800
      IF(NLVL-NLVO) 700,420,450
  420 IF(NBND-NBNO) 450,430,700
  430 IF(NENV-NENO) 450,800,700
C**   +-------------------------------------------------------+    **
C**   | Find new ROOT from the last level with minimum degree |    **
C**   +-------------------------------------------------------+    **
  450 IROO=ROOT
      MINDEG=NSIZ
      DO 500 I=LVBGN,LVEND
      NOD=LVSN(I)
      NDEG=IABS(EOJP(NOD+1))-IABS(EOJP(NOD))
      IF(NDEG.GE.MINDEG) GO TO 500
      MINDEG=NDEG
      ROOT=NOD
  500 CONTINUE
C**   +-------------------------------------------------------+    **
C**   | Reset the mark to restart the re-numbering            |    **
C**   +-------------------------------------------------------+    **
  550 DO 600 I=1,NSIZ
      NOD=LVSN(I)
  600 EOJP(NOD)=-EOJP(NOD)
      DO 620 MEM=1,NELM
  620 JOEP(MEM)=IABS(JOEP(MEM))
      GO TO 100
C**   +-------------------------------------------------------+    **
C**   | Reconstruct previous level structure                  |    **
C**   +-------------------------------------------------------+    **
  700 ROOT=IROO
      GO TO 550
C**   +-------------------------------------------------------+    **
C**   | Reverse the Cuthill-Mckee ordering                    |    **
C**   +-------------------------------------------------------+    **
  800 NS=NSIZ/2
      L=NSIZ
      DO 900 I=1,NS
      NOD=LVSN(I)
      LVSN(I)=LVSN(L)
      LVSN(L)=NOD
  900 L=L-1
      DO 920 MEM=1,NELM
  920 JOEP(MEM)=IABS(JOEP(MEM))
      IF(NENV.GT.NENO) GO TO 550
      IF(IROO.LT.0) RETURN
      IROO=-ROOT
C**   +-------------------------------------------------------+    **
C**   | Use all nodes in the last KLVL levels as Multiple Roots |    **
C**   +-------------------------------------------------------+    **
      I1=LVEND-LVBG(1)+2
      DO 960 I=I1,NSIZ
      NOD=LVSN(I)
  960 EOJP(NOD)=-EOJP(NOD)
      NSIZ=I1-1
      GO TO 110
      END
*************************************************************************
```

［表四］ 結點重排之試用主程式

```
*************************************************************************
```

```
      PROGRAM TRCM
C**   ============================================================    **
      DIMENSION IA(2000)
      DATA NDIM/2001/,N1/1/
C**   ============================================================    **
   10 READ(*,'(2I5)') NELM,NODE
      N2=N1+NELM+1
      CALL GRAPHI(IA(N1),IA(N2),NELM,NPNT)
C**   ------------------------------------------------------------    **
      N3=N2+NPNT
      N4=N3+NODE
      N5=N4+NODE+1
      N6=N5+NPNT
      IF(N6.GT.NDIM) STOP
      CALL FEMRCM(IA(N1),IA(N2),IA(N3),IA(N4),IA(N5),NELM,NODE)
      CALL GRAPHO(IA(N3),IA(N4),IA(N5),NODE,0)
C**   ------------------------------------------------------------    **
      CALL GRAPHS(IA(N1),IA(N2),IA(N3),IA(N4),IA(N5),NELM,NODE,NEDG)
      N6=N5+NEDG
      IF(N6.GT.NDIM) STOP
      CALL GENRCM(NODE,IA(N4),IA(N5),IA(N3))
      CALL GRAPHO(IA(N3),IA(N4),IA(N5),NODE,1)
      GO TO 10
      END
*************************************************************************
      SUBROUTINE GRAPHI(JOEP,JOES,NELM,NPNT)
C**   ============================================================    **
      DIMENSION JOEP(1),JOES(1,1)
      CHARACTER*15 IFM
C**   ============================================================    **
      READ(*,'(A)') IFM                              !   Input data
      IX=1                                           !  ------------
      JOEP(1)=IX                                     !     8,9
      DO 200 IE=1,NELM                               !     (20I4)
C**   READ(*,IFM) NX,(JOES(J,IX),J=1,NX)             !     1,9,8
      READ(*,IFM) (JOES(J,IX),J=1,20)                !     1,2,9
      NX=0                                           !     2,4,9
      DO 150 J=1,20                                  !     2,3,4
      IF(JOES(J,IX).LE.0) GO TO 160                  !     8,6,7
  150 NX=J                                           !     8,9,6
  160 IX=IX+NX                                       !     9,5,6
  200 JOEP(IE+1)=IX                                  !     9,4,5
      NPNT=IX-1
      WRITE(*,'(/'' JOEP/JOES:''/(10I8))') (JOEP(I),I=1,NELM+1)
      WRITE(*,'(10I8)')                    (JOES(J,1),J=1,NPNT)
      RETURN
      END
*************************************************************************
      SUBROUTINE GRAPHO(LVSN,EOJP,EOJS,NODE,M)
C**   ============================================================    **
      INTEGER LVSN(1),EOJP(1),EOJS(1)
C**   ============================================================    **
      IF(M.EQ.0) WRITE(*,'(/'' LVSN/EOJP/EOJS:'')')
      IF(M.EQ.1) WRITE(*,'(/'' LVSN/XADJ/ADJN:'')')
      WRITE(*,'(10I8)') (LVSN(J),J=1,NODE)
      WRITE(*,'(10I8)') (EOJP(J),J=1,NODE+1)
      WRITE(*,'(10I8)') (EOJS(J),J=1,EOJP(NODE+1)-1)
      RETURN
      END
*************************************************************************
```

19.11 有限元素分析程式之修改範例

[表五]列示有限元素中編排結點自由度號碼之一範例副程式，右側部分為原來之程式；左側者為考慮結點重排時之程式。注意其中僅改了三行指令。原程式係照原來之結點號碼J之順序編排結點自由度之號碼$NVAR(K,J)$；新程式係照結點重排之順序，即$LVSN(1), LVSN(2), \ldots,$ $LVSN(NODE)$之順序編排結點自由度之號碼。

[表五] 結點重排後結點自由度號碼之編排

```
**************************************************************************
      SUBROUTINE EQNO(NODE,NDF,NFILE,NRL,NVAR,LVSN)
      INTEGER NRL(NDF,NODE),NVAR(NDF,NODE),LVSN(NODE)
C**   ============================================================     **
C*I   NODE      = Number of Nodes                                       **
C*I   NDF       = Number of Degrees of Freedom for each Node            **
C*I   NFILE     = File number which stores the array LVSN(NODE)         **
C*I   NRL(K,J)  = Node Restraint List                                   **
C**             > 0 : Restraint direction, its Displacement = 0         **
C**             = 0 : Allow displacement,                               **
C**                   Its displacement = DISP(NVAR(K,J))                **
C*O   NVAR(K,J) = 0 : For restraint direction                           **
C**             > 0 : Sequence number of the unknown displacement       **
C**                   for K-th direction of node J                      **
C*W   LVSN(New) = The original node number of the new New-th node       **
C**   ============================================================     **
C**     J   | NRL(*,J)   |  NVAR(*,J)   |  LVSN(J) !!   NVAR(*,J)      **
C**   ------+------------+--------------+----------++-----------------    **
C**     1   |  1  1  0   |   0  0 17    |    4     !!    0  0  1         **
C**     2   |  0  0  0   |  11 12 13    |    5     !!    2  3  4         **
C**     3   |  0  0  0   |   5  6  7    |    3     !!    5  6  7         **
C**     4   |  1  0  0   |   0  1  2    |    6     !!    0  8  9         **
C**     5   |  1  0  0   |   0  3  4    |    2     !!    0 10 11         **
C**     6   |  0  0  0   |   8  9 10    |    7     !!   12 13 14         **
C**     7   |  0  0  0   |  14 15 16    |    1     !!   15 16 17         **
C**     8   |  1  1  0   |   0  0 18    |    8     !!    0  0 18         **
C**   ============================================================     **
C**   New program                        !    Original program          **
C**   -----------------------------------+-------------------------    **
      NFREE=0                            !    NFREE=0
      READ(NFILE,*) (LVSN(N),N=1,NODE)   !!
      DO 60 New=1,NODE                   !!   DO 60 J=1,NODE
      J=LVSN(New)                        !!
      DO 50 K=1,NDF                      !    DO 50 K=1,NDF
        IF(NRL(K,J).EQ.0) THEN           !      IF(NRL(K,J).EQ.0) THEN
          NFREE=NFREE+1                  !        NFREE=NFREE+1
          NVAR(K,J)=NFREE                !        NVAR(K,J)=NFREE
        ELSE                             !      ELSE
          NVAR(K,J)=0                    !        NVAR(K,J)=0
        ENDIF                            !      ENDIF
   50 CONTINUE                           ! 50 CONTINUE
   60 CONTINUE                           ! 60 CONTINUE
      RETURN                             !    RETURN
      END                                !    END
**************************************************************************
```

習題

1. 試修改$GENRCM$或$FEMRCM$副程式，將其中求$C(i)$之法改用方式二運算。

2. 試對$FEMRCM$副程式，做下列之修改：(1)增加陣列$NDEG(1{:}Nn)$，以儲存結點之度數。(2)以$LVSN(1{:}Nn)$做為某一結點之相鄰諸結點之暫存空間。因一個結點之相鄰結點總數小於Nn，因此若一次僅考慮一個結點的相鄰結點，則上述$LVSN$陣列即夠用。(3)由$EOJS$，$EOJP$，$JOES$與$JOEP$等資料，以$LVSN$做暫存空間，求各結點之度數$NDEG(1{:}Nn)$。(4)副程式$EMRRCM$中指令250至270間之排序改以$NDEG$為依據。

參考文獻

1. George, Alan and Liu, Joseph W-H, *Computer Solution of Large Sparse Positive Definite Systems,* Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1981.

2. 林聰悟，"Multiple Roots Reverse Cuthill Mckee Method for Reducing the Bandwidth and Profile of Finite Element Systems," Journal of the Chinese Institute of Engineers, Vol.12, No.2, 155-164, Taipei, 1989.

第二十章

隨機亂數之產生法

20.1 前言

在統計應用及模擬分析上常需用到符合某種分布函數之隨機亂數。利用隨機亂數亦可解決一些數學問題，如求面積或体積之積分問題等，一般稱為蒙地卡羅法(Monte Carlo method)。常用之分布函數有均勻分布函數、常態分布函數、對數常態分布函數、指數分布函數、芮萊分布函數、包松分布函數、雙分布函數等。其中以常態分布函數最常用，而以均勻分布函數之隨機亂數最容易產生。因此目前以電腦產生亂數最典型的辦法為先產生一系列之均勻分布之隨機亂數(簡稱均布亂數)，再由此產生其他各種分布之隨機亂數。本章將先介紹均布亂數之產生原理及方法，再介紹如何利用均布亂數產生其他各種分布之隨機亂數，並介紹最常用之常態分布隨機亂數之數種求法。

20.2 均勻分布隨機亂數之產生

產生均布亂數之最簡單而最常用之方法為線性調和法(Linear congruential method) (D.H.Lehmer 1948)。利用該法所產生的一系列隨機亂數定義於式(20.1)，此數列稱為線性調和數列(Linear congruential sequence, LCS)。

$$X_{i+1} = (aX_i + c) \bmod m \tag{20.1}$$

式中 m = 模數；$m > 0$。一般取$m = 2^j$或$m = 10^j$。

X_0 = 起始值；$m > X_0 \geq 0$。

a = 乘數；$m > a \geq 0$。

c = 增量；$m > c \geq 0$。

利用上法第$i+k$個值X_{i+k}亦可直接由X_i求得如下：

$$X_{i+k} \quad = \quad (a^k X_i + c(\frac{a^k - 1}{a - 1})) \bmod m \tag{20.2}$$

利用該法產生的數列達到一固定的數量後(設為n)，會產生與起始值相同的亂數，即$X_n = X_0$。根據上式之定義，隨後之亂數必然重覆前面所產生的數列，即$X_{n+i} = X_i$。因此為一具有固定週期的數列，n稱為該數列之週期。例如

以$X_0 = 2,\ a = 3,\ c = 1,\ m = 8$為例，其LCS如下(週期$n = 4$)：

$$2,\ 7,\ 6,\ 3,\ \ 2,\ 7,\ 6,\ 3,\ \ldots$$

以$X_0 = 2,\ a = 5,\ c = 1,\ m = 8$為例，其LCS如下(週期$n = 8$)：

$$2,\ 3,\ 0,\ 1,\ 6,\ 7,\ 4,\ 5,\ \ 2,\ 3,\ \ldots$$

理論上此數列之週期n一定不會大於模數m。在採用隨機亂數做統計分析時，如果所取隨機亂數之數目超過n則失去亂數之意義，因此週期n應儘可能大些。以下三個條件為使LCS之週期n等於其模數m之充分必要條件：

1. c與m互質。
2. $b\ \ (= a - 1)$為能整除m之所有質數之倍數。
3. b為4之倍數，若m為4之倍數。

例如上例中，$a = 5,\ c = 1,\ m = 8$即符合此充要條件，其週期$n = m = 8$。

由於$c = 0$可減少計算時間，因此亦常被採用。針對常用之$m = 2^j$以及$m = 10^j$之情形，使週期分別為2^{j-2}及$5 \cdot 10^{j-2}$之充分條件為：

1. $m = 2^j,\ \ j \geq 3$時：X_0為奇數，即不為2之倍數，$a \bmod 8 = \pm 3$。
2. $m = 10^j,\ j \geq 5$時：X_0不為2或5之倍數，$a \bmod 200$等於下列16個值或其負值：

$$3, 11, 13, 19, 21, 27, 29, 37, 53, 59, 61, 67, 69, 77, 83, 91$$

以$X_0 = 3,\ a = 3,\ c = 0,\ m = 8$為例，其$LCS$為：$\{3,\ 1,\ 3,\ 1,\ \ldots\}$。

對於所產生的亂數是否符合所求的分布函數通常可利用χ^2試驗法或柯莫哥洛夫－史密爾諾夫(Kolmogorov-Smirnov)試驗法試驗之，有興趣的讀者可參考[1]。一般應用對於a值的選取如遵照下列原則即可求得滿意的

亂數。

1. a 值接近 $\sqrt{m}$ 。
2. a 的最後一半的位元值為 0 與 1 的位元數大約相等。

20.3 產生均勻分布隨機亂數之副程式

由於模數 m 愈大愈好，故一般電腦程式均取 m 等於最大整數加 1 ，如此則使 mod 之運算自動完成。但因一般整數包含正負值，通常正負值以最高位元表示，正值最高位元為 0 ，負值為 1 。於線性調和法計算 X_{i+1} 時，雖然都是用正值做運算，但因電腦於乘算及加算後，所得位元會超過正值所佔之位元數，而可能在表示正負值的最高位元為 1 ，並被取回做為新亂數 X_{i+1} ，故會有負值出現。此時必須將此多餘的最高位元改為 0 。

以四個位元之整數為例，則最高位元為 0 之最大正值為 7 ，其位元值為 0111 ，因此取 $m = 8$ 。現以 $a = 5,\ c = 1,\ X_i = 5$ 為例，得 $X_{i+1} = aX_i + c = 26$ ，其位元值為 11010 ，電腦將自動取最後四個位元 1010 做為 X_{i+1} 。但該值應取 mod 8 而為 0010 。為了做此項處理必須瞭解電腦中負值的表示法。

負值的表示法目前被採用的有二種：其一為 1 的補餘法，其二為 2 的補餘法。大部分電腦採用 2 的補餘法；有些電腦則採用 1 的 補餘法。今以三個位元為例，由下表說明負值的表示法。

電　腦	所表示之整數值		去掉負值之最高位元	
位元值	1 的補餘法	2 的補餘法	1 的補餘法	2 的補餘法
000	0	0		
001	1	1		
010	2	2		
011	3	3		
100	–3	–4	–3 +3 = 0	–4 +3 +1 = 0
101	–2	–3	–2 +3 = 1	–3 +3 +1 = 1
110	–1	–2	–1 +3 = 2	–2 +3 +1 = 2
111	–0	–1	–0 +3 = 3	–1 +3 +1 = 3

所謂 1 的補餘法為將正值的每一位元單獨求其對於 1 的補餘數做為負值（以 1 減某單一位元值所得之差值稱該位元值之 1 的補餘數，即 0 與 1 互

為1的補餘數)。例如+2的位元值為010，其1的補餘數的位元值為101，即為負數−2。注意正值與負值互為1的補餘數。

所謂2的補餘法為將正值之1的補餘數加1後做為負值。例如+2的位元值為010，其1的補餘數為101，加1後的位元值為110，即為負數−2。注意除0與−4外，正值與負值互為2的補餘數。

如果想要將表示負值的最高位元的1改為0，而不改變其他位元時，例如將101改為001，對於1的補餘數，可將負值−2 (即101)加上最大之正整數3(即011)，即得1(即001)。對於2的補餘數，除須先將負值−3 (即101)加上最大之正整數3(即011)外，須再加1方可得1(即001)。

下表所示之程式為產生單位均布亂數之函數，雖然相當簡單但使用該程式時，必須先瞭解所用電腦之負值表示法及其位元數，再參考上述說明做適當修改。

［表二］ 產生均布亂數之程式

```
************************************************************************
      FUNCTION RANDOM(IX)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA MAXN/2147483647/
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Random number by linear congruential method              |    **
C**   | For 32 bits INTEGER*4: C=51605, M-1=2147483647           |    **
C**   | For 16 bits INTEGER*2: C=405,   M-1=32767                |    **
C**   | IX = MOD( 51605*IX+3, 2**32)                             |    **
C**   +----------------------------------------------------------+    **
      IX = 51605*IX + 3
C**   +----------------------------------------------------------+    **
C**   | Next statement changes the sign bit from 1 to 0          |    **
C**   | That is : IX = MOD( 51605*IX+3, 2**31)                   |    **
C**   |           for 1'S complement, cancel +1                  |    **
C**   +----------------------------------------------------------+    **
      IF(IX .LT. 0) IX = IX+MAXN+1
C**   +----------------------------------------------------------+    **
C**   | RANDOM = Uniform distributed number between 0 and 1      |    **
C**   +----------------------------------------------------------+    **
      RANDOM = IX/2147483647.0D0
C**   +----------------------------------------------------------+    **
C**   | If you can not avoid overflow error                      |    **
C**   | Use follows for 32 bits INTEGER*4:                       |    **
C**   +----------------------------------------------------------+    **
C     IX = MOD(405*IX+3,32768)
C     RANDOM = IX/32767.0D0
      RETURN
      END
************************************************************************
```

20.4 均勻分布隨機亂數之其他求法

另二個亦常被採用的方法為加算調和法 (Additive congruential method) 及 M 氏演算法(Algorithm M)。茲簡述其做法於後。

一、加算調和法之計算式定義如下：

$$X_{i+1} = (X_i + X_{i-k}) \bmod m, \qquad k \geq 1 \tag{20.3}$$

此方法之週期 $p = \alpha m/2$ ，其中 $1 < \alpha < 2^k - 1$ 。當 $k = 1$ 之特例時，則為有名之懷柏納西(Fabonacci)數列： $F_{i+1} = F_i + F_{i-1}$ 。

二、M 氏演算法由MacLaren與Marsaglin提出，步驟如下：

先產生 k 個 (k 一般可選約100以下之值) 線性調和數列 X_i 存於陣列 $V(0 : k-1)$ 。

M1. 產生二個線性調和亂數 X, Y (X, Y 最好分別由週期為互質之二個數列產生)。

M2. 由 Y 決定陣列指標 $j = INT(k * Y/m)$ ，注意其中 m 為 Y 數列之模數，故 $0 \leq j < k$ 。

M3. 令 $V(j)$ 為所求之亂數，並設 $V(j) = X$ 。

以 $k = 4$ 為例，並取 $X_{i+1} = (5X_i + 5) \bmod 8,\ X_0 = 1$ 及 $Y_{i+1} = (5Y_i + 3) \bmod 8,\ Y_0 = 1$ 。最初設定陣列 $\{V\} = \{1, 2, 7, 0\}$ ，隨後之數列 $\{X\} = \{5, 6, 3, 4, 1, 2, 7, 0\}$ ，數列 $\{Y\} = \{1, 0, 3, 2, 5, 4, 7, 6\}$ ，$\{j\} = \{kY/m\} = \{0, 0, 1, 1, 2, 2, 3, 3\}$ ，因此可得亂數 $\{x\} = \{1, 5, 2, 3, 7, 1, 0, 7, 6, 5, 4, 3, 2\}$ 。

此數列將重覆出現最後8個亂數。因此最好須使 $X,\ Y$ 兩數列之週期互質。此例所得最後週期性之數列正巧是一個遞減等差級數。此種巧合雖非特別造成，但可看出某些演算法雖然相當複雜，表面上看來似乎不易得到某種簡單之關係，事實上反而容易碰到巧合的情況。

20.5 其他分布之隨機亂數之求法

利用均勻分布隨機亂數求其他分布之隨機亂數之一般方法主要有四類：一為反函數法 (Invert method)；二為捨去選取法 (Reject select method)；三為混合法 (Mixed method)；四為組成法 (Composition method)。前二

個方法各有其優劣及不能適用的情況；後二個方法兼取前二法之優點；第三法較第四法之程式容易撰寫，但第四法運算效率最好。

一、反函數法

若隨機亂數x之密度函數為$f(x)$，則其分布函數為 $F(x)=\int_{-\infty}^{x} f(x)\,dx$。今若定義一隨機亂數$u=F(x)$，則其密度函數可由$f_u(u)du=f(x)dx$之關係求得為$f_u(u)=f(x)/F'(x)=1$。亦即$u$為0至1間均勻分布之隨機亂數(簡稱單位均布亂數)。因此可証知各種分布之隨機亂數均可經轉換而得一單位均布亂數u。

而反函數法即為先產生一個單位均布亂數u，再解$F(x)=u$，或由其反轉換$x=F^{-1}(u)$，求得x即為所求之隨機亂數(其密度函數為$f(x)$，分布函數為$F(x)$)。能夠直接用反函數法求隨機亂數的分布函數有偉柏(Weibull)分布，指數(Exponential)分布，芮萊(Rayleigh)分布，柯其(Cauchy)分布等，茲將其密度函數及分布函　數彙列如下：

1.偉柏分布函數$(\sigma>0,\ \eta>0,\ x\geq 0)$

$$f(x)=\frac{\eta}{\sigma}(\frac{x}{\sigma})^{\eta-1}e^{-(\frac{x}{\sigma})^{\eta}}$$
$$F(x)=1-e^{-(\frac{x}{\sigma})^{\eta}}$$
$$F^{-1}(u)=\sigma\ (-\ln(1-u))^{1/\eta}$$

2.指數分布函數$(\sigma>0,\ x\geq 0)$

$$f(x)=\frac{1}{\sigma}e^{-\frac{x}{\sigma}}$$
$$F(x)=1-e^{-\frac{x}{\sigma}}$$
$$F^{-1}(u)=-\sigma\ln(1-u)$$

3.芮萊隨機亂數$(\sigma>0,\ x\geq 0)$

$$f(x)=\frac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}}$$
$$F(x)=1-e^{-\frac{x^2}{2\sigma^2}}$$
$$F^{-1}(u)=\sigma\sqrt{-2\ln(1-u)}$$

4.部分芮萊隨機亂數$(\sigma>0,\ x\geq a)$

$$f(x)=\frac{x}{\sigma^2}e^{-\frac{x^2-a^2}{2\sigma^2}}$$

$$F(x) = 1 - e^{-\frac{x^2-a^2}{2\sigma^2}}$$
$$F^{-1}(u) = \sqrt{a^2 - 2\sigma^2 \ln(1-u)}$$

5.柯其隨機亂數($\sigma > 0$)

$$f(x) = \frac{1}{\sigma\pi(1 + \frac{(x-\mu)^2}{2\sigma^2})}$$
$$F(x) = \frac{1}{2} + \frac{1}{\pi}\tan^{-1}\frac{x-\mu}{\sigma}$$
$$F^{-1}(u) = \mu + \sigma\tan[\pi(u-0.5)]$$

對於離散(Discrete)密度函數亦均可用反函數法求其隨機亂數。 今以二項式(Binomial)分布為例：

$$\begin{aligned} f(x) &= \binom{n}{x} p^x (1-p)^{(n-x)} \\ F(x) &= \sum_{k=0}^{x} f(k) \end{aligned}$$

先求單位均布亂數u，若$F(x-1) < u \le F(x)$，則x即為所求之隨機亂數。

利用反函數法時最好預先將$x = 0, 1, \ldots, n$之所有$F(x)$值算出放在一陣列內，再以二分尋值法(Binary search)由u找出符合$F(x-1) < u \le F(x)$之x。

二、捨去選取法

捨去選取法適用於隨機亂數有一定之範圍者，否則須選定一個近似之範圍，設為a至b，使$1 - F(b) + F(a)$小至可以忽略的程度。以及選取一個密度函數的上限值設為c，使$f(x) \le c$。即可照下列步驟求隨機亂數x。

1.先產生二個單位均布亂數$u_1,\ u_2$。
2.計算$x = a + (b-a)u_1,\ y = f(x),\ z = cu_2$。
3.若$z \le y$，則x即為所求之隨機亂數；否則捨去x，回步驟1.重算。

此方法原理很簡單，隨機亂數被選取的機率與$f(x)$之大小成正比。但該法有一缺點，為$a,\ b$間之範圍如太小，則結果較不準確。且範圍外之亂數將永遠不會被選上。如$a,\ b$間之範圍太大，則選取率不高，即效率較差。

三、混合法

反函數法對於能直接寫出反函數$F^{-1}(u)$之分布函數均可正確而快速求出。否則即須求解非線性方程式$u = F(x)$的根，將較費時，尤其當$F(x)$之積分無法以解析法求得時(此時常可表示成誤差函數 (Error function)再以多項式近似)。

捨去選取法之求解速度不會因分布函數之特性而有很大的影響，但其做法因須對隨機亂數選取一固定範圍而僅為近似之解，且範圍外之值將被去除而不出現，為其缺點。

對於不能直接寫出反函數或非有限範圍之亂數，可考慮用下列二種混合辦法求取：

(1)選取一個比較容易求反函數之分布函數，其密度函數為$\bar{f}(x)$(亂數範圍相同)，並選取一常數c，使$f(x) \le c\,\bar{f}(x)$。即可照下列步驟求$f(x)$之隨機亂數：

1. 產生$\bar{f}(x)$之隨機亂數x，並計算$z = c\,\bar{f}(x)$。
2. 計算$y = f(x)$。
3. 產生0至1間均布亂數u。
4. 若$y \ge z\,u$，則x為所求之隨機亂數；否則捨去x，回步驟1.重算。

(2)將隨機亂數之範圍分為$N+1$段，設各段間以$x_1, x_2, ..., x_N$為界。對第2至N段：求各段$f(x)$之上限值c_i及下限值與上限值之比d_i，即使$d_i\,c_i \le f(x) \le c_i$。對第1及$N+1$段：各選取較容易求反函數之分布函數，稱參考分布函數，其密度函數為$\bar{f}_i(x)$，並選c_i，使$f(x) \le c_i\,\bar{f}_i(x)$；或使該二段之出現機率小至可以忽略的程度。因此，可不產生該二段內之亂數，即假設$c_i = 0$或$F_1 = 0,\ F_N = 1$。然後計算$F_i = F(x_i)$；或計算$S_1 = c_1,\ S_i = S_{i-1} + c_i(x_i - x_{i-1}),\ S_{N+1} = S_N + c_{N+1}$，及$\bar{F}_i = S_i/S_{N+1}$。即可照下列步驟求$f(x)$之隨機亂數：

1. 先產生一個單位均布亂數u_0。
2. 由下列條件之一決定隨機亂數之區間為x_{i-1}至x_i。
   (a) $F_{i-1} < u_0 \le F_i$
   (b) $\bar{F}_{i-1} < u_0 \le \bar{F}_i$
3. 再產生二個單位均布亂數$u_1,\ u_2$。

4. 對第2至N段：

   計算$x = x_{i-1} + (x_i - x_{i-1})u_1$。若$d_i \geq u_2$，則$x$為所求之隨機亂數；否則令$z = c_i$。

   對第1及$N+1$段：

   直接以反函數法由u_1求得x，並計算$z = c_i \bar{f}_i(x)$。

5. 計算$y = f(x)$。

6. 若$y \geq z\,u_2$，則x為所求之隨機亂數；否則捨去x，以2(a)為條件時回步驟3.重算；以2(b)為條件時回步驟1.重算。

以下即為對應之程式，同時可適用於捨去選取法及混合法。

［表三］ 產生任意亂數之程式

```
**************************************************************************
      FUNCTION RANFUN(IX)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
C**   ============================================================    **
C**   General program for any distributed random number               **
C**   By mixed method : invert method & reject-select method          **
C**   ------------------------------------------------------------    **
C**   Given : N,X(1:N),F(1:N),D(2:N),C(1:N+1) & following routines    **
C**   DENFUN(XX) = Density function of XX                             **
C**   FINVL(DIS,VAR,DEN) & FINVR(DIS,VAR,DEN)                         **
C**   = Referential distribution function for first & last sections   **
C**     (VAR can be find by invert method from DIS)                   **
C**   Input : DIS = Distribution function                             **
C**   Output: VAR = Random number                                     **
C**           DEN = Density function                                  **
C**   ================================================              **
C**    N   |    Conditions    |    Reqd. Functions                    **
C**   =====+==================+======================                **
C**   N=0  |                  |  DENFUN, FINVL                        **
C**   -----+------------------+----------------------                **
C**        |       F(1)=0     |  DENFUN,        FINVR                 **
C**   N=1  |   0 < F(1) < 1   |  DENFUN, FINVL, FINVR                 **
C**        |       F(1)=1     |  DENFUN, FINVL                        **
C**   -----+------------------+----------------------                **
C**        |  0=F(1)  F(N)=1  |  DENFUN                               **
C**        |  0=F(1)  F(N)<1  |  DENFUN,        FINVR                 **
C**   N>1  |  0<F(1)  F(N)<1  |  DENFUN, FINVL, FINVR                 **
C**        |  0<F(1)  F(N)=1  |  DENFUN, FINVL                        **
C**   ================================================              **
C**   +------------------------------------------------------------+  **
C**   | IF F(I) = Referential distribution function                |  **
C**   | Here is for GO TO 50                                       |  **
C**   +------------------------------------------------------------+  **
   50 CONTINUE
C*1   ------------------------------------------------------------    **
      U0 = RANDOM(IX)
C*2   +------------------------------------------------------------+  **
C**   | Find I by binary search such that                          |  **
```

```
C**   | F(I-1) < U0 <= F(I)                                          |    **
C**   +-------------------------------------------------------------+    **
      IBGN=1
      IEND=N+1
   10 IF (IBGN.GE.IEND) THEN
        I=IEND
      ELSE
        I=(IBGN+IEND)/2
        IF (U0.LT.F(I)) THEN
          IEND=I
          GO TO 10
        ELSE IF (U0.GT.F(I)) THEN
          IBGN=I+1
          GO TO 10
        ENDIF
      ENDIF
C**   +-------------------------------------------------------------+    **
C**   | IF F(I) = Actual distribution function                      |    **
C**   | Here is for GO TO 50                                        |    **
C**   +-------------------------------------------------------------+    **
C**50 CONTINUE
C*3   ---------------------------------------------------------------    **
      U1 = RANDOM(IX)
      U2 = RANDOM(IX)
C*4   ---------------------------------------------------------------    **
      IF (I.LT.2) THEN
        CALL FINVL (U1,XX,DEN)
        ZZ = C(1)   * DEN
      ELSE IF (I.GT.N) THEN
        CALL FINVR (U1,XX,DEN)
        ZZ = C(N+1) * DEN
      ELSE
        XX = X(I-1) + (X(I)-X(I-1)) * U1
        IF (D(I) .GE. U2) THEN
          RANFUN = XX
          RETURN
        ENDIF
        ZZ = C(I)
      ENDIF
C*5   ---------------------------------------------------------------    **
      YY = DENFUN (XX)
C**   IF (YY .GT. ZZ) STOP 1111
C*6   +-------------------------------------------------------------+    **
C**   | Reject XX                                                   |    **
C**   +-------------------------------------------------------------+    **
      IF (YY .LT. ZZ*U2) GO TO 50
C**   +-------------------------------------------------------------+    **
C**   | Select XX                                                   |    **
C**   +-------------------------------------------------------------+    **
      RANFUN = XX
      RETURN
      END
*************************************************************************
```

上列程式在某些情況下不須用到副程式$FINVL$或$FINVR$時亦須給一虛設之副程式，即僅用$SUBROUTINE\ \ FINVL(DIS,VAR,DEN)$及$END$兩行。

四、組成法

將密度函數下的面積分割成許多較大塊之矩形及矩形上方之不規則小塊，每塊之面積大小做為點出現在該塊內之機率。根據此機率，由一單位均布亂數之大小可決定點應出現在那一塊內。若該塊為一矩形，其左右邊之水平座標為$x_{i-1},\ x_i$，則由另一單位均布亂數u可決定點出現在該塊內之水平座標為$x = x_{i-1} + (x_i - x_{i-1}) * u$，該值即可做為所考慮之密度函數之隨機亂數。若該塊為一不規則小塊，則可用前述捨去選取法決定點落在該小塊內之水平座標並取為隨機亂數。當矩形所佔面積接近1時，該法之平均效率會遠較其他方法為佳。參考文獻[2]即以此觀念提供一個產生常態分布亂數之快速方法。詳細做法亦可參考[1]。

利用該法寫程式須預做許多分析計算工作，且需用到許多儲存常數之記憶空間，故除非需要很高的效率，一般很少採用該法。

20.6 常態分布之隨機亂數之求法

由於常態分布之隨機亂數最常被使用，其他分布之亂數，如要求精度不高，亦常以常態分布之亂數近似，因此以下特別介紹數種該亂數之求法。

一、利用中心極限定理之求法

按中心極限定理：如果某隨機亂數$x = \sum_{i=1}^{n} u_i$，其中n個u_i為獨立之亂數，則當n很大時，x之分布為常態分布，其平均值$E[x] = \sum_{i=1}^{n} E[u_i]$，變異數$V[x] = \sum_{i=1}^{n} V[u_i]$。

若取u_i為單位均布亂數，則$E[u_i] = 1/2,\ V[u_i] = \int_0^1 (u - 0.5)^2 du = 1/12$。若取$n = 12$時，可得$E[x] = 6,\ V[x] = 1$。以下即為對應之程式。

［表四］ 利用中心極限定理產生常態分布亂數

```
************************************************************************
      FUNCTION GAUSS(AMEAN,SIGMA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA IX/11111/
C**   ============================================================    **
C**   GAUSS  = Normal distributed random number                       **
C**   AMEAN  = Mean value of the random number                        **
```

```
C**   SIGMA  = Standard diviation of the random number               **
C**          = SQRT(variance)                                        **
C**   ============================================================   **
      SU = -6.0
      DO 10 I=1,12
   10 SU = SU + RANDOM(IX)
C**   +-------------------------------------------------------+      **
C**   | SU = Unit std. div. zero mean normal dist.            |      **
C**   +-------------------------------------------------------+      **
      GAUSS1 = AMEAN + SIGMA * SU
      RETURN
      END
*************************************************************************
```

此法之缺點為：n若太大則速度慢；太小則誤差大。就以$n = 12$為例，$|x - E[x]|$大於3倍標準偏差之亂數之出現率就過於偏低，而$|x - E[x]|$大於6倍標準偏差之亂數則永不出現。

二、直接求法—利用二維分布函數之反函數之求法

今考慮一個二維之密度函數，式(20.4)以直角座標表示，式(20.5)以極座標表示，二者所表示之密度函數完全相等。

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{20.4}$$

$$g(r,\theta) = \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \tag{20.5}$$

上式所示之密度函數之特性為出現在距原點等距離之圓上之密度函數值相同。如果我們只要知道出現的點與原點之距離r，而不要知道其方向θ，則對於距離r之密度函數$g_r(r)$可由下式求得

$$g_r(r) = \int_0^{2\pi} g(r,\theta)\, r\, d\theta = 2\pi r g(r,\theta) = \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \tag{20.6}$$

此密度函數為芮萊分布。而芮萊分布已於前面說明過，可以用反函數法直接求得，即$r = \sigma\sqrt{-2\ln(1-u)}$，式中$u$為單位均布亂數。

如果我們只要知道出現的點對原點之方向θ，而不要知道其與原點之距離r，則由式(20.5)知$g(r,\theta)$只與距離r有關而與θ無關，故對於方向角θ之密度函數$g_\theta(\theta)$可由下式求得

$$g_\theta(\theta) = \int_0^{\infty} g(r,\theta)\, r\, dr = \frac{1}{2\pi} \tag{20.7}$$

此密度函數為均勻分布。

如果我們只要知道出現的點之x座標值，而不要知道其y座標值，則其密度函數$f_x(x)$可由下式求得

$$f_x(x) \;=\; \int_{-\infty}^{\infty} f(x,y)\,dy = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \qquad (20.8)$$

此密度函數為常態分布。

綜合式(20.6)至式(20.8)知，利用芮萊分布可得r及均勻分布可得θ，亦即可得式(20.4)或式(20.5)之分布之隨機位置為(r,θ)，將其投影至x軸，即$x = r\,cos\,\theta$，即可得常態分布之隨機亂數。以下即為對應之程式。

［表五］ 利用芮萊分布產生常態分布亂數

```
************************************************************************
      FUNCTION GAUSS(AMEAN,SIGMA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA IX/11111/
C**   ============================================================    **
C**   GAUSS  = Normal distributed random number                       **
C**   AMEAN  = Mean value of the random number                        **
C**   SIGMA  = Standard diviation of the random number                **
C**          = SQRT(variance)                                         **
C**   ------------------------------------------------------------    **
C**   UR     = RANDOM(IX)                                             **
C**   UA     = RANDOM(IX)                                             **
C**   GAUSS2 = AMEAN + SIGMA * SQRT(-2*ALOG(UR))*COS(6.2831853*UA)    **
C**   ------------------------------------------------------------    **
C**   Note that : UR (0:1) can be replaced by RR  (RR=U1*U1+U2*U2)    **
C**               COS(6.2831853*UA) can be replaced by U1/SQRT(RR)    **
C**   ============================================================    **
   10 U1 = RANDOM(IX)*2.0 - 1.0
      U2 = RANDOM(IX)
      RR = U1*U1 + U2*U2
      IF(RR.GT.1.0) GOTO 10
      GAUSS2 = AMEAN + SIGMA * DSQRT(-2.0*DLOG(RR)/RR)*U1
      RETURN
      END
************************************************************************
```

三、利用混合法

以下列舉三種方式主要目的係做為混合法之範例，以為其他分布函數之參考。實際求常態分布之隨機亂數仍以採用前述二法程式較為簡潔且效率亦不太差。

下列程式係將隨機亂數之範圍以$x_1 = -a,\ x_2 = a$分為三段，前後二段以部分芮萊隨機亂數做為參考分布函數。因

$$\frac{f(x)}{\bar{f}(x)} = \frac{\sigma}{\sqrt{2\pi}x} e^{-\frac{a^2}{2\sigma^2}}$$

為遞減函數，故取下列c值即可使$f(x) \le c\,\bar{f}(x)$。

$$c = f(a)/\bar{f}(a) = \frac{\sigma}{\sqrt{2\pi}a} e^{-\frac{a^2}{2\sigma^2}}$$

以下即為對應之程式。

［表六］ 利用混合法產生常態分布亂數之一

```
*************************************************************************
      FUNCTION GAUSS(AMEAN,SIGMA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
      DATA IX/-1/
C**   ============================================================    **
C**   Use mixed method (invert method & reject-select method)         **
C**   to find the normal distributed number                           **
C**   ============================================================    **
      IF(IX.LT.0) THEN
         IX=11111
C**      +-----------------------------------------------------+    **
C**      | Set N,X(1:N),F(1:N),D(2:N),C(1:N+1)                 |    **
C**      | for unit std. div. zero mean normal dist.           |    **
C**      +-----------------------------------------------------+    **
         A=1.0
         N=2
         X(1)=-A
         X(2)=+A
         C(1)=DEXP(-0.5*X(1)**2)/(DSQRT(6.283185307179586D0)*DABS(X(1)))
         C(2)=1.0/DSQRT(6.283185307179586D0)
         C(3)=C(1)
         D(2)=EXP(-0.5*X(1)**2)
C**      +-----------------------------------------------------+    **
C**      | Calculate referential F(1:N) from X(1:N) and C(1:N+1) |    **
C**      | Follows are general procedure (for N>0)             |    **
C**      +-----------------------------------------------------+    **
C**      F(1)=C(1)                                                   **
C**      DO 30 I=2,N                                                 **
C**30    F(I)=F(I-1)+(X(I)-X(I-1))*C(I)                              **
C**      SNP1=F(N)+C(N+1)                                            **
C**      DO 50 I=1,N                                                 **
C**50    F(I)=F(I)/SNP1                                              **
C**      Next st. is to prevent choose from 1st section              **
C**      IF (F(1).EQ.0.0) F(1)=-0.00001                              **
C**      +-----------------------------------------------------+    **
C**      | But for this special case we use                    |    **
C**      +-----------------------------------------------------+    **
         F(1)=C(1)/(C(1)+2.*A*C(2)+C(3))
         F(2)=1.0-F(1)
      ENDIF
C**   ------------------------------------------------------------    **
      GAUSS = AMEAN + SIGMA * RANFUN(IX)
      RETURN
      END
*************************************************************************
      FUNCTION DENFUN(X)
C**   ============================================================    **
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
C**   COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
      DATA RSQR2P/0.3989422804014327D0/
C**   ============================================================    **
C**   DENFUN = density function of a random number X                  **
C**   For unit std. div. zero mean normal dist.                       **
C**   ============================================================    **
      DENFUN = RSQR2P * DEXP(-0.5*X*X)
      RETURN
      END
*************************************************************************
      SUBROUTINE FINVL(DIS,VAR,DEN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
C**   ============================================================    **
C**   Input : DIS = Distribution function                             **
C**   Output: VAR = Random number                                     **
C**           DEN = Density function                                  **
C**   ------------------------------------------------------------    **
C**   Partial Rayleigh distribution (VAR .GE. A) for SIGMA = 1        **
C**   For better result use A .GE. SIGMA                              **
C**   ============================================================    **
      ASQ = X(1)**2
      XSQ = ASQ-2.*DLOG(DIS)
      VAR = -DSQRT(XSQ)
      DEN = -VAR * DEXP(-0.5*(XSQ-ASQ))
      RETURN
      END
*************************************************************************
      SUBROUTINE FINVR(DIS,VAR,DEN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
C**   ============================================================    **
      ASQ = X(N)**2
      XSQ = ASQ-2.*DLOG(DIS)
      VAR = DSQRT(XSQ)
      DEN = VAR * DEXP(-0.5*(XSQ-ASQ))
      RETURN
      END
*************************************************************************
```

若將隨機亂數之範圍以$x_1 = 0$分為二段，二段均以指數隨機亂數做為參考分布函數。因$f(x)/\bar{f}(x) = \frac{\bar{\sigma}}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}+\frac{x}{\bar{\sigma}}}$在$x = a = \sigma^2/\bar{\sigma}$時為極大，故取下列$c$值即可使$f(x) \le c\,\bar{f}(x)$。

$$c = f(a)/\bar{f}(a) = \frac{\bar{\sigma}}{\sqrt{2\pi}\sigma} e^{\frac{\sigma^2}{2\bar{\sigma}^2}}$$

以下即為對應之程式。

［表七］ 利用混合法產生常態分布亂數之二

```
*************************************************************************
      FUNCTION GAUSS(AMEAN,SIGMA)
```

```
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
      DATA IX/-1/
C**   ============================================================    **
C**   Use mixed method (invert method & reject-select method)         **
C**   to find the normal distributed number                           **
C**   ============================================================    **
      IF(IX.LT.0) THEN
         IX=11111
C**      +----------------------------------------------------+    **
C**      | Set N,X(1:N),F(1:N),D(2:N),C(1:N+1)                |    **
C**      | for unit std. div. zero mean normal dist.          |    **
C**      +----------------------------------------------------+    **
         N=1
         X(1)=0.0
         C(1)=DEXP(0.5D0)/DSQRT(6.283185307179586D0)
         C(2)=C(1)
         F(1)=0.5
      ENDIF
C**   ------------------------------------------------------------    **
      GAUSS = AMEAN + SIGMA * RANFUN(IX)
      RETURN
      END
*************************************************************************
      FUNCTION DENFUN(X)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
      DATA RSQR2P/0.3989422804014327D0/
C**   ============================================================    **
C**   DENFUN = density function of a random number X                  **
C**   For unit std. div. zero mean normal dist.                       **
C**   ============================================================    **
      DENFUN = RSQR2P * DEXP(-0.5*X*X)
      RETURN
      END
*************************************************************************
      SUBROUTINE FINVL(DIS,VAR,DEN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
C**   ============================================================    **
C**   Input : DIS = Distribution function                             **
C**   Output: VAR = Random number                                     **
C**           DEN = Density function                                  **
C**   ------------------------------------------------------------    **
C**   Exponential distribution (VAR .GE. 0)
C**   MEAN = Standard diviation = SIGMA = 1
C**   ============================================================    **
      VAR = DLOG(DIS)
C**   DEN = DEXP(VAR)
      DEN = DIS
      RETURN
      END
*************************************************************************
      SUBROUTINE FINVR(DIS,VAR,DEN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
```

```
C**   ============================================================    **
      VAR = -DLOG(DIS)
C**   DEN = DEXP(-VAR)
      DEN = DIS
      RETURN
      END
*************************************************************************
```

若將隨機亂數之範圍取至使二個最外分段之出現機率小至可以忽略的程度，例如取$x_1 = -6\sigma,\ F(x_1) = 10^{-9}$，因此可不產生該二段內之亂數，即假設$F_1 = 0,\ F_N = 1$，則可不必用到副程式$FINVL,\ FINVR$。以下即為對應之程式，注意該程式因不會叫用$FINVL,\ FINVR$，故所列該二程式均僅二行。

［表八］ 利用混合法產生常態分布亂數之三

```
*************************************************************************
      FUNCTION GAUSS(AMEAN,SIGMA)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
      DATA IX/-1/
C**   ============================================================    **
C**   Use mixed method (invert method & reject-select method)         **
C**   to find the normal distributed number                           **
C**   ============================================================    **
      IF(IX.LT.0) THEN
         IX=11111
C**      +---------------------------------------------------+    **
C**      | Set N,X(1:N),F(1:N),D(2:N),C(1:N+1)               |    **
C**      | for unit std. div. zero mean normal dist.         |    **
C**      +---------------------------------------------------+    **
         DX = 0.25
         N = 49
C**      -----------------------------------------------------    **
         NH = (N+1)/2
         N = NH*2-1
         DO 10 I=1,NH
         X(I) = (I-NH)*DX
         C(I) = DENFUN(X(I))
         X(N-I+1) = -X(I)
   10    C(N-I+2) = C(I)
         DO 20 I=2,NH
         D(I) = C(I-1)/C(I)
   20    D(N-I+2) = D(I)
C**      +---------------------------------------------------+    **
C**      | Neglect the infinite sections                     |    **
C**      +---------------------------------------------------+    **
         C(1)=0.0
         C(N+1)=0.0
C**      +---------------------------------------------------+    **
C**      | Calculate referential F(1:N) from X(1:N) and C(1:N+1) |  **
C**      | Follows are general procedure (for N>0)           |    **
C**      +---------------------------------------------------+    **
         F(1)=C(1)
```

```
         DO 30 I=2,N
   30    F(I)=F(I-1)+(X(I)-X(I-1))*C(I)
         SNP1=F(N)+C(N+1)
         DO 50 I=1,N
   50    F(I)=F(I)/SNP1
C**      Next st. is to prevent choose from 1st section              **
         IF (F(1).EQ.0.0) F(1)=-0.00001
      ENDIF
C**   ------------------------------------------------------------   **
      GAUSS = AMEAN + SIGMA * RANFUN(IX)
      RETURN
      END
*************************************************************************
      FUNCTION DENFUN(X)
C**   ============================================================   **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   COMMON/RAN000/ N,X(99),F(99),D(99),C(100)
      DATA RSQR2P/0.3989422804014327D0/
C**   ============================================================   **
C**   DENFUN = density function of a random number X                 **
C**   For unit std. div. zero mean normal dist.                      **
C**   ============================================================   **
      DENFUN = RSQR2P * DEXP(-0.5*X*X)
      RETURN
      END
*************************************************************************
      SUBROUTINE FINVL(DIS,VAR,DEN)
      END
*************************************************************************
      SUBROUTINE FINVR(DIS,VAR,DEN)
      END
*************************************************************************
```

以上五個程式產生亂數之時間比例依序約為$7:6:10:12:5$。

習題

1. 試按加算調和法或M氏演算法寫一副程式以產生單位均布亂數。
2. 試以反函數法寫一副程式以產生自選之離散密度函數之亂數。
3. 試以混合法寫一副程式以產生自選之密度函數之亂數。
4. 對數常態分布之亂數y可由常態分布之亂數x推求。以下為二者之變數、平均數及標準偏差之關係。試寫一副程式為之。

$$\begin{aligned} \sigma_x^2 &= \ln\left(\frac{\sigma_y^2}{\mu_y^2}+1\right) \\ \mu_x &= \ln\mu_y - \frac{\sigma_x^2}{2} \\ y &= e^x \end{aligned}$$

參考文獻

1. Knuth, D. E., *Seminumerical Algorithms,* Vol.2 of *The Art of Computer Programming,* Addison-Wesley, 1969.

2. MacLaren, M. D. and Marsaglia, G., "Uniform Random Number Generators," Journal of ACM, Vol.12, No.1, January 1965.

3. Marsaglia, G., MacLaren, M. D. and Bray, T. A., "A Fast Procedure for Generating Normal Random Variables," Comm. of ACM, Vol.7, 1964.

第二十一章

函數的極值問題

21.1 前言

求函數（此函數又稱為目標函數）的極小值最早用於曲線之近似，使近似之曲線與已知點間之距離之平方和為最小。如果欲予近似之曲線係由一些已知函數之線性組合，則其組合係數可直接由一組線性聯立方程式求得（詳見第九章）；否則就不那麼簡單，一般均須經過許多反覆試算方可求得，這些方法將於本章介紹。因極大值變號後即為極小值，除非特別提及，本章以後各節均將以極小值為例做說明。求極值問題可依其複雜度分為：(1)單變數函數之極值問題；(2)多變數函數之極值問題；(3)有限制條件之極值問題。因有限制條件之極值問題常須解一系列之無限制條件之多變數函數之極值問題，而多變數函數之極值問題，又須以線性搜尋方法解一系列之單變數函數之極值問題。有限制條件之極值問題轉化為一系列之無限制條件之極值問題之方式一般分為二大類：(1)拉格蘭治乘數法；(2)懲罰函數法。而在線性搜尋方法中須決定搜尋方向或路徑。搜尋方向之求法頗多，一般可分為五類：(1)最陡方向法；(2)共軛方向法；(3)牛頓法，(4)準牛頓法；(5)單變數法。而搜尋路徑可為(1)直線，亦可為(2)曲線。有限制條件之極值問題還有另一類解法為轉化成一系列之線性規劃問題以求解。由此可見極值問題之求解方法確實琳瑯滿目，上述各種做法在本章亦僅能做概要性的介紹。共軛方向法與準牛頓法之有關公式為數不少且變化多端，公式之使用雖然簡單但其推導則非淺顯易得，因此特以二節十數頁之篇幅作一系統性之推導。

21.2 單變數函數之極值

因實用上大部分函數均為多變數函數，求單變數函數之極值似乎少有實用價值；但事實上，幾乎所有多變數函數之極值問題均係反覆利用單變數函數之極值問題求解。因為，當多變數函數自一已知起始點沿著一已知直線方向（即固定各變數之比值）尋找函數在線上之極值時，即為單變數（設為移動距離）之極值問題。當然找到的點不一定是函數之極值所在之點，簡稱極點，因此須改變直線方向（方向之決定將於下節說明）繼續尋找，一直找到或接近極點為止。

在還沒有介紹求極值之方法以前，先說明一個簡單原理：當函數$F(X)$在某一區間（設為$X_L \le X \le X_R$）為連續時，若X_M在此區間內，且$F(X_M) < F(X_L)$及$F(X_M) < F(X_R)$，則$F(X)$在(X_L, X_R)間有極小值。

下列數法除第四法外，均係針對函數在(X_L, X_R)間有極小值之情形，考慮如何縮小(X_L, X_R)之範圍，以漸次求得極點或其近似位置。

一. 若已知(X_L, X_R)二點間函數有極小值時，令$X_A = (X_L + X_R)/2$，$X_B = X_A + dX$（dX為任意很小之值），計算$F(X_A)$，$F(X_B)$。若$F(X_A) < F(X_B)$，則極小值之區間在(X_L, X_B)，因此令新的$X_R = X_B$；反之若$F(X_A) > F(X_B)$，則極小值之區間在(X_A, X_R)，因此令新的$X_L = X_A$。本法每次計算二個函數值，區間減少一半，效率較下法差，很少被採用。

二. 若已知(X_L, X_R)二點間函數有極小值時，令$X_A = 0.618 * X_L + 0.382 * X_R$，$X_B = 0.382 * X_L + 0.618 * X_R$，計算$F(X_A)$，$F(X_B)$。若$F(X_A) < F(X_B)$，則極小值之區間在$(X_L, X_B)$，因此令新的$X_R = X_B$；反之若$F(X_A) > F(X_B)$，則極小值之區間在$(X_A, X_R)$，因此令新的$X_L = X_A$。本法於第一次以後之$X_A$及$X_B$中均有一點與前一次區間之$X_B$或$X_A$相同，因此每次只需再計算一個函數值，而區間減為原來之0.618，效率較上法佳。以上之係數0.618稱為黃金分割比，其正確值為$((\sqrt{5}-1)/2)$，係$X^2 + X - 1 = 0$或$1 : X = X : (1 - X)$之根。

三. 若已有X_L, X_M, X_R三點，其函數值滿足$F(X_M) < F(X_L)$及$F(X_M) < F(X_R)$之關係，則可找到一個X之二次函數$F_2(X)$，其$F_2(X_L) = F(X_L)$，$F_2(X_R) = F(X_R)$，$F_2(X_M) = F(X_M)$。此二次函數之極小值之位置，設為X_X，必在(X_L, X_R)之區間。計算$F(X_X)$並比較$F(X_X)$

與$F(X_M)$及X_X與X_M，則可得新的X_L，X_R及X_M如下表所示：

情況		新的X_L	新的X_M	新的X_R
$F(X_X) < F(X_M)$	$X_X < X_M$	X_L	X_X	X_M
	$X_M < X_X$	X_M	X_X	X_R
$F(X_M) < F(X_X)$	$X_X < X_M$	X_X	X_M	X_R
	$X_M < X_X$	X_L	X_M	X_X

四．若X_1, X_2, X_3為最新找到的三點，利用此最新三點可求得一個X之二次函數$F_2(X)$，其$F_2(X_1) = F(X_1)$，$F_2(X_2) = F(X_2)$，$F_2(X_3) = F(X_3)$。計算此二次函數之極小值之位置，設為X_X。以此點做為最新的點，因此最新的三點即改為$X_1 = X_2$，$X_2 = X_3$，$X_3 = X_X$。

第四法因一直採用最新點之函數計算其二次函數之極小值之位置，其準確性較高，如會收斂則收斂性較好，屬二次收斂。但其缺點為不一定會收斂。因其不像前三法永遠保證極小值落在一確定之區間。而前三法雖保證一定會收斂，但其收斂性均不好，屬一次收斂。為了克服這些困難，茲提供下列方法：

五．若已有X_L, X_M, X_R三點，其函數值滿足$F(X_M) < F(X_L)$及$F(X_M) < F(X_R)$之關係。另設X_1, X_2, X_3為最新找到的三點，利用此最新三點可找到一個X之二次函數$F_2(X)$，其$F_2(X_1) = F(X_1)$，$F_2(X_2) = F(X_2)$，$F_2(X_3) = F(X_3)$。計算此二次函數之極小值之位置，設為X_X。若X_X介於(X_L, X_R)之間，則予採用；若X_X在(X_L, X_R)之區間外，則予捨去，另改由X_L, X_M, X_R三點之函數值之二次函數求其極小值之位置做為X_X，則此X_X必在(X_L, X_R)之區間。計算$F(X_X)$後同第三法比較$F(X_X)$與$F(X_M)$及X_X與X_M，可得新的X_L，X_R及X_M如第三法之表所示。而最新的三點則如第四法改為$X_1 = X_2$，$X_2 = X_3$，$X_3 = X_X$。

第五法表面上雖兼用第三第四法，但基本上係以第四法為主，並加上第三法之監視作用而確保會收斂。因其只有在可能會發散時才偶而自動改用第三法，故其收斂性為第四法之二次收斂。

[表一]之副程式係根據第五法寫成。其輸入輸出之參數詳見副程式中之註解。

[表一] 單變數函數之極值

```
************************************************************************
      FUNCTION XMin(X,F,dX,Istep,IVY,B)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(14),B(14)
      EQUIVALENCE (A(13),ST),(A(14),VY)
     *  ,(A(1),X1),(A(2),X2),(A(3),X3),(A(4),F1),(A(5),F2),(A(6),F3)
     *  ,(A(7),XL),(A(8),XM),(A(9),XR),(A(10),FL),(A(11),FM),(A(12),FR)
      DATA Pmax/1.0D30/
C**   ============================================================    **
C**   FIND  XMin                                                     **
C**   ------------------------------------------------------------   **
C**   This program combines 3 methods to get the best guess of X :   **
C**   (1) from quadratic eq. of newest_3_points (if concave_upward)  **
C**       but the max_step_size is set to 3.75*previous_step_size    **
C**   (2) from quadratic eq. of valley_set_of_3_points (if any)      **
C**       when (1)_is_NA or X_from_(1) outside the bounds_of_valley  **
C**   (3) step on the downward_side twice the previous_step_size     **
C**       when both (1)&(2)_are_NA                                   **
C**   Method(1) makes the program so fast as quadratic convergence   **
C**   Method(2) makes the program so stable as global convergence    **
C**   Method(3) makes the program so insensitive to init_step_size   **
C**   ------------------------------------------------------------   **
C**   XMin  = new guess of X for minimizing F(X)                     **
C**   X     = argument of function                                   **
C**   F     = F(X) = function value of X                             **
C**   dX    = return XMin=X+dX if no better formula to find XMin     **
C**   Istep = number of call to Function XMin                        **
C**   IVY   = 1/0 if valley set is found/not found                   **
C**   B(14) = buffer must NOT be changed between each calling Istep  **
C**   ------------------------------------------------------------   **
C**   for Istep = 1  :  (X1=X,F1=F) (ST=1) (VY=0)                    **
C**       XMin  = X1 + dX   if dX.NE.0                               **
C**       XMin  = X1 + 0.1  if dX.EQ.0                               **
C**   for Istep = 2  :  (X2=X,F2=F) (ST=2) (VY=0)                    **
C**       XMin  = X1 + (X1-X2)*2  if F1<F2                           **
C**       XMin  = X2 + (X2-X1)*2  if F2<F1                           **
C**   for Istep > 2  :  (X3=X,F3=F) (ST>2) (VY=1 if found valley)    **
C**       XMin  = Y1  from quadratic eq. of newest point (X1,X2,X3)  **
C**       XMin  = Y2  from quadratic eq. of valley set (XL,XM,XR)    **
C**                   if VY=1 and when Y1 is NA or Outside (XL,XR)   **
C**   ============================================================   **
      IF(Istep.EQ.1) THEN
        DO 10 I=1,14
        A(I)=Pmax
  10    CONTINUE
        ST=0
        VY=0
      ELSE
        DO 20 I=1,14
        A(I)=B(I)
  20    CONTINUE
      ENDIF
      ST=ST+1
C**   +----------------------------------------------------------+   **
C**   | SAVE  the newest 3 points to (X1,X2,X3)                  |   **
C**   +----------------------------------------------------------+   **
      X1=X2
      X2=X3
      X3=X
```

```
      F1=F2
      F2=F3
      F3=F
C**   +--------------------------------------------------------+    **
C**   | FIND  new points (XL,XM,XR) that bracket the root      |    **
C**   +--------------------------------------------------------+    **
      IF(X.GT.XM) THEN
        IF(F.GT.FM.OR.(F.EQ.FM.AND.FR.GT.FL)) THEN
          XR=X
          FR=F
        ELSE
          XL=XM
          XM=X
          FL=FM
          FM=F
        ENDIF
      ELSE IF(X.LT.XM) THEN
        IF(F.GT.FM.OR.(F.EQ.FM.AND.FL.GT.FR)) THEN
          XL=X
          FL=F
        ELSE
          XR=XM
          FR=FM
          XM=X
          FM=F
        ENDIF
      ENDIF
      IF(FL.LT.Pmax.AND.FR.LT.Pmax) VY=1
      Iget=0
      XX=0.0
C**   +--------------------------------------------------------+    **
C**   | FIND a new point from qradratic equation on (X1,X2,X3) |    **
C**   | if it is concave upward  d2F/dX2 = 1/ddF > 0            |    **
C**   | and make sure that ABS(XX-(X3+X1)/2) < 3*ABS(X3-X1)    |    **
C**   |                                                        |    **
C**   | X1  X2      X3    <---3.75*(X3-X2)--->    XXmax        |    **
C**   |  *---*---+---*---+---x---+---x---+---x---+-*           |    **
C**   |        *---+---+---x---+---+---x---+---+---*           |    **
C**   |        0        <---3*(X3-X1)--->        -ddX          |    **
C**   +--------------------------------------------------------+    **
      IF(ST.GT.2) THEN
        FF=(F3-F2)*(X1-X2)-(F1-F2)*(X3-X2)
        IF(FF.NE.0) THEN
          ddF=(X3-X1)*(X3-X2)*(X1-X2)/FF
          IF(ddF.GT.0) THEN
            ddX=0.5*(F3-F1)/(X3-X1)*ddF
            Rate=DABS(ddX/(X3-X1))/3.0
            IF(Rate.LE.1) Rate=1.0
            XX=0.5*(X3+X1)-ddX/Rate
            Iget=4
          ENDIF
        ENDIF
      ENDIF
C**   +--------------------------------------------------------+    **
C**   | if minimum is bracketed by (XL,XM,XR) and if no XX get  |    **
C**   | or XX goes outside (XL,XR) then use quadratic equation |    **
C**   +--------------------------------------------------------+    **
      IF(VY.NE.0) THEN
        IF(Iget.EQ.0.OR.(XX-XL)*(XX-XR).GT.0) THEN
          FF=(FR-FM)*(XL-XM)-(FL-FM)*(XR-XM)
```

```
          XX=0.5*(XR+XL)
          IF(FF.NE.0) XX=XX-0.5*(FR-FL)*(XR-XM)*(XL-XM)/FF
          Iget=3
        ENDIF
      ENDIF
C**   +-------------------------------------------------------+    **
C**   | if none XX get so far, then:                          |    **
C**   | if known at least 2 points, go the downward side      |    **
C**   | if known only one point, set XX=X+dX or XX=X+0.1      |    **
C**   +-------------------------------------------------------+    **
      IF(Iget.EQ.0) THEN
        IF(ST.GE.2) THEN
          IF(F3.GT.F2) THEN
            XX=X2
            X2=X3
            X3=XX
            XX=F2
            F2=F3
            F3=XX
          ENDIF
          XX=X3+(X3-X2)*2.0
          Iget=2
        ELSE
          IF(dX.NE.0) THEN
            XX=X+dX
          ELSE
            XX=X+0.1D0
          ENDIF
          Iget=1
        ENDIF
      ENDIF
C**   ---------------------------------------------------------    **
      DO 30 I=1,14
      B(I)=A(I)
  30  CONTINUE
      IVY=VY
      XMin=XX
      RETURN
      END
*************************************************************************
```

21.3 多變數函數之極值–直線搜尋

前節已提到求多變數函數$F(\{X\})$之極值均係反覆利用單變數函數之極值問題求解。每一單變數函數$W(T)$之極值問題均自一已知起始點$\{X_k\}$開始沿著一已知直線方向$\{D_{k+1}\}$求直線上之極值所對應之步幅T，一般稱直線搜尋(Line search)。即（其中$k=0,1,\ldots$為反覆次數）：

$$\text{極小化：}\quad W(T) \;=\; F(\{X_k\}+T*\{D_{k+1}\}) \tag{21.1}$$

於求得上列問題極點之$T = T_{k+1}$後可得新的起始點為

$$\{X_{k+1}\} = \{X_k\} + T_{k+1}\{D_{k+1}\} \tag{21.2}$$

本節之重點即為直線方向$\{D_{k+1}\}$之決定。其主要方法有下列四種：

一.最陡方向法（Steepest descent method）

本法所採用之方向為沿該方向之函數值之減少率為最大者。即方向$\{D_{k+1}\}$為

$$\{D_{k+1}\} = -\{\nabla F_k\} \tag{21.3}$$

$$\text{式中} \qquad \{\nabla F_k\} = \{\frac{\partial F(\{X\})}{\partial x_i}\}\Big|_{\{X\}=\{X_k\}} \tag{21.4}$$

$\{\nabla F_k\}$稱為梯度向量 (Gradient)，為函數變化率最大的方向，而與$F(\{X\}) = F_k$之曲面垂直。以平面地形為例：設高程為欲求最小值之函數，其變數為二個平面座標，梯度向量即垂直於等高線。因此最陡方向法為沿著起始點處最陡之方向走到最低處。再以此處做為新的起始點重覆此過程即可走到最低點。該最低點可由$\{D_{k+1}\}$之分量均很小判定之。本法之缺點為收斂很慢，屬線性收斂 。重覆次數太多。

二.共軛方向法（Conjugate gradient method）

本法所採用之方向$\{D_{k+1}\}$為

$$\{D_{k+1}\} = -\{\nabla F_k\}\,, \qquad \text{當 } k = 0 \tag{21.5}$$

$$\{D_{k+1}\} = -\{\nabla F_k\} + \{D_k\}\frac{\langle\nabla F_k\rangle\{\nabla F_k\}}{\langle\nabla F_{k-1}\rangle\{\nabla F_{k-1}\}}\,, \qquad \text{當 } k > 0 \tag{21.6}$$

本法之推導將詳述於次節。有關程式見[表二]。

三.牛頓法（Newton method）

牛頓法所採用之方向$\{D_{k+1}\}$為

$$\{D_{k+1}\} = -[H_k^{-1}]\{\nabla F_k\} \tag{21.7}$$

$$[H_k] = [\frac{\partial^2 F(\{X\})}{\partial x_i \partial x_j}]\Big|_{\{X\}=\{X_k\}} \tag{21.8}$$

$[H_k]$稱為$N \times N$之黑森矩陣（Hessian matrix）。如果原目標函數為二次函數則$[H_k]$為常數矩陣，利用牛頓法一次即可求得解且不必在直線上尋找極值，因$T_{k+1} = 1$即為函數之極值位置。

四.準牛頓法 (Quasi–Newton method)

準牛頓法基本上係將牛頓法之黑森矩陣用近似之矩陣$[A_k]$替代，而這些近似矩陣或其逆矩陣係隨試算過程之結果而修改，故有關之近似矩陣之計算式常稱為修訂式。以DFP (Davidon–Fletcher–Powell) 修訂式(21.10)為例， 所採用之搜尋方向$\{D_{k+1}\}$為

$$\{D_{k+1}\} = -[A_k^{-1}]\{\nabla F_k\} \tag{21.9}$$

$$[A_o^{-1}] = [H_o^{-1}] \quad 或 \quad [I] \tag{21.10}$$

$$[A_k^{-1}] = [A_{k-1}^{-1}] + \frac{\{\Delta x\}\langle\Delta x\rangle}{\langle\Delta x\rangle\{\Delta g\}} - \frac{[A_{k-1}^{-1}]\{\Delta g\}\langle\Delta g\rangle[A_{k-1}^{-1}]}{\langle\Delta g\rangle[A_{k-1}^{-1}]\{\Delta g\}} \tag{21.11}$$

$$\{\Delta x\} = \{X_k\} - \{X_{k-1}\} \tag{21.12}$$

$$\{\Delta g\} = \{\nabla F_k\} - \{\nabla F_{k-1}\} \tag{21.13}$$

該法尚可採用多種其他修訂式，常用修訂式之推導將詳述於另一節。有關程式見[表三]。

五.單變數法 (Univariate method)

該法一次僅改變一個變數之值，其餘變數保持固定值。即依次採用單位矩陣之第 mod (k, N) 行做為 $\{D_{k+1}\}$ 。以 $N = 5$ 為例，$\{D_1\} = \{1,0,0,0,0\}$，$\{D_2\} = \{0,1,0,0,0\}$，...，$\{D_9\} = \{0,0,0,1,0\}$，... 。該法表面上收斂速度不快，但因其不必計算梯度向量$\{\nabla F_k\}$，故整體效率並不差，而且程式容易撰寫。

[表二] 共軛方向法–FR法與PR法

```
************************************************************************
      SUBROUTINE FR(N,Xp,Gc,Gp,S,ISTEP)
C**   ================================================================   **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Xp(N),Gc(N),Gp(N),S(N)
C**   ================================================================   **
C**   Find conjugate direction Sp(N) for given Gp(N),Sc(N)               **
C**   ----------------------------------------------------------------   **
C*I   N     = Number of variables                                        **
C*I   Xp(N) = This point (Unused in this subroutine)                     **
C*I   Gc(N) = Gradient at Xc (Unused, for consistence with PR)           **
C*I   Gp(N) = Gradient at Xp                                             **
C*I   S (N) = Sc(N) : Conjugate direction at Xc (for ISTEP>0 only)       **
C*O         = Sp(N) : Conjugate direction at Xp                          **
C*I   ISTEP = 0 : Sp(I) = -Gp(I)   (Sc(N) unused)                        **
C*I         > 0 : Sp(I) = -Gp(I) + Sc(I) * (Gp.Gp)/(Gc.Gc)              **
C*I   CC    = Gc.Gc   (for ISTEP>0 only)                                 **
C*O         = Gp.Gp                                                      **
C**   ================================================================   **
```

```
C**   CALL PR(N,Xp,Gp,Gp,S,-ISTEP)
      PP=0.0
      DO 10 I=1,N
   10 PP=PP+Gp(I)*Gp(I)
      IF(ISTEP.EQ.0) THEN
        DO 20 I=1,N
   20   S(I)=-Gp(I)
      ELSE
        POC=PP/CC
        DO 40 I=1,N
   40   S(I)=-Gp(I)+S(I)*POC
      ENDIF
      CC=PP
      RETURN
      END
*************************************************************************
      SUBROUTINE PR(N,Xp,Gc,Gp,S,ISTEP)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Xp(N),Gc(N),Gp(N),S(N)
C**   ============================================================    **
C**   Find conjugate direction Sp(N) for given Gc(N),Gp(N),Sc(N)      **
C**   ------------------------------------------------------------    **
C*I   N     = Number of variables                                     **
C*I   Xp(N) = This point (Unused in this subroutine)                  **
C*I   Gc(N) = Gradient at Xc (for ISTEP>0 only)                       **
C*I   Gp(N) = Gradient at Xp                                          **
C*I   S (N) = Sc(N) : Conjugate direction at Xc (for ISTEP<>0 only)   **
C*O         = Sp(N) : Conjugate direction at Xp                       **
C*I   ISTEP = 0 : Sp(I) = -Gp(I)   (Gc(N) and Sc(N) unused)          **
C*I         > 0 : Sp(I) = -Gp(I) + Sc(I) * (Gp.Gp-Gc.Gp)/(Gc.Gc)     **
C*I         < 0 : Sp(I) = -Gp(I) + Sc(I) * Gp.Gp/(Gc.Gc)             **
C*I   CC    = Gc.Gc   (for ISTEP<>0 only)                             **
C*O         = Gp.Gp                                                   **
C**   ============================================================    **
      PP=0.0
      DO 10 I=1,N
   10 PP=PP+Gp(I)*Gp(I)
      IF(ISTEP.EQ.0) THEN
        DO 20 I=1,N
   20   S(I)=-Gp(I)
      ELSE
        CP=PP
        IF(ISTEP.GT.0) THEN
          DO 30 I=1,N
   30     CP=CP-Gc(I)*Gp(I)
        ENDIF
        POC=CP/CC
        DO 40 I=1,N
   40   S(I)=-Gp(I)+S(I)*POC
      ENDIF
      CC=PP
      RETURN
      END
*************************************************************************
```

［表三］ 準牛頓法–BFGS 法與 DFP 法

```
*************************************************************************
      SUBROUTINE BFGSLG(N,Xc,Xp,Gc,Gp,EPSMAH,ETA,ANGRAD,D,L,H,S,Y,T,VM)
```

```
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Xc(N),Xp(N),Gc(N),Gp(N),S(N),Y(N),T(N),D(N),L(N),H(1)
C**   ============================================================    **
C**   Update Hp from Hc                                               **
C**   ------------------------------------------------------------    **
C*I   N      = Number of variables                                    **
C*I   Xc(N)  = Last point                                             **
C*I   Xp(N)  = This point                                             **
C*I   Gc(N)  = Gradient at Xc                                         **
C*I   Gp(N)  = Gradient at Xp                                         **
C*I   EPSMAH = Machine epsilon                                        **
C**   ETA,ANGRAD = Dummy (Unused in this subroutine)                  **
C*I   D(N)   = Diagonal matrix for adaptive Lagrange multiplier       **
C*I   L(N)   = Location index of matrix H                             **
C*I   H(N,N) = Hessian matrix Hc                                      **
C*O          = Hessian matrix Hp                                      **
C*W   S(N)   = Xp(N)-Xc(N)  -> sqrt((1-VM)*TTS)*(Y(N)/YTS-T(N)/TTS)   **
C*W   Y(N)   = Gp(N)-Gc(N)  -> Y(N)/sqrt(YTS)                         **
C*W   T(N)   = Hc(N,N)*S(N) -> T(N)/sqrt(TTS)                         **
C*I   VM     = BFGS/(DFP+BFGS) : Proportion of update methods         **
C**            H = H_BFGS + (1-VM) * (H_DFP-H_BFGS)                   **
C**   Hp     = Hc + Y(I)*Y(J)/YTS - T(I)*T(J)/TTS                     **
C**             +(1-VM)*TTS*(Y(I)/YTS-T(I)/TTS)*(Y(J)/YTS-T(J)/TTS)   **
C**   ============================================================    **
      SS=0.0
      YY=0.0
      YS=0.0
      DO 10 I=1,N
      S(I)=Xp(I)-Xc(I)
      Y(I)=Gp(I)-Gc(I) + D(I)*S(I)
      SS=SS+S(I)*S(I)
      YY=YY+Y(I)*Y(I)
   10 YS=YS+Y(I)*S(I)
      IF(DABS(YS).LE.EPSMAH*DSQRT(SS*YY)) RETURN
C**   ------------------------------------------------------------    **
      TT=0.0
      TS=0.0
      DO 30 I=1,N
      T(I)=0.0 + D(I)*S(I)
      DO 20 J=1,I
   20 T(I)=T(I)+H(J+L(I))*S(J)
      DO 25 J=I+1,N
   25 T(I)=T(I)+H(I+L(J))*S(J)
      TT=TT+T(I)*T(I)
   30 TS=TS+T(I)*S(I)
      IF(DABS(TS).LE.EPSMAH*DSQRT(TT*SS)) RETURN
C**   ------------------------------------------------------------    **
      YS=DSQRT(YS)
      TS=DSQRT(TS)
      DO 50 I=1,N
      Y(I)=Y(I)/YS
      T(I)=T(I)/TS
   50 S(I)=(Y(I)*TS/YS-T(I))*DSQRT(1-VM)
C**   ------------------------------------------------------------    **
      DO 80 J=1,N
      DO 80 I=1,J
   80 H(I+L(J))=H(I+L(J))+Y(I)*Y(J)-T(I)*T(J)+S(I)*S(J)
      RETURN
      END
```

```
************************************************************************
      SUBROUTINE BFGS(N,Xc,Xp,Gc,Gp,EPSMAH,ETA,ANGRAD,L,B,S,Y,T,VM)
C**   ============================================================   **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Xc(N),Xp(N),Gc(N),Gp(N),S(N),Y(N),T(N),L(N),B(1)
C**   ============================================================   **
C**   Update Bp from Bc  (B is the inverse of Hessian matrix H)      **
C**   ------------------------------------------------------------   **
C*I   N      = Number of variables                                   **
C*I   Xc(N)  = Last point                                            **
C*I   Xp(N)  = This point                                            **
C*I   Gc(N)  = Gradient at Xc                                        **
C*I   Gp(N)  = Gradient at Xp                                        **
C*I   EPSMAH = Machine epsilon                                       **
C**   ETA,ANGRAD = Dummy (Unused in this subroutine)                 **
C*I   L(N)   = Location index of matrix B                            **
C*I   B(N,N) = Inverse of Hessian matrix Hc                          **
C*O          = Inverse of Hessian matrix Hp                          **
C*W   Y(N)   = Gp(N)-Gc(N)  -> sqrt(  VM * TTY)*(S(N)/STY-T(N)/TTY)  **
C*W   S(N)   = Xp(N)-Xc(N)  -> S(N)/sqrt(STY)                        **
C*W   T(N)   = Bc(N,N)*Y(N) -> T(N)/sqrt(TTY)                        **
C*I   VM     = BFGS/(DFP+BFGS) : Proportion of update methods        **
C**            B = B_DFP + VM * (B_BFGS-B_DFP)                       **
C**   Bp     = Bc + S(I)*S(J)/STY - T(I)*T(J)/TTY                    **
C**               + VM*TTY*(S(I)/STY-T(I)/TTY)*(S(J)/STY-T(J)/TTY)   **
C**   ============================================================   **
C**   DO 5 I=1,N   ! Follows can be replaced by these 3 statements   **
C** 5 T(I)=0.0                                                       **
C**   CALL BFGSLG(N,Gc,Gp,Xc,Xp,EPSMAH,ETA,ANGRAD,T,L,B,Y,S,T,1-VM)  **
      SS=0.0
      YY=0.0
      SY=0.0
      DO 10 I=1,N
      S(I)=Xp(I)-Xc(I)
      Y(I)=Gp(I)-Gc(I)
      SS=SS+S(I)*S(I)
      YY=YY+Y(I)*Y(I)
   10 SY=SY+S(I)*Y(I)
      IF(DABS(SY).LE.EPSMAH*DSQRT(SS*YY)) RETURN
C**   ------------------------------------------------------------   **
      TT=0.0
      TY=0.0
      DO 30 I=1,N
      T(I)=0.0
      DO 20 J=1,I
   20 T(I)=T(I)+B(J+L(I))*Y(J)
      DO 25 J=I+1,N
   25 T(I)=T(I)+B(I+L(J))*Y(J)
      TT=TT+T(I)*T(I)
   30 TY=TY+T(I)*Y(I)
      IF(DABS(TY).LE.EPSMAH*DSQRT(TT*YY)) RETURN
C**   ------------------------------------------------------------   **
      SY=DSQRT(SY)
      TY=DSQRT(TY)
      DO 50 I=1,N
      S(I)=S(I)/SY
      T(I)=T(I)/TY
   50 Y(I)=(S(I)*TY/SY-T(I))*DSQRT(VM)
C**   ------------------------------------------------------------   **
      DO 80 J=1,N
```

```
      DO 80 I=1,J
   80 B(I+L(J))=B(I+L(J))+S(I)*S(J)-T(I)*T(J)+Y(I)*Y(J)
      RETURN
      END
*************************************************************************
```

上述前三種方法之原理係與下列線性聯立方程式之解法有直接之關係：設$[A]$為正定對稱矩陣，則下列線性聯立方程式

$$[A]\{X\}-\{B\}=\{0\} \tag{21.14}$$

可視為下列二次函數之極小化問題

$$\text{極小化：}\quad F(\{X\}) \;=\; C-\langle B\rangle\{X\}+\frac{1}{2}\langle X\rangle[A]\{X\} \tag{21.15}$$

若將一般目標函數$F(\{X\})$以泰勒級數展開取至第二階導函數近似之，即

$$F(\{X\}) \;\doteq\; F_k+\langle\nabla F_k\rangle\{\Delta X\}+\frac{1}{2}\langle\Delta X\rangle[H_k]\{\Delta X\} \tag{21.16}$$

式中

$$F_k \;=\; F(\{X_k\}) \tag{21.17}$$

$$\{\nabla F_k\} \;=\; \{\frac{\partial F_k}{\partial x_i}\}\,,\qquad [H_k] \;=\; [\frac{\partial^2 F_k}{\partial x_i\partial x_j}] \tag{21.18}$$

$$\{\Delta X\} \;=\; \{X\}-\{X_k\} \tag{21.19}$$

式(21.16)之近似目標函數之極值位置$\{X\}$，可由相當於式(21.14)之下列線性方程式算得$\{\Delta X\}$後再由式(21.19)求得。

$$[H_k]\{\Delta X\}+\{\nabla F_k\}=\{0\} \tag{21.20}$$

直接解上列線性方程式所得之$\{\Delta X\}$做為尋找原目標函數之極值之方向$\{D_{k+1}\}$即為牛頓法。

如果原目標函數為二次函數，則$[H_k]$為常數矩陣，利用牛頓法一次即可求得解且不必在直線上尋找極點，因$T_{k+1}=1$即為函數之極點位置；而共軛方向法 一般須做N次之線性搜尋；而最陡方向法則須做N次以上之線性搜尋。如果原目標函數為高次函數且起始點離極點甚遠時，則利用牛頓法不一定會指向極點，甚至會指向相反之方向；此時最陡方向則可能為最有效之搜尋方向。一般而言，在離極點較遠處，最陡方向法能確保求得更佳之新點，稱為具有整體收斂性 (Global convergence)；反之，在極點之近處，牛頓法所得近似點之誤差能以二次方式遞減而快速求得準確之極點，稱為具有局部二次收斂性 (Local quadratic convergence)。以上各法均無法兼具二種收斂性。以下介紹之曲線搜尋法應較能兼具之。

21.4 多變數函數之極值－曲線搜尋

如將上節之直線搜尋改為曲線搜尋，而此曲線在起始點與最陡方向相切，同時又能通過牛頓點或準牛頓點，則能兼具二種收斂性。曲線如以參數表示，則沿曲線搜尋仍為單一變數（即曲線參數）函數之極值問題，並未因直線改為曲線而增加複雜度。具上述特性之曲線可能有無窮之多，以下介紹一種以比吉爾曲線 (Bezier curve) 所定之搜尋曲線，對於二次目標函數而言，沿該曲線之函數值具有單調遞減之特性，此特性表示在牛頓點之前沒有會擋住往牛頓點繼續搜尋之局部升高點。

利用 $(\{X_O\},\{X_C\},\{X_N\})$ 三點由式 (21.23) 所定義之曲線，如符合式 (21.24) 至式 (21.28) 之條件，則曲線會在該三點所圍成之三角形平面內。其中 $\{X_O\}$ 為起始點，即式 (21.15) 之 $\{X_k\}$ 點；$\{X_N\}$ 為牛頓點或準牛頓點，即式 (21.19) 與式 (21.20) 之解； $\{X_C\}$ 為柯其點 (Cauchy point)，為式 (21.16) 之二次近似函數在最陡方向 $-\{\nabla F_k\}$ 之極小點，令其 $\{\Delta X\}=-\lambda_C\nabla F_k$，即可求得極值之 λ_C 及柯其點如式 (21.22) 及式 (21.21)。

$$\{X_C\}=\{X_O\}-\lambda_C\{\nabla F_O\} \tag{21.21}$$

$$\lambda_C=\frac{\langle\nabla F_O\rangle\{\nabla F_O\}}{\langle\nabla F_O\rangle[H_O]\{\nabla F_O\}} \tag{21.22}$$

$$\begin{aligned}\{X(t)\}&=\{X_O\}+\alpha(t)(\{X_N\}-\{X_O\})+\beta(t)(\{X_C\}-\{X_O\})\\&=\alpha(t)\{X_N\}+\beta(t)\{X_C\})+\gamma(t)\{X_O\}\end{aligned} \tag{21.23}$$

$$1>t\geq 0 \tag{21.24}$$

$$\alpha(t)\geq 0 \tag{21.25}$$

$$\beta(t)\geq 0 \tag{21.26}$$

$$\gamma(t)>0 \tag{21.27}$$

$$\alpha(t)+\beta(t)+\gamma(t)=1 \tag{21.28}$$

如該曲線之係數又符合式 (21.29) 之條件則可証明式 (21.30) 之關係成立，即二次目標函數 $F(X)$ 沿曲線由 $\{X_O\}$ 至 $\{X_N\}$ 為單調遞減。

$$\alpha'(t)\geq 0\,,\qquad -\gamma'(t)>0 \tag{21.29}$$

$$\langle\nabla F(\{X(t)\})\rangle\{X'(t)\}<0 \tag{21.30}$$

由 $i=0,1,\ldots,n$ 等 $n+1$ 個控制點 $\{P_i\}$ 所定義之 n 階比吉爾曲線為：

（其中$B_{i,n}(t)$為t之n階多項式稱為Mernstein polynomial）

$$\{P(t)\} = \sum_{i=0}^{n} B_{i,n}(t)\,\{P_i\}\,, \quad 0 \le t \le 1 \tag{21.31}$$

$$B_{i,n}(t) = C_i^n\, t^i\,(1-t)^{n-i} \tag{21.32}$$

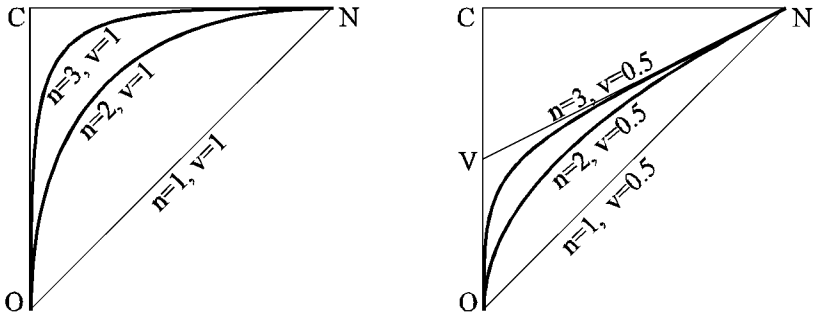
$$C_i^n = \frac{n!}{i!\,(n-i)!} \tag{21.33}$$

以$\{X_O\},\{X_N\}$為控制點之比吉爾曲線為牛頓方向之直線；以$\{X_O\},\{X_C\},\{X_N\}$為控制點之比吉爾曲線為二次曲線；若重複使用中間點，即得式(21.34)之n階比吉爾曲線。該式所建立之比吉爾曲線會通過$\{X_O\}$與$\{X_N\}$二點，且分別與$\overline{OC}$或$\overline{CN}$相切。式中之$\{X_C\}$亦可用$\{X_O\}$至$\{X_C\}$間之點$\{X_V\}$取代，即$\{X_V\} = \{X_O\} + v\,(\{X_C\} - \{X_O\})$，式中$1 \ge v > 0$，而曲線仍然會與$\{\nabla F_O\}$相切，但通過$N$點之曲線則改與$\overline{VN}$相切。比較式(21.34)與式(21.23)，可得式(21.35)之係數，且於$1 > t \ge 0$時式(21.36)會成立，故二次目標函數沿曲線為單調遞減。有關程式見[表四]。

$$\{X(t)\} = t^n\,\{X_N\} + (1 - t^n - (1-t)^n)\,\{X_C\} + (1-t)^n\,\{X_O\} \tag{21.34}$$

$$\alpha(t) = t^n\,, \qquad \beta(t) = 1 - t^n - (1-t)^n\,, \qquad \gamma(t) = (1-t)^n \tag{21.35}$$

$$\alpha'(t) = n\,t^{n-1} \ge 0\,, \qquad -\gamma'(t) = n\,(1-t)^{n-1} > 0 \tag{21.36}$$



圖一 比吉爾曲線

[表四(a)] 線性或曲線搜尋副程式

```
************************************************************************
      SUBROUTINE LINSCH
     *(NX,NS,NE,N,XOLD,XNEW,SN,SC,PFAC,PPEN,C,BEZTYP,GFN)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XOLD(N),XNEW(N),SN(N),SC(N),PFAC(NX)
      DIMENSION PPEN(6),C(1),BEZTYP(2),BUFFER(14)
```

```
      DATA NSTP,ERRX,ERRD,DMAG,DX/15,1D-6,1D-4,0D0,0.25D0/
C**   ============================================================    **
C**   Find minimier XNEW(N) of GFN(XNEW(N))                           **
C**   Search along the Bezier curve of (Xo,Xc,Xn) or (Xo,Xv,Xn)        **
C**   ------------------------------------------------------------    **
C*I   NX        = Number of original variables                        **
C*I   NS        = Number of constraints Gj(Xp(NX)) <= 0               **
C*I   NE        = Number of constraints Hi(Xp(NX))  = 0               **
C*I   XOLD(N)   = Xold = Xp(NX)[,S(NS),Ls(NS),Le(NE)]                 **
C*O   XNEW(N)   = Xnew = Xp(NX)[,S(NS),Ls(NS),Le(NE)]                 **
C*I   SN(N)     = Xn - Xold = Newton direction                        **
C*I   SC(N)     = Xc - Xold = Cauchy direction                        **
C*I   PFAC(NX)  = Scalar factor of Xp(NX)                             **
C*I   BEZTYP(2) = Nb,v for type of Bezier curve (See sub. SPT)        **
C*S   SPT       = Sub. to get Xnew from XX                            **
C*S   GFN       = Sub. to get general obj. function of Xnew(N)        **
C**             = GFNPEN,GFNLAG,GFMLAG                                **
C**   ::For Penalty function                                          **
C*I   N         = NX                                                  **
C*I   XOLD(N)   = Xp(NX)                                              **
C*I   PPEN(6)   = Xp(NX),R(3),T,C1,C2                                 **
C*W   C(2*NC)   = Gj(Xp(NX)),Hi(Xp(NX)),dYsj/dGj,dYei/dHi             **
C**   ::For Lagrange multiplier                                       **
C*I   N         = NX+NS+NS+NE                                         **
C*I   XOLD(N)   = Xp(NX),S(NS),Ls(NS),Le(NE)                          **
C*I   PFAC(NX+1  :NX+N  ) = Newton point Xn(N)     : for GFMLAG       **
C*I   PFAC(NX+N+1:NX+N+N) = Modified diagonal D(N) : for GFMLAG       **
C*W   C(NC)     = Gj(Xp(NX)),Hi(Xp(NX))                               **
C**   ============================================================    **
      ERRD=1D-4
      XX=1.0D30
      FX=1.0D30
      XN=0.0                                        ! Initialize
      XM=XX                                         ! Minimizer
      FM=FX                                         ! Minimum F
      DO 10 ISTEP=1,NSTP
        XO=XX
        FO=FX
        XX=XN
        CALL SPT(N,XOLD,SN,SC,XNEW,XX,BEZTYP)     ! get XNEW(N) from XX
        CALL GFN(NX,NS,NE,XNEW,PFAC,PPEN,C,FX)    ! FX=F(XX) from XNEW(N)
        IF(FX.LT.FM) THEN
          FM=FX
          XM=XX
        ENDIF
        XN=XMIN(XX,FX,DX,ISTEP,IVY,BUFFER)
        IF(ISTEP.EQ.2.AND.DMAG.EQ.0.0) ERRD=ERRD*ABS((FX-FO)/(XX-XO))
        WRITE(*,'('' BUF(14)''/(1X,1P6E13.5))') BUFFER
        IF(ISTEP.GE.4.OR.IVY.NE.0) THEN
          IF(ABS(XX-XO).LE.ERRX) GO TO 20
          IF(ABS(FX-FO).LE.ERRD*ABS(XX-XO)) GO TO 20
        ENDIF
   10 CONTINUE
   20 CALL SPT(N,XOLD,SN,SC,XNEW,XM,BEZTYP)
      RETURN
      END
*************************************************************************
```

[表四(b)] 求比吉爾曲線上之點

```
*************************************************************************
```

```
      SUBROUTINE SPT(N,XOLD,SN,SC,XNEW,T,BEZTYP)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XOLD(N),SN(N),SC(N),XNEW(N),BEZTYP(2)
C**   ============================================================    **
C**   Find a point on Bezier curve for given T and (Xo,Xc,Xn)         **
C**   ------------------------------------------------------------    **
C*I   N       = Number of variables                                   **
C*I   XOLD(N) = Xo      = Initial point                               **
C*I   SN(N)   = Xn - Xo = Newton direction                            **
C*I   SC(N)   = Xc - Xo = Cauchy direction                            **
C*O   XNEW(N) = Xo + t^Nb*(Xn-Xo) + (1-t^Nb-(1-t)^Nb) * (Xv-Xo)       **
C**           = Xo + t^Nb*(Xn-Xo) + (1-t^Nb-(1-t)^Nb)*v*(Xc-Xo)       **
C**             Xv = v*Xc + (1-v)*Xo                                  **
C*I   T = t   = Parameter of Bezier curve                            **
C*I   Nb      = DABS(BEZTYP(1)) = Order of Bezier curve              **
C**           = 0 : For straight line OC (SN,BEZTYP(2:2) Unuse)       **
C**           = 1 : For straight line ON (SC,BEZTYP(2:2) Unuse)       **
C**           > 1 : For Bezier curve of order Nb, BUT IF              **
C**                 BEZTYP(1) < 0 : Use line ON for T>=1 or T<=0      **
C*I   v       = DABS(BEZTYP(2)) <= 1                                  **
C**   ============================================================    **
      NB=DABS(BEZTYP(1))
      IF(NB.EQ.0) THEN
        DO  5 I=1,N
    5   XNEW(I)=XOLD(I)+T*SC(I)
      ELSE IF(NB.EQ.1.OR.BEZTYP(2).EQ.0
     *  .OR.(BEZTYP(1).LT.0.AND.(T.GE.1.OR.T.LE.0))) THEN
        DO 10 I=1,N
   10   XNEW(I)=XOLD(I)+T*SN(I)
      ELSE
        TN=T**NB
        TC=(1-TN-(1-T)**NB)*DABS(BEZTYP(2))
        DO 20 I=1,N
   20   XNEW(I)=XOLD(I)+TN*SN(I)+TC*SC(I)
      ENDIF
      RETURN
      END
*************************************************************************
```

21.5 共軛方向之推導

考慮下列線性聯立方程式，其中$[A]$為正定對稱矩陣：

$$[A]\{X\} = \{B\} \tag{21.37}$$

$\{u\}$與$\{v\}$二向量若滿足下列關係：

$$\langle u\rangle[A]\{v\} = 0 \tag{21.38}$$

則稱$\{u\}$與$\{v\}$對矩陣$[A]$互為共軛，或簡稱$\{u\}$與$\{v\}$互為共軛。對任一$n \times n$之正定對稱矩陣，至少會有一組n個互為共軛之向量：例如其n個特

徵向量即互為共軛，因特徵向量亦符合式(21.38)。且互為共軛之向量亦為互相獨立之向量：蓋若p個互為共軛向量為相依，即$\{d_p\}=\sum_{i=1}^{p-1}x_i\{d_i\}$，則$\langle d_p\rangle[A]\{d_p\}=\langle d_p\rangle[A](\sum_{i=1}^{p-1}x_i\{d_i\})=\sum_{i=1}^{p-1}x_i\langle d_p\rangle[A]\{d_i\}=0$，即得矛盾之結果$\{d_p\}=\{0\}$。因最多只能有$n$個互相獨立之向量，故一組互為共軛之向量最多為$n$個。既然一組$n$個互為共軛之向量為互相獨立，故任一向量$\{w\}$均可寫成$n$個互為共軛之向量$\{d_i\}$之線性組合，即$\{w\}=\sum_{i=1}^{n}x_i\{d_i\}$，其中$x_i=\langle d_i\rangle[A]\{w\}/\langle d_i\rangle[A]\{d_i\}$，可由該式兩邊前乘$\langle d_i\rangle[A]$並引用式(21.38)之共軛關係而得。同理式(21.37)之解$\{X\}=[A^{-1}]\{B\}$，或$\{X\}-\{X_o\}=[A^{-1}]\{B\}-\{X_o\}$亦可分別表為

$$\{X\} = \sum_{i=1}^{n}x_i\{d_i\}\,,\quad x_i=\frac{\langle d_i\rangle\{B\}}{\langle d_i\rangle[A]\{d_i\}} \tag{21.39}$$

$$\{X\}-\{X_o\} = \sum_{i=1}^{n}x_i\{d_i\}\,,\quad x_i=\frac{\langle d_i\rangle(\{B\}-[A]\{X_o\})}{\langle d_i\rangle[A]\{d_i\}} \tag{21.40}$$

根據式(21.40)，令

$$\{X_i\}=\{X_{i-1}\}+x_i\{d_i\}\,,\quad i=1,2,\ldots,n \tag{21.41}$$

則$\{X\}=\{X_n\}$。亦即式(21.37)之解可視為由任一假設之點$\{X_o\}$依次沿$\{d_1\},\{d_2\},\ldots,\{d_n\}$之方向移動至$\{X_1\},\{X_2\},\ldots,\{X_n\}$。而最後之點$\{X_n\}$即為式(21.37)之解$\{X\}$。

令

$$\{g_i\}=\{\nabla F(\{X_i\})\} \tag{21.42}$$

則因（由式(21.14)及式(21.41)）

$$\{\nabla F(\{X_i\})\} = [A](\{X_o\}+\sum_{s=1}^{i}x_s\{d_s\})-\{B\} \tag{21.43}$$

故當$p\le i$時(第二等式之第一項引用式(21.40))

$$\begin{aligned}\langle d_p\rangle\{g_i\} &= \langle d_p\rangle([A](\{X_o\}+\sum_{s=1}^{i}x_s\{d_s\})-\{B\})\\ &= \langle d_p\rangle([A]\{X_o\}-\{B\})+x_p\langle d_p\rangle[A]\{d_p\}\\ &= 0\end{aligned} \tag{21.44}$$

上式中$\langle d_i\rangle\{g_i\}=0$，意即向量$\{d_i\}$與向量$\{\nabla F(\{X_i\})\}$正交，亦即表示$\{X_i\}$點之目標函數$F(\{X_i\})$為自$\{X_{i-1}\}$沿$\{d_i\}$方向上之最小者。因此，

$\{X_i\}$之位置可由$\{X_{i-1}\}$開始，沿向量$\{d_i\}$之方向尋找目標函數$F(\{X\})$之極點而求得x_i，亦即對於二次函數之目標函數，沿$\{d_i\}$找極點所得之x_i，與利用式(21.40)所得者相同。用此求法於其他非二次函數之目標函數，可僅用一階微分之$\{\nabla F_{i-1}\}$，而不必用式(21.40)所需之二階微分$[H_{i-1}]$。

另注意：對於二次目標函數，式(21.40)之x_i大小與$\{d_i\}$之順序無關。亦即所有x_i均可自$\{X_o\}$開始沿各個$\{d_i\}$方向尋找各該方向上之最小目標函數F之位置$\{X_o\}+x_i\{d_i\}$而求得相同之x_i。

至目前為止尚未介紹如何計算一組共軛向量$\{d_i\}$。以下介紹者為最常用於解極小化問題之方法，稱FR（Fletcher–Reeves）法。

若第$i+1$個共軛向量欲由下式求出：

$$\{d_{i+1}\} \;=\; -\{g_i\}+x_i\{d_i\}+\sum_{s=1}^{i-1} y_s\{d_s\} \tag{21.45}$$

則由上式（令$i=p$）及式(21.44)可証得，當$p<i$時：

$$\langle g_p\rangle\{g_i\} \;=\; (-\{d_{p+1}\}+x_p\{d_p\}+\sum_{s=1}^{p-1} y_s\{d_s\})\{g_i\}=0 \tag{21.46}$$

利用共軛之條件，即

$$\langle d_s\rangle[A]\{d_{i+1}\}=0\,,\quad s\le i \tag{21.47}$$

由式(21.45)前乘$\langle d_s\rangle[A]$或$\langle d_i\rangle[A]$，可得式中之y_s及x_i如下：

$$y_s \;=\; \frac{\langle d_s\rangle[A]\{g_i\}}{\langle d_s\rangle[A]\{d_s\}}\,,\quad s<i \tag{21.48}$$

$$x_i \;=\; \frac{\langle d_i\rangle[A]\{g_i\}}{\langle d_i\rangle[A]\{d_i\}} \tag{21.49}$$

但由式(21.41)至式(21.45)及式(21.46)可得：

$$\begin{aligned}\langle d_s\rangle[A]\{g_i\} &= \frac{1}{x_s}(\langle X_s\rangle-\langle X_{s-1}\rangle)[A]\{g_i\}\\ &= \frac{1}{x_s}(\langle g_s\rangle-\langle g_{s-1}\rangle)\{g_i\}=0\,,\quad s<i \end{aligned}\tag{21.50}$$

$$\begin{aligned}\langle d_i\rangle[A]\{g_i\} &= \frac{1}{x_i}(\langle g_i\rangle-\langle g_{i-1}\rangle)\{g_i\}\\ &= \frac{1}{x_i}\langle g_i\rangle\{g_i\}\end{aligned}\tag{21.51}$$

$$\begin{aligned}\langle d_i\rangle[A]\{d_i\} &= \frac{1}{x_i}(\langle g_i\rangle-\langle g_{i-1}\rangle)(-\{g_{i-1}\}+x_{i-1}\{d_{i-1}\}+\sum_{s=1}^{i-2}y_s\{d_s\})\\ &= \frac{1}{x_i}\langle g_{i-1}\rangle\{g_{i-1}\} \qquad (21.52)\end{aligned}$$

因此得：

$$y_s = 0\,, \qquad s<i \qquad (21.53)$$

$$x_i = \frac{\langle g_i\rangle\{g_i\}}{\langle g_{i-1}\rangle\{g_{i-1}\}} \qquad (21.54)$$

注意式(21.54)不必用式(21.49)中之$[A]$，即$[H_{i-1}]$。而式(21.45)簡化為：

$$\{d_{i+1}\} = -\{g_i\}+\frac{\langle g_i\rangle\{g_i\}}{\langle g_{i-1}\rangle\{g_{i-1}\}}\{d_i\} \qquad (21.55)$$

上式即為前節共軛方向法所用之式(21.5)。式(21.55)中之$\langle g_i\rangle\{g_i\}$如改為$(\langle g_i\rangle-\langle g_{i-1}\rangle)\{g_i\}$即為$PR$(Polak–Ribiere)法。該法對二次目標函數之結果相同， 因$\langle g_{i-1}\rangle\{g_i\}=0$，但對一般目標函數則收斂性有時會好些。

21.6 有限制條件之最佳化問題

前面所介紹之方法除可直接用於解無限制條件之極值問題外，下列之有限制條件之最佳化問題，亦可經轉換為一系列之無限制條件之極值問題後，再利用前述方法求系列中各問題之解，而此系列問題之解會逐漸趨近有限制條件之最佳化問題之解。

極小化： $F(X)$ (21.56)

受制於： $g_j(X) \le 0\,, \qquad j=1,\ldots,N_s$ (21.57)

$h_i(X) = 0\,, \qquad i=1,\ldots,N_e$ (21.58)

多數方法對等式限制與不等式限制均可處理，故一般均分別考慮之。但若等式限制較方便處理時，可增設一變數S_j將不等式改為等式$g_j(X)+S_j^2=0$；反之若不等式限制較方便處理時，可將等式改為二個不等式$h_i(X)\le 0$及$-h_i(X)\le 0$。

轉化為無限制條件之最佳化問題之方法可分為二類：一為拉格蘭治乘數(Lagrange multiplier)法；一為懲罰函數(Penalty function)法。茲分別介紹於下列各節。本節及以後各節中大部分向量$\{X\}$, $\{S\}$, $\{L_s\}$, $\{L_e\}$均省略$\{\}$以簡化符號。

21.7 拉格蘭治乘數法

利用拉格蘭治乘數於每一個等式限制式可得如下之無限制條件之目標函數：

$$U(X,S,L_s,L_e)=F(X)+\sum_{j=1}^{N_s}L_{sj}(g_j(X)+S_j^2)+\sum_{i=1}^{N_e}L_{ei}h_i(X) \tag{21.59}$$

該函數有穩定值 (Stational value) 之必要條件為：

$$\{\nabla U\}=\left\{\begin{array}{c}\{\frac{\partial U}{\partial X}\}\\ \{\frac{\partial U}{\partial S}\}\\ \{\frac{\partial U}{\partial L_s}\}\\ \{\frac{\partial U}{\partial L_e}\}\end{array}\right\}=\{0\} \tag{21.60}$$

或

$$\begin{aligned}\{\frac{\partial U}{\partial X}\} &= \{\frac{\partial F}{\partial X}\}+\sum L_{sj}\{\frac{\partial g_j}{\partial X}\}+\sum L_{ei}\{\frac{\partial h_i}{\partial X}\}\\ &= \{\nabla F\}+\sum L_{sj}\{\nabla g_j\}+\sum L_{ei}\{\nabla h_i\}=0 &(21.61)\\ \{\frac{\partial U}{\partial S}\} &= 2\{L_{sj}S_j\}=0\,, \qquad j=1,\ldots,N_s &(21.62)\\ \{\frac{\partial U}{\partial L_s}\} &= \{g_j(X)+S_j^2\}=0\,, \qquad j=1,\ldots,N_s &(21.63)\\ \{\frac{\partial U}{\partial L_e}\} &= \{h_i(X)\}=0\,, \qquad i=1,\ldots,N_e &(21.64)\end{aligned}$$

或表示為下列之 (Kuhn–Tucker) 最佳值條件：

$$\begin{aligned}\{\nabla F\} &= -\sum_{j=1}^{N_s}L_{sj}\{\nabla g_j\}-\sum_{i=1}^{N_e}L_{ei}\{\nabla h_i\} &(21.65)\\ (\ L_{sj} &= 0 \quad 若 \quad g_j=-S_j^2<0 \quad 為無效限制條件)\\ (\ L_{sj} &\geq 0 \quad 若 \quad g_j=-S_j^2=0 \quad 為有效限制條件)\\ L_{sj} &\geq 0\,, \qquad j=1,\ldots,N_s &(21.66)\\ L_{sj}g_j(X) &= 0\,, \qquad j=1,\ldots,N_s &(21.67)\\ g_j(X) &\leq 0\,, \qquad j=1,\ldots,N_s &(21.68)\\ h_i(X) &= 0\,, \qquad i=1,\ldots,N_e &(21.69)\end{aligned}$$

注意條件式(21.65)、式(21.68)、式(21.69)相當於條件式(21.61)、式(21.63)、式(21.64)。由條件式(21.63)可知條件式(21.67)亦相當於條件式(21.62)。注意式(21.67)表示$L_{sj}=0$或$g_j(X)=0$，因此式(21.65)中之$\sum$相當於不含無效限制條件（即$g_j(X)<0$，$L_{sj}=0$）之部分。至於條件式(21.66)則為使原來之最佳化問題之目標函數$F(X)$為“極小值”之條件，茲說明如下：

令$\{f\}$為最佳解處之任一合理方向，則合理方向之條件為：

$$\langle f\rangle\{\nabla g_j\} \leq 0 \qquad (S_j=0) \tag{21.70}$$

$$\langle f\rangle\{\nabla h_i\} = 0 \tag{21.71}$$

目標函數為“極小值”之條件為：

$$\langle f\rangle\{\nabla F\} = -\sum_{j=1}^{N_s} L_{sj}\langle f\rangle\{\nabla g_j\} - \sum_{i=1}^{N_e} L_{ei}\langle f\rangle\{\nabla h_i\} \geq 0 \tag{21.72}$$

若$L_{sj}\geq 0$，則滿足條件式(21.70)(21.71)之任一合理方向均能使條件式(21.72)滿足。但若其中某一個$L_{sk}<0$，設$\{\nabla g\}$與$\{\nabla h\}$為線性獨立，則可選一合理方向$\{f\}$使

$$\langle f\rangle\{\nabla g_k\} < 0 \qquad (S_k=0) \tag{21.73}$$

$$\langle f\rangle\{\nabla g_j\} = 0 \qquad (S_j=0,\ j\neq k) \tag{21.74}$$

$$\langle f\rangle\{\nabla h_i\} = 0 \tag{21.75}$$

則$\langle f\rangle\{\nabla F\} = -L_{sk}\langle f\rangle\{\nabla g_k\} < 0$，即不滿足極小值之條件式(21.72)，故必須$L_{sk}\geq 0$。

根據(Kuhn-Tucker)條件式(21.65)至式(21.69)，注意其中已不含變數S，故可重新定義拉格蘭治函數為：

$$U(X,L_s,L_e) = F(X) + \sum_{j=1}^{N_s} L_{sj}g_j(X) + \sum_{i=1}^{N_e} L_{ei}h_i(X) \tag{21.76}$$

但須式中之

$$L_{sj}\geq 0\,, \qquad j=1,\ldots,N_s \tag{21.77}$$

則求式(21.76)拉格蘭治函數之穩定解即可得式(21.56)之極小化問題之解。

利用拉格蘭治乘數法之最大困難為最佳解之$U(X^*,L_s^*,L_e^*)$，雖對變數(X)為極小值，但對滿足$L_{sj}\geq 0$之變數(L_s,L_e)則為極大值，即

$$U(X^*,L_s,L_e) \leq U(X^*,L_s^*,L_e^*) \leq U(X,L_s^*,L_e^*) \tag{21.78}$$

上式第一個關係証明如下：因對於滿足(Kuhn-Tucker)條件之最佳解：

$$h_i(X^*) = 0\ , \quad L^*_{sj} g_j(X^*) = 0\ , \quad g_j(X^*) \le 0 \tag{21.79}$$

但$L_{sj} \ge 0$，故

$$\begin{aligned} \sum L_{sj} g_j(X^*) &\le 0 && (21.80)\\ U(X^*, L^*_s, L^*_e) &= F(X^*) + \sum L^*_{sj} g_j(X^*) + \sum L^*_{ei} h_i(X^*) \\ &= F(X^*) && (21.81)\\ U(X^*, L_s, L_e) &= F(X^*) + \sum L_{sj} g_j(X^*) + \sum L_{ei} h_i(X^*) \\ &= F(X^*) + \sum L_{sj} g_j(X^*) && (21.82) \end{aligned}$$

因此得$U(X^*, L_s, L_e) \le U(X^*, L^*_s, L^*_e)$。

反之若對所有$L_{sj} \ge 0$之(L_s, L_e)均能滿足下列不等式：

$$U(X^*, L_s, L_e) \le U(X^*, L^*_s, L^*_e) \tag{21.83}$$

其中

$$\begin{aligned} U(X^*, L^*_s, L^*_e) &= F(X^*) + \sum L^*_{sj} g_j(X^*) + \sum L^*_{ei} h_i(X^*) && (21.84)\\ U(X^*, L_s, L_e) &= F(X^*) + \sum L_{sj} g_j(X^*) + \sum L_{ei} h_i(X^*) && (21.85) \end{aligned}$$

則因L_{ei}為任意：若令$L_{ei} = +\infty$則須$h_i(X^*) \le 0$，若令$L_{ei} = -\infty$則須$h_i(X^*) \ge 0$，故須$h_i(X^*) = 0$，方可使式(21.83)成立。因$L_{sj} \ge 0$：若令$L_{sj} = +\infty$則須$g_j(X^*) \le 0$，若令$L_{sj} = 0$則須$L^*_{sj} g_j(X^*) \ge 0$，方可使式(21.83)成立。但$L^*_{sj} \ge 0$且$g_j(X^*) \le 0$，故須$L^*_{sj} g_j(X^*) = 0$。

式(21.78)第二個關係除可由式(21.72)得知外，亦可由式(21.87)看出：如黑森矩陣$[H(F(X))]$為半正定且$L_{sj} \ge 0$，則黑森矩陣$[H(U(X, S, L_s, L_e))]$中對應於(X, S)之部分黑森矩陣為半正定矩陣。因此固定(L_s, L_e)時，對應於變數(X, S)之函數之極值為極小值。（式中$\lceil\cdot\rfloor$為對角矩陣）

$$[H(U(X, L_s, L_e))] = \begin{bmatrix} [H(F(X))] & [\{\nabla g_j\}] & [\{\nabla h_i\}] \\ [\langle \nabla g_j \rangle] & [0] & [0] \\ [\langle \nabla h_i \rangle] & [0] & [0] \end{bmatrix} \tag{21.86}$$

$$[H(U(X,S,L_s,L_e))]=\begin{bmatrix}[H(F(X))] & [0] & [\{\nabla g_j\}] & [\{\nabla h_i\}]\\ [0] & \lceil 2L_{sj}\rfloor & \lceil 2S_j\rfloor & [0]\\ [\langle\nabla g_j\rangle] & \lceil 2S_j\rfloor & [0] & [0]\\ [\langle\nabla h_i\rangle] & [0] & [0] & [0]\end{bmatrix} \tag{21.87}$$

既然由拉格蘭治乘數所得之目標函數之解為滿足式(21.78)之鞍點，故前面所述之線性搜尋法均不能適用於以(X,L_s,L_e)為變數之$U(X,L_s,L_e)$。但可用下列極大化極小值 (Max.min) 問題求解：

$$U_*(L_s^*,L_e^*)=U(X_*(L_s^*,L_e^*),L_s^*,L_e^*)=$$

以(L_s,L_e)極大化：$U_*(L_s,L_e)=U(X_*(L_s,L_e),L_s,L_e)$。而

$$U_*(L_s,L_e)=U(X_*(L_s,L_e),L_s,L_e)=$$

固定(L_s,L_e)以(X)極小化：$U(X,L_s,L_e)=F(X)+\sum L_{sj}g_j(X)+\sum L_{ei}h_i(X)$

因上列問題為下列極小化極大值 (Min.max) 問題之基本問題之對偶問題：

$$U^*(X_*)=U(X_*,L_s^*(X_*),L_e^*(X_*))=$$

以(X)極小化：$U^*(X)=U(X,L_s^*(X),L_e^*(X))$。而

$$U^*(X)=U(X,L_s^*(X),L_e^*(X))=$$

固定(X)以(L_s,L_e)極大化：$U(X,L_s,L_e)=F(X)+\sum L_{sj}g_j(X)+\sum L_{ei}h_i(X)$

而上列之固定(X)以求(L_s,L_e)之極大化問題之解為：

$$\begin{aligned}U^*(X) &= F(X) \quad 若\quad g_j(X)\le 0 \quad 且\quad h_i(X)=0\\ &= +\infty \quad 若\quad g_j(X)>0 \quad 則\quad L_{sj}=+\infty\\ &\qquad\quad 若\quad h_i(X)>0 \quad 則\quad L_{ei}=+\infty\\ &\qquad\quad 若\quad h_i(X)<0 \quad 則\quad L_{ei}=-\infty\end{aligned}$$

但在求$U^*(X)$之極小值時，該值等於$+\infty$之部分顯然可以略去，因此以(X)極小化$U^*(X)$即等於極小化$F(X)$受制於$g_j(X)\le 0$及$h_i(X)=0$。此極小化問題即為原來之有限制條件之最佳化問題，因此原來之最佳化問題（即被稱為基本問題）可用其對偶問題之極大化極小值問題求解。

21.8 機動調整拉格蘭治函數法

一般線性搜尋法只能搜尋函數之極值，無法求函數之鞍點。本節介紹之法可以將拉格蘭治函數做機動之調整，使調整後之函數之牛頓點與原函數之牛頓點相同，但調整函數在牛頓點變為極小值， 因此可用前二節之線性搜尋法尋找調整函數之極點，而求得原函數之鞍點。

為敘述方便，原函數寫成$F(\{X\})$，令變數$\{X\}$含拉格蘭治乘數，即包含所有變數。調整函數寫成$\bar{F}(\{X\})$，定義如下：

$$\bar{F}(\{X\}) = F(\{X\}) + \frac{1}{2}(\langle X\rangle - \langle X_N\rangle)\,[D]\,(\{X\} - \{X_N\}) \tag{21.88}$$

式中$[D]$為對角矩陣，該調整函數在$\{X_O\}$處之梯度向量及黑森矩陣為：

$$\{\nabla \bar{F}_O\} = \{\nabla F_O\} + [D]\,(\{X_O\} - \{X_N\}) \tag{21.89}$$

$$[\bar{H}_O] = [H_O] + [D] \tag{21.90}$$

其牛頓點為：

$$\{X_N\} = \{X_O\} - ([H_O] + [D])^{-1}\,(\{\nabla F_O\} + [D]\,(\{X_O\} - \{X_N\})) \tag{21.91}$$

上式二邊前乘$([H_O] + [D])$，化簡，再前乘$[H_O^{-1}]$可得：

$$\{X_N\} = \{X_O\} - [H_O^{-1}]\,\{\nabla F_O\} \tag{21.92}$$

上式即為原函數之牛頓點之計算式，可知調整函數並未改變牛頓點之位置。如$[D]$夠大，則其黑森矩陣即由非正定變為正定矩陣，牛頓點亦由原函數之鞍點變成調整函數之極小點， 因此可用線性搜尋法找尋調整函數之極小值而求得原函數之鞍點。因調整函數隨起始點之牛頓點而改變，故稱為機動調整。 與$BFGS$修訂式之配合程式見[表五]。

[表五] 計算梯度向量與黑森矩陣之副程式

```
************************************************************************
      SUBROUTINE FDGRAD(NX,NS,NE,N,X,PFAC,PPEN,C,Fp,ETA,G)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),PFAC(N),PPEN(6),C(1),G(N)
C**   ============================================================    **
C**   Compute Gradient G(N) by difference & partial analysis          **
C**   ------------------------------------------------------------    **
C*I   NX       = Number of original variables                         **
C*I   NS       = Number of constraints Gj(Xp(NX)) <= 0                **
```

```
C*I   NE       = Number of constraints Hi(Xp(NX))  = 0                **
C*I   N        = NX            : for Penalty function                 **
C*I            = NX+NS+NS+NE   : for Lagrange multiplier              **
C*I   X(N)     = Xp(NX)[,S(NS),Ls(NS),Le(NE)]                         **
C*I   PFAC(NX) = Scalar factor of Xp(NX)                              **
C*I   PPEN(6)  = R(1),R(2),R(3): Penalty parameters                   **
C*I              T,C1,C2       : Hyperbolic penalty function coef.   **
C*W   C(2*NC)  = Gj(Xp(NX)),Hi(Xp(NX)),dYsj/dGj,dYei/dHi : Penalty    **
C*W   C(NS+NE) = Gj(Xp(NX)),Hi(Xp(NX))                    : Lagrange   **
C*O   Fp       = F(X(N))                                              **
C*I   ETA      = Machine epsilon                                      **
C*O   G(N)     = Gradient at X(N)                                     **
C*S   GFNPN0,1 = Sub. to compute diff. Penalty obj. func. F(X(N))     **
C*S   GFNLAG   = Sub. to compute Lagrange objective func. F(X(N))     **
C**   ================================================================**
      NC=NS+NE
      ETAH=DSQRT(ETA)
      IF(N.EQ.NX) CALL GFNPN0(NX,NS,NE,X,PFAC,PPEN,C,Fp,C(NC+1),C(NC+1))
      IF(N.NE.NX) CALL GFNLAG(NX,NS,NE,X,PFAC,PPEN,G(NX+1),Fp)
      DO 10 J=1,NX
      Xp=X(J)
      X(J)=Xp+ETAH*DMAX1(DABS(Xp),1.0D0)
      IF(N.EQ.NX) CALL GFNPN1(NX,NS,NE,X,PFAC,PPEN,C,FJ,C(NC+1),C(NC+1))
      IF(N.NE.NX) CALL GFNLAG(NX,NS,NE,X,PFAC,PPEN,C,FJ)
      G(J)=(FJ-Fp)/(X(J)-Xp)
   10 X(J)=Xp
C**   +----------------------------------------------------------+    **
C**   | For Lagrange multiplier method                           |    **
C**   +----------------------------------------------------------+    **
      DO 20 J=NX+NS+1,N-NE
      G(J-NS)=2*X(J-NS)*X(J)
   20 G(J)=G(J)+X(J-NS)**2
C     DO 25 I=NX+NS+NS+1,N
C  25 G(I)=G(I)
      RETURN
      END
*************************************************************************
      SUBROUTINE FDHESS(NX,NS,NE,N,X,PFAC,PPEN,C,F,DX,ETA,L,H)
C**   ================================================================**
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),PFAC(NX),PPEN(6),C(1),F(NX),DX(NX),L(N),H(1)
C**   ================================================================**
C**   Compute Hessian matrix H(N,N) by difference & partial analysis  **
C**   ----------------------------------------------------------------**
C*I   NX       = Number of original variables                         **
C*I   NS       = Number of constraints Gj(Xp(NX)) <= 0                **
C*I   NE       = Number of constraints Hi(Xp(NX))  = 0                **
C*I   N        = NX            : for Penalty function                 **
C*I            = NX+NS+NS+NE   : for Lagrange multiplier              **
C*I   X(N)     = Xp(NX)[,S(NS),Ls(NS),Le(NE)]                         **
C*I   PFAC(NX) = Scalar factor of Xp(NX)                              **
C*I   PPEN(6)  = R(1),R(2),R(3): Penalty parameters                   **
C*I              T,C1,C2       : Hyperbolic penalty function coef.   **
C*W   C(NC*(NX+4)) = C(NC),HH(NC,NX),O(NC),D(NC),DD(NC) : Penalty     **
C*W   C(NC*2)      = C(NC),O(NC)                        : Lagrange    **
C*W   F(NX)    = F(X(N)+DX(I)*ei)                                     **
C*W   DX(NX)   = Step size                                           **
C*I   ETA      = Machine epsilon                                      **
C*I   L(N)     = Location index of matrix H(N,N)                      **
C*O   H(N+L(N))= Hessian matrix H(N,N) at X(N)                        **
```

```
C*S   GFNPN0,1 = Sub. to compute diff. Penalty obj. func. F(X(N))     **
C*S   GFNLAG   = Sub. to compute Lagrange objective func. F(X(N))     **
C**   ============================================================    **
      NC=NS+NE
      NO=NC+NC*NX
      ND=NO+NC+1
      NDD=NO+NC+NC
      ETAH=ETA**0.333333
      DO 10 I=1,N*(N+1)/2
   10 H(I)=0.0
      IF(N.EQ.NX) THEN
        CALL GFNPN0(NX,NS,NE,X,PFAC,PPEN,C(NO+1),Fp,C(ND),C(NDD+1))
      ELSE
        CALL GFNLAG(NX,NS,NE,X,PFAC,PPEN,C(NC+1),Fp)
      ENDIF
      DO 30 I=1,NX
      XI=X(I)
      X(I)=XI+ETAH*DMAX1(DABS(XI),1.0D0)
      DX(I)=X(I)-XI
      IF(N.EQ.NX) THEN
        CALL GFNPN1(NX,NS,NE,X,PFAC,PPEN,C,F(I),C(ND),C(NDD+1))
        DO 20 J=1,NS+NE
   20   C(NC*I+J)=(C(J)-C(NO+J))/DX(I)
      ELSE
        CALL GFNLAG(NX,NS,NE,X,PFAC,PPEN,C,F(I))
        DO 25 J=1,NS+NE
   25   H(I+L(NX+NS+J))=(C(J)-C(NC+J))/DX(I)
      ENDIF
   30 X(I)=XI
C**   +-------------------------------------------------------+    **
C**   | Use forward difference for off-diagonal H(I,J)        |    **
C**   +-------------------------------------------------------+    **
      DO 40 J=1,NX
      XJ=X(J)
      X(J)=X(J)+DX(J)
      DO 35 I=1,J-1
      XI=X(I)
      X(I)=X(I)+DX(I)
      IF(N.EQ.NX) CALL GFNPN1(NX,NS,NE,X,PFAC,PPEN,C,FIJ,C(ND),C(NDD+1))
      IF(N.NE.NX) CALL GFNLAG(NX,NS,NE,X,PFAC,PPEN,C,FIJ)
      H(I+L(J))=((FIJ-F(J))-(F(I)-Fp))/(DX(I)*DX(J))
   35 X(I)=XI
C**   +-------------------------------------------------------+    **
C**   | Use central difference for diagonal H(J,J)            |    **
C**   +-------------------------------------------------------+    **
      X(J)=XJ-DX(J)
      IF(N.EQ.NX) CALL GFNPN1(NX,NS,NE,X,PFAC,PPEN,C,FIJ,C(ND),C(NDD+1))
      IF(N.NE.NX) CALL GFNLAG(NX,NS,NE,X,PFAC,PPEN,C,FIJ)
      H(J+L(J))=((F(J)-Fp)-(FIJ-Fp))/(DX(J)*DX(J))
   40 X(J)=XJ
C**   +-------------------------------------------------------+    **
C**   | For Lagrange multiplier method                        |    **
C**   +-------------------------------------------------------+    **
      IF(N.NE.NX) THEN
      DO 50 J=NX+1,N
      H(J+L(J+NS))=2*X(J)
   50 H(J+L(J))=2*X(J+NS)
      ELSE
C**   +-------------------------------------------------------+    **
C**   | For Penalty function method                           |    **
```

```
C**   +-------------------------------------------------------+    **
      DO 80 J=1,NX
      DO 80 I=1,J
      GIJ=0.0
      HIJ=0.0
      DO 60 K=1,NS
   60 GIJ=GIJ+C(NDD+K)*C(NC*I+K)*C(NC*J+K)
      DO 70 K=NS+1,NS+NE
   70 HIJ=HIJ+C(NDD+K)*C(NC*I+K)*C(NC*J+K)
   80 H(I+L(J))=H(I+L(J))+PPEN(2)*GIJ+PPEN(1)*HIJ
      ENDIF
      RETURN
      END
*************************************************************************
      SUBROUTINE GFNLAG(NX,NS,NE,X,PFAC,PPEN,C,W)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(1),PFAC(1),C(1)
C**   ============================================================    **
C**   Compute general objective function by Lagrange multiplier       **
C**   ------------------------------------------------------------    **
C*I   N        = NX+NS+NS+NE                                          **
C*I   X(N)     = Xp(NX),S(NS),Ls(NS),Le(NE)                           **
C**   PFAC(NX) = Scalar factor of Xp(NX)    (unused)                  **
C**   PPEN(6)  = R(3),T,C1,C2               (unused)                  **
C*W   C(NS+NE) = Gj(Xp(NX)),Hi(Xp(NX))                               **
C*O   W        = F + Ls(j)*(G(j) + S(j)**2)+ Le(i)*H(i)              **
C*S   FFF      = Sub. to compute F(Xp(NX)),Gj(Xp(NX)),Hi(Xp(NX))      **
C**   ============================================================    **
      CALL FFF(NX,NS,NE,X,F,C)
      W=0.0
      DO 20 J=1,NS
   20 W=W+X(NX+NS+J)*(C(J)+X(NX+J)**2)
      DO 30 I=NS+1,NS+NE
   30 W=W+X(NX+NS+I)*C(I)
      W=F+W
      RETURN
      END
*************************************************************************
      SUBROUTINE GFMLAG(NX,NS,NE,X,PFAC,PPEN,C,W)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(1),PFAC(1),PPEN(6),C(1)
C**   ============================================================    **
C**   Compute Adaptive modified Lagrange function                     **
C**   ------------------------------------------------------------    **
C*I   N        = NX+NS+NS+NE                                          **
C*I   X(N)     = Xp(NX),S(NS),Ls(NS),Le(NE)                           **
C*I   PFAC(NX) = Scalar factor of Xp(NX)                              **
C*I   PFAC(NX+1  :NX+N  ) = Newton point Xn(N)                        **
C*I   PFAC(NX+N+1:NX+N+N) = Modified diagonal D(N)                    **
C**   PPEN(6)  = R(3),T,C1,C2               (unused)                  **
C*W   C(NS+NE) = Gj(Xp(NX)),Hi(Xp(NX))                               **
C*O   W        = FL + (X(i)-Xn(i))*D(i)*(X(i)-Xn(i)) / 2             **
C**   FL       = F(Xp(NX)) + Ls(j)*(G(j) + S(j)**2)+ Le(i)*H(i)      **
C*S   GFNLAG   = Sub. to compute Lagrange objective func. FL(X(N))    **
C**   ============================================================    **
      N=NX+NS+NS+NE
      CALL GFNLAG(NX,NS,NE,X,PFAC,PPEN,C,FL)
      W=0.0
```

```
      DO 20 I=1,N
      DX=X(I)-PFAC(I+NX)
   20 W=W+DX*PFAC(I+NX+N)*DX
      W=FL+0.5*W
      RETURN
      END
*************************************************************************
      SUBROUTINE GFNPEN(NX,NS,NE,X,PFAC,PPEN,C,W)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(NX),PFAC(NX),PPEN(6),C(NS+NE)
C**   DATA T,C1,C2/0.1D-3,1.0D0,0.1D0/
C**   ============================================================    **
C**   Compute general objective function by Penalty function          **
C**   ------------------------------------------------------------    **
C*I   N        = NX                                                   **
C*I   X(NX)    = Xp(NX)                                               **
C*I   PFAC(NX) = Scalar factor of Xp(NX)    (unused)                  **
C*I   PPEN(6)  = R(3),T,C1,C2                                         **
C*W   C(NS+NE) = Gj(Xp(NX)),Hi(Xp(NX))                                **
C*O   W        = R(3)*F + R(2)*Ys(Gj;T,C1,C2) + R(1)*Ye(Hi;T,C1,C2)   **
C*S   FFF      = Sub. to compute F(Xp(NX)),Gj(Xp(NX)),Hi(Xp(NX))      **
C**   ============================================================    **
      CALL FFF(NX,NS,NE,X,F,C)
      PG=0.0
      PH=0.0
      DO 20 J=1,NS
   20 PG=PG+PF(C(J),PFAC(J),PPEN(4),PPEN(5),PPEN(6),-1)
      DO 30 I=NS+1,NS+NE
   30 PH=PH+PF(C(I),PFAC(I),PPEN(4),PPEN(5),PPEN(6), 0)
      W=PPEN(3)*F+PPEN(2)*PG+PPEN(1)*PH
      RETURN
      END
*************************************************************************
      SUBROUTINE GFNPN0(NX,NS,NE,X,PFAC,PPEN,C,W,D,DD)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(NX),PFAC(NX),PPEN(6),C(NS+NE),D(NS+NE),DD(NS+NE)
C**   DATA T,C1,C2/0.1D-3,1.0D0,0.1D0/
C**   ============================================================    **
C**   Compute differential Penalty function for the gradient          **
C**   ------------------------------------------------------------    **
C*I   N         = NX                                                  **
C*I   X(NX)     = Xp(NX)                                              **
C*I   PFAC(NX)  = Scalar factor of Xp(NX)                             **
C*I   PPEN(6)   = R(3),T,C1,C2                                        **
C*W   C(NS+NE)  = Gj(Xp(NX)),Hi(Xp(NX))                               **
C*O   W         = R(3)*F + R(2)*DYj*Gj+ R(1)*DYi*Hi                   **
C*O   D(NS+NE)  = dYsj/dGj   or dYei/dHi                              **
C*O   DD(NS+NE) = d2Ysj/dGj2 or d2Yei/dHi2                            **
C*S   FFF       = Sub. to compute F(Xp(NX)),Gj(Xp(NX)),Hi(Xp(NX))     **
C**   ============================================================    **
      CALL FFF(NX,NS,NE,X,F,C)
      PG=0.0
      PH=0.0
      DO 20 J=1,NS
      CALL PD(C(J),PFAC(J),PPEN(4),PPEN(5),PPEN(6),-1,Y,D(J),DD(J))
   20 PG=PG+D(J)*C(J)
      DO 30 I=NS+1,NS+NE
      CALL PD(C(I),PFAC(I),PPEN(4),PPEN(5),PPEN(6), 0,Y,D(I),DD(I))
```

```
   30 PH=PH+D(I)*C(I)
      W=PPEN(3)*F+PPEN(2)*PG+PPEN(1)*PH
      RETURN
      END
*************************************************************************
      SUBROUTINE GFNPN1(NX,NS,NE,X,PFAC,PPEN,C,W,D,DD)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(NX),PFAC(NX),PPEN(6),C(NS+NE),D(NS+NE),DD(NS+NE)
C**   DATA T,C1,C2/0.1D-3,1.0D0,0.1D0/
C**   ============================================================    **
C**   Compute differential Penalty function for the gradient          **
C**   ------------------------------------------------------------    **
C*I   N         = NX                                                  **
C*I   X(NX)     = Xp(NX)                                              **
C*I   PFAC(NX)  = Scalar factor of Xp(NX)                             **
C*I   PPEN(6)   = R(3),T,C1,C2                                        **
C*W   C(NS+NE)  = Gj(Xp(NX)),Hi(Xp(NX))                               **
C*O   W         = R(3)*F + R(2)*DYj*Gj + R(1)*DYi*Hi                  **
C1I   D(NS+NE)  = dYsj/dGj   or dYei/dHi   : input  for GRDPN1        **
C0O   D(NS+NE)  = dYsj/dGj   or dYei/dHi   : output for GRDPN0        **
C1I   DD(NS+NE) = d2Ysj/dGj2 or d2Yei/dHi2 : input  for GRDPN1        **
C0O   DD(NS+NE) = d2Ysj/dGj2 or d2Yei/dHi2 : output for GRDPN0        **
C*S   FFF       = Sub. to compute F(Xp(NX)),Gj(Xp(NX)),Hi(Xp(NX))     **
C**   ============================================================    **
      CALL FFF(NX,NS,NE,X,F,C)
      PG=0.0
      PH=0.0
      DO 20 J=1,NS
C0    CALL PD(C(J),PFAC(J),PPEN(4),PPEN(5),PPEN(6),-1,Y,D(J),DD(J))
   20 PG=PG+D(J)*C(J)                    ! D(1:NS) are given
      DO 30 I=NS+1,NS+NE
C0    CALL PD(C(I),PFAC(I),PPEN(4),PPEN(5),PPEN(6), 0,Y,D(I),DD(I))
   30 PH=PH+D(I)*C(I)                    ! D(NS+1:NS+NE) are given
      W=PPEN(3)*F+PPEN(2)*PG+PPEN(1)*PH
      RETURN
      END
*************************************************************************
```

21.9 懲罰函數法

懲罰函數法係將有限制條件之最佳化問題轉化為如下之無限制條件之最佳化問題：

$$\text{最小化：}\ W(X;R_s,R_e)=F(X)+R_s\sum_{j=1}^{N_s}Y_s(g_j(X))+R_e\sum_{i=1}^{N_e}Y_e(h_i(X)) \quad (21.93)$$

式中R_s及R_e為懲罰參數。這些懲罰參數於最佳化時為固定之值，即最小化函數為原來設計變數X之函數，此函數稱為虛擬目標函數。除倒數懲罰函數外，多數懲罰函數之懲罰效果如太小則結果之精度不高，但若太大則使原目標函數不顯著，而無法接近最佳點。故一般常須以漸近方式調

整懲罰參數，使懲罰效果由小漸大，因此而需求解多次不同之虛擬目標函數之最佳化問題。通常等式限制與不等式限制所用之懲罰函數並不相同，且不同懲罰函數所用之懲罰參數之調整方式亦不盡相同，茲將常用之懲罰函數介紹如下：

1. 倒數懲罰函數

不等式： $Y_s(g(X)) = -1/g(X),\ \ g(X) < 0$

等式 ： 不適用

$R_s = R$，修正方式為$R_m = R_{m-1}/Rate$（$Rate$為一修正變率，大小為$10 - 100$）。

此函數僅適用於不等限制式，且於不可行區（$g(X) \geq 0$）無定義，故初始點及搜尋過程之中間點之X均不可超過可行區，運算上較難處理，為其缺點；但因所有解均在可行區內，可隨時停止運算而得合理解為其優點，此懲罰函數亦因此稱為內部懲罰函數。此懲罰函數之懲罰參數之修正方式亦較特別，R值愈小時即懲罰效果愈小時，其解才會愈接近正確解。

2. 平方懲罰函數

不等式： $Y_s(g(X)) = [max(0, g(X))]^2$

等式 ： $Y_e(h(X)) = [h(X)]^2$

$R_s = R_e = R$，修正方式為$R_m = R_{m-1} * Rate$（$Rate = 10 - 100$）。

此懲罰函數之等式者屬C^2函數，可適用於二階之方法（即用到目標函數之二階微分者，如牛頓法）；不等式者屬C^1函數，僅可適用於一階之方法。此函數之主要缺點為懲罰函數在最佳解附近較為平緩，懲罰效果不大，須用較大之R值才會得到較正確之解。

3. 正確懲罰函數

不等式： $Y_s(g(X)) = |max(0, g(X))|$

等式 ： $Y_e(h(X)) = |h(X)|$

$R_s = R_e = R$選擇一適當值後不必再改變。

此函數之主要缺點為不可微分，僅適用於效率較差之零階之方法（本書未介紹）。

4. 雙曲線懲罰函數

不等式： $Y_s(g(X)) = Y = \frac{1}{2}(\sqrt{g(X)^2 + T^2} + g(X))$

等式 ： $Y_e(h(X)) = Y = \sqrt{h(X)^2 + T^2}$

$R_s = R_e = R$，修正方式為$R_m = R_{m-1} * 2$。

T之修正方式為$T_m = T_{m-1}/(Rate * R)$（$Rate = 10 - 100$）。

注意此懲罰函數多含一參數T稱為形狀參數，此參數控制雙曲線之形狀，當$T = 0$此雙曲線懲罰函數變成正確懲罰函數。不過只要$T > 0$此雙曲線懲罰函數即為連續可微分至任何階之函數。不像正確懲罰函數為不可微分函數。故此函數可稱為近乎正確可微分懲罰函數。

5. 理想懲罰函數（Y為雙曲線懲罰函數）

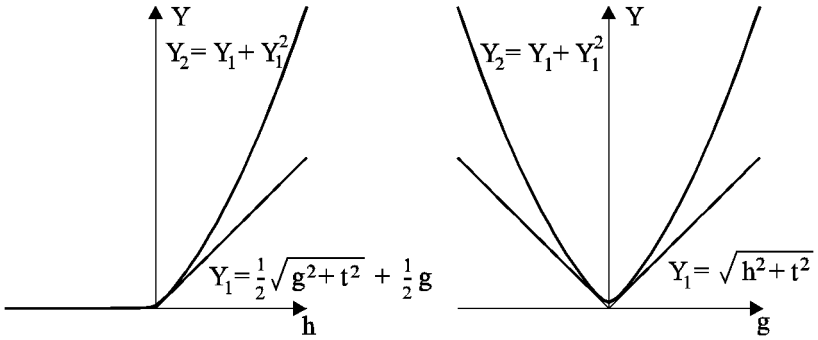
不等式：第一型 $Y_s(g(X)) = c_1 * Y + c_2 * Y^2$

第二型 $Y_s(g(X)) = e^{c_1 * Y + c_2 * Y^2} - 1$

等式　：第一型 $Y_e(h(X)) = c_1 * Y + c_2 * Y^2$

第二型 $Y_e(h(X)) = e^{c_1 * Y + c_2 * Y^2} - 1$

此懲罰函數以雙曲線懲罰函數為基本函數。第二型適用於指數型之目標函數，一般目標函數採用第一型即可。第一型之$c_2 = 0$時即為雙曲線懲罰函數。Y項對於接近限值之限制條件有較正確之懲罰效果，使所得結果較為正確而更符合限制條件；而Y^2項對於偏離限值較遠之限制條件有較合理之懲罰作用，使限制條件離限值愈遠者調近限值之輻度愈大。因此該懲罰函數能兼顧遠近之限制條件。事實上，除上列二型懲罰函數外，亦可包含Y之高次項，或採用Y之其他單調函數 (Monotonic function) 做為懲罰函數。有關程式見[表六]。



圖二　理想懲罰函數

[表六]　理想懲罰函數

```
************************************************************************
      SUBROUTINE PD(G,P,T,C1,C2,NCTP,F,DF,DDF)
C**   ============================================================    **
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
C*I   G     = value of constraint                                     **
C*I   P     = positive scale factor of G                              **
C*I   T     = shape parameter of the hyperbola                        **
C*I   C1,C2 = coefficients for modified penalty function              **
C*I   NCTP  = type of constraint                                      **
C**         > 0 for constraint P*G>=0  :   S=-P ==> X = S*G <= 0      **
C**         < 0 for constraint P*G<=0  :   S=+P ==> X = S*G <= 0      **
C**         = 0 for constraint P*G =0  :   S=+P ==> X = S*G  = 0      **
C*O   F     = C1*Y + C2*Y*Y      = Penalty function for T>0           **
C**         = exp(C1*Y+C2*Y*Y)-1 = Penalty function for T<0           **
C*O   DF    = (dF/dY)*DY                     :  DY  =  dY/dG          **
C*O   DDF   = (dF/dY)*DDY + (d2F/dY2)*DY*DY  :  DDY = d2Y/dG2         **
C**   ============================================================    **
      S=P
      IF(NCTP.GT.0) S=-S
      X=S*G
C**   +----------------------------------------------------------+    **
C**   | Penalty function for X=0 constraint                      |    **
C**   | from  Y**2 - X**2 = T**2                 \        /      |    **
C**   | Y   = sqrt(X*X+T*T)                        \    /        |    **
C**   | DY  = S*X/sqrt(X*X+T*T) = S*X/Y              \/          |    **
C**   | DDY = (S*T)**2/sqrt(X*X+T*T)**3                          |    **
C**   +----------------------------------------------------------+    **
      Y=DSQRT(X*X+T*T)
      DY=S*X/Y
      DDY=(S*T/Y)**2/Y
C**   +----------------------------------------------------------+    **
C**   | Penalty function for X<=0 constraint                     |    **
C**   | from  (2Y)**2 - 2*X*(2Y) = T**2                   /      |    **
C**   | 2Y  = sqrt(X*X+T*T)+X                            /       |    **
C**   | 2DY = S*(sqrt(X*X+T*T)+X)/sqrt(X*X+T*T)   ______/        |    **
C**   | 2DDY= (S*T)**2/sqrt(X*X+T*T)**3                          |    **
C**   +----------------------------------------------------------+    **
      IF(NCTP.NE.0) THEN
      Y2=Y+DABS(X)
      IF(X.LT.0.0) Y2=T*T/Y2
      DDY=0.5*DDY
      DY=0.5*S*Y2/Y
      Y=0.5*Y2
      ENDIF
C**   +----------------------------------------------------------+    **
C**   | Modified penalty function                                |    **
C**   +----------------------------------------------------------+    **
      F=C1*Y+C2*Y*Y
      D1=C1+2*C2*Y
      D2=2*C2
      IF(T.LT.0) THEN
        F=DEXP(DMIN1(F,64.0D0))
        D2=(D1*D1+D2)*F
        D1=D1*F
        F=F-1
      ENDIF
C**   +----------------------------------------------------------+    **
C**   | For gradient, only DF is required, Loc.DF=Loc.DDF is OK  |    **
C**   +----------------------------------------------------------+    **
      DDF=D1*DDY+D2*DY*DY
      DF=D1*DY
      RETURN
```

```
      END
************************************************************************
      SUBROUTINE PE(G,P,T,C1,C2,NCTP,F)
CD    SUBROUTINE PD(G,P,T,C1,C2,NCTP,F,DF,DDF)
C**   ================================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ================================================================    **
C*I   G     = value of constraint                                         **
C*I   P     = positive scale factor of G                                  **
C*I   T     = shape parameter of the hyperbola                            **
C*I   C1,C2 = coefficients for modified penalty function                  **
C*I   NCTP  = type of constraint                                          **
C**         > 0 for constraint P*G>=0  :   S=-P ==> X = S*G <= 0          **
C**         < 0 for constraint P*G<=0  :   S=+P ==> X = S*G <= 0          **
C**         = 0 for constraint P*G =0  :   S=+P ==> X = S*G  = 0          **
C*O   F     = C1*Y + C2*Y*Y      = Penalty function for T>0               **
C**         = exp(C1*Y+C2*Y*Y)-1 = Penalty function for T<0               **
C*O   DF    = (dF/dY)*DY                      :  DY  =  dY/dG             **
C*O   DDF   = (dF/dY)*DDY + (d2F/dY2)*DY*DY  :  DDY = d2Y/dG2             **
C**   ================================================================    **
      S=P
      IF(NCTP.GT.0) S=-S
      X=S*G
C**   +----------------------------------------------------------+    **
C**   | Penalty function for X=0 constraint                      |    **
C**   | from  Y**2 - X**2 = T**2                  \        /     |    **
C**   | Y   = sqrt(X*X+T*T)                         \    /       |    **
C**   | DY  = S*X/sqrt(X*X+T*T) = S*X/Y               \/         |    **
C**   | DDY = (S*T)**2/sqrt(X*X+T*T)**3                          |    **
C**   +----------------------------------------------------------+    **
      Y=DSQRT(X*X+T*T)
CD    DY=S*X/Y
CD    DDY=(S*T/Y)**2/Y
C**   +----------------------------------------------------------+    **
C**   | Penalty function for X<=0 constraint                     |    **
C**   | from  (2Y)**2 - 2*X*(2Y) = T**2                    /     |    **
C**   | 2Y  = sqrt(X*X+T*T)+X                             /      |    **
C**   | 2DY = S*(sqrt(X*X+T*T)+X)/sqrt(X*X+T*T)   ______/        |    **
C**   | 2DDY= (S*T)**2/sqrt(X*X+T*T)**3                          |    **
C**   +----------------------------------------------------------+    **
      IF(NCTP.NE.0) THEN
      Y2=Y+DABS(X)
      IF(X.LT.0.0) Y2=T*T/Y2
CD    DDY=0.5*DDY
CD    DY=0.5*S*Y2/Y
      Y=0.5*Y2
      ENDIF
C**   +----------------------------------------------------------+    **
C**   | Modified penalty function                                |    **
C**   +----------------------------------------------------------+    **
      F=C1*Y+C2*Y*Y
CD    D1=C1+2*C2*Y
CD    D2=2*C2
      IF(T.LT.0) THEN
        F=DEXP(DMIN1(F,64.0D0))
CD      D2=(D1*D1+D2)*F
CD      D1=D1*F
        F=F-1
      ENDIF
C**   +----------------------------------------------------------+    **
```

```
C**   | For gradient, only DF is required, Loc.DF=Loc.DDF is OK  |    **
C**   +----------------------------------------------------------+    **
CD    DDF=D1*DDY+D2*DY*DY
CD    DF=D1*DY
      RETURN
      END
*************************************************************************
      FUNCTION PF(G,P,T,C1,C2,NCTP)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
C**   ============================================================    **
      CALL PE(G,P,T,C1,C2,NCTP,PF)
      RETURN
      END
*************************************************************************
```

21.10 比例問題

為了使數個限制式在虛擬目標函數內之懲罰效果相當，可將式(21.93)之虛擬目標函數改為：

$$W(X;R_s,R_e)=\frac{F(X)}{|F(X_r)|}+R_s\sum_{j=1}^{N_s}Y_s(\frac{g_j(X)}{|g_j(X_r)|})+R_e\sum_{i=1}^{N_e}Y_e(\frac{h_i(X)}{|h_i(X_r)|}) \quad (21.94)$$

或

$$W(X;R_s,R_e)=F(X)+R_s\sum_{j=1}^{N_s}Y_s(\frac{|F(X_r)|}{|g_j(X_r)|}\,g_j(X))+R_e\sum_{i=1}^{N_e}Y_e(\frac{|F(X_r)|}{|h_i(X_r)|}\,h_i(X)) \quad (21.95)$$

除雙曲線懲罰函數外，以上二式之效果相同；但對於雙曲線懲罰函數，因其多含一個形狀參數T，如欲使以上二式之效果相同，則式(21.95)中之T必須等於式(21.94)中之T乘以$|F(X_r)|$。但式(21.94)中之T值與限制式之梯度大小之相對誤差約成正比，不受$|F(X_r)|$值之影響，故使用式(21.94)較為理想。式(21.94)之比例參數相當於使調整後之目標函數及限制式之變化率（即梯度大小）相等且為固定值。實用上比例參數不須很準，故不必每次隨$\{X\}$改變，只須在系列中前二次最佳化問題之初始點各計算一次即可，因第二次最佳化問題之初始點（即第一次最佳化問題之解）已相當接近有限制條件之最佳化問題之解，故以該點做為$\{X_r\}$（即參考點）算得之比例參數應頗具代表性。

21.11 系列線性規劃法

本章最後介紹一種與前述做法較少關聯之有限制條件之最佳化問題之另一解法。該法係將非線性之目標函數與限制條件以線性式近似，再以線性規劃法求解。但其解為近似之線性式之解，須再以新解重新線性化，而須做一系列之線性規劃問題。由於新解若離線性化之起始點太遠則線性式之誤差太大而可能無法收斂至極點，因此另須限制變數之變化範圍，如式(21.99)，其中σ_l與σ_u為常數，可設約0.1。將式(21.56)至式(21.58)各函數對起始點$\{X_o\}$做泰勒級數展開，並捨去二次及以上之高次項，可得下列之線性規劃問題：

$$\text{最小化：} \quad \langle \nabla F \rangle \{\Delta X\} \tag{21.96}$$

$$\text{受制於：} \quad \langle \nabla g_j \rangle \{\Delta X\} \le -g_{jo}\,, \qquad j = 1, \ldots, N_s \tag{21.97}$$

$$\langle \nabla h_i \rangle \{\Delta X\} = -h_{io}\,, \qquad i = 1, \ldots, N_e \tag{21.98}$$

$$\{\sigma_u X_o\} \ge \{\Delta X\} \ge -\{\sigma_l X_o\} \tag{21.99}$$

上式中梯度向量之計算，亦可分別針對變數的正向增量$\{\Delta X^+\}$與負向減量$\{\Delta X^-\}$做差分而得：如$\{\nabla^+ F\} = \{\dfrac{F(\{X_o\} + \sigma_u X_{oi}\{e_i\}) - F(\{X_o\})}{\sigma_u X_{oi}}\}$與$\{\nabla^- F\} = \{\dfrac{F(\{X_o\} - \sigma_l X_{oi}\{e_i\}) - F(\{X_o\})}{\sigma_l X_{oi}}\}$，其中$\{e_i\}$為$n \times n$之單位矩陣之第$i$行。然後改用下列線性規劃問題求解。該法雖然變數多了一倍，梯度的計算需增加一半的函數值計算量，但因線性化之誤差較小，每次線性規劃之變數增量可以較大。有關程式詳[表七]。

$$\text{最小化：} \quad \langle \nabla^+ F \rangle \{\Delta X^+\} + \langle \nabla^- F \rangle \{\Delta X^-\} \tag{21.100}$$

$$\text{受制於：} \quad \langle \nabla^+ g_j \rangle \{\Delta X^+\} + \langle \nabla^- g_j \rangle \{\Delta X^-\} \le -g_{jo} \tag{21.101}$$

$$\langle \nabla^+ h_i \rangle \{\Delta X^+\} + \langle \nabla^- h_i \rangle \{\Delta X^-\} = -h_{io} \tag{21.102}$$

$$\{\sigma_u X_o\} \ge \{\Delta X^+\} \ge \{0\} \tag{21.103}$$

$$\{\sigma_l X_o\} \ge \{\Delta X^-\} \ge \{0\} \tag{21.104}$$

[表七] 系列線性規劃法

```
**************************************************************************
      PROGRAM SLPDSM
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(50000)
      DATA NDIM/50001/
```

```
C**   ============================================================    **
C**   Sequential Linear Programming by Double Secant Method           **
C**   ============================================================    **
      READ(*,'(6I4,F8.0)') NX,NS,NE,IS,IC,MAXS,EPS
C**   ------------------------------------------------------------    **
      NGW =NS+NE+1
      NGWA=NS+NE+2      ! Constrained eqs.+ objective fun.+ big.M
      NXXB=NX+NX+1      ! Positive increment + Negative inc.+ RHS
      NXXS=NX+NX+NS     ! Positive inc.+ Negative inc.+ Slack var.
C**   ------------------------------------------------------------    **
      N1=1
      N2=N1+NX                  ! XX(NX)
      N3=N2+NX                  ! DX(NX)
      N4=N3+NX                  ! DXM(NX)
      N5=N4+NX                  ! XXU(NX)
      N6=N5+NX                  ! XXL(NX)
      N7=N6+NS                  ! GGU(NS)
      N8=N7+NGW                 ! GWO(NGW)
      M1=N8+NGW                 ! GWN(NGW)
      M2=M1+NGWA*NXXB           ! A(NIA,NJA) = A(NS+NE+2,NX+NX+NS)
      M3=M2+NGWA                ! LI(NIA)
      M4=M3+NXXB                ! LJ(NJA)
      M5=M4+NXXS                ! D(NJA)
      K1=M5+NXXS                ! X(NJA)
      K2=K1+NGWA*NXXB           ! B(NIA,NJA)
      IF(IS.GE.0) K2=K1         ! B is not reqd for IS.GE.0
      K3=K2+NGWA                ! MI(NIA)
      K4=K3+NXXB                ! MJ(NJA)
      IF(IS.EQ.0) K4=K1         ! MI,MJ are not reqd for IS.EQ.0
      IF(K4.GE.NDIM) STOP 1111
C**   +------------------------------------------------------------+    **
C**   | Input XX(NX),DXM(NX),XXU(NX),XXL(NX),GGU(NS)               |    **
C**   +------------------------------------------------------------+    **
      CALL INPXDX(A(N1),A(N3),A(N4),A(N5),A(N6),NX,NS)
C**   +------------------------------------------------------------+    **
C**   | Call SLP to perform Sequential Linear Programming          |    **
C**   +------------------------------------------------------------+    **
      CALL SLP(A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8)
     *        ,NX,NS,NE,NGW,EPS,MAXS
     *        ,A(M1),A(M2),A(M3),A(M4),A(M5),NGWA,NXXB,NXXS
     *        ,A(K1),A(K2),A(K3),IS,IC)
      STOP
      END
*************************************************************************
      SUBROUTINE INPXDX(XX,DXM,XXU,XXL,GGU,NX,NS)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XX(NX),DXM(NX),XXU(NX),XXL(NX),GGU(NS)
C**   ============================================================    **
C**   Read initial values XX(NX), and maximum move size DXM(NX)       **
C**   Read upper bounds  XXU(NX), and lower bounds XXL(NX)             **
C**   Read upper bounds of slack variables GGU(NS)                    **
C**   ============================================================    **
      READ(*,'(10F8.0)') (XX (J),J=1,NX)
      READ(*,'(10F8.0)') (DXM(J),J=1,NX)
      READ(*,'(10F8.0)') (XXU(J),J=1,NX)  ! If unbound, XXU(J)=XXL(J)=0
      READ(*,'(10F8.0)') (XXL(J),J=1,NX)
      READ(*,'(10F8.0)') (GGU(I),I=1,NS)  ! If unbound, GGU(I)=0
      RETURN
      END
```

```
************************************************************************
      SUBROUTINE SLP(XX,DX,DXM,XXU,XXL,GGU,GWO,GWN
     *              ,NX,NS,NE,MGW,EPS,MAXS
     *              ,A,LI,LJ,D,X,NIA,NJA,NJX,B,MI,MJ,IS,IC)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XX(NX),DX(NX),DXM(NX),XXU(NX),XXL(NX),GGU(NS)
      DIMENSION GWO(MGW),GWN(MGW)
      DIMENSION A(NIA,NJA),LI(NIA),LJ(NJA),D(NJX),X(NJX)
      DIMENSION B(NIA,NJA),MI(NIA),MJ(NJA)
C**   ============================================================    **
C**   Sequential Linear Programming by Double Secant Method          **
C**   ------------------------------------------------------------    **
C**   Input   : NX,NS,NE,MGW,EPS,MAXS,NIA,NJA,NJX,IS,IC              **
C**           : XX(NX),DXM(NX),XXU(NX),XXL(NX),GGU(NS)               **
C**   Output  : XX(1:NX),GWO(1:NS+NE),GWO(NS+NE+1)                   **
C**   Working : DX(NX),GWN(MGW),A(NIA,NJA)                           **
C**           : A,LI,LJ,D,X : For linear programming (see LINEAR)    **
C**           : MI,MJ : For linear programming (required if IS<>0)   **
C**           : B     : For linear programming (required if IS< 0)   **
C**   ------------------------------------------------------------    **
C*I   NX        = Number of original variables                       **
C*I   NS        = Number of .LE. constraints                         **
C*I   NE        = Number of .EQ. constraints                         **
C*I   MGW      >= NGW  = NS+NE+1                                     **
C*I   EPS       = Relative error for convergence control            **
C*I   MAXS      = Maximum allowed numbers of linear programming     **
C*I   NIA      >= NGWA = NS+NE+2  : Dimension of Simplex tabulea    **
C*I   NJA      >= NXXB = NX+NX+1  : Dimension of Simplex tabulea    **
C*I   NJX      >= NXXS = NX+NX+NS : Varˆ+, Varˆ-, Slack-var         **
C*I   IS,IC     = See SUBROUTINE LINEAR                             **
C*I   XX(NX)    = Initial values of original variables              **
C*O             = Optimum values of original variables              **
C*W   DX(NX)    = Move limits                                       **
CoI   DXM(NX)   = Maximum Move size                                 **
CoI   XXU(NX)   = Upper bounds of original variables                **
CoI   XXL(NX)   = Lower bounds of original variables                **
CoI   GGU(NS)   = Upper bounds of slack variables for .LE. constr.  **
C*O   GWO(1:NG) = Constrained values at optimum (NG=NS+NE)          **
C*O   GWO(NG+1) = Objective function at optimum                     **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Set constants                                            |    **
C**   +----------------------------------------------------------+    **
      NGW =NS+NE+1
      NGWA=NS+NE+2
      NXXB=NX+NX+1
      NXXS=NX+NX+NS
C**   +----------------------------------------------------------+    **
C**   | Check initial setting of XX,DXM,XXU,XXL(NX) and GGU(NS)  |    **
C**   | Set initial DX(NX), WOLD, NSH = Number of LP performed   |    **
C**   +----------------------------------------------------------+    **
C**   CALL INPXDX(XX,DXM,XXU,XXL,GGU,NX,NS)
      DO 120 J=1,NX
      IF(DXM(J).LE.0.0) DXM(J)=DMAX1(0.10*DABS(XX(J)),0.10D0)
      DX(J)=DABS(DXM(J))
      IF(XXU(J).GT.XXL(J)) GO TO 120
      XXU(J)= 1.0D10                    ! Set to infinity for XXU=XXL=0
      XXL(J)=-1.0D10
  120 XX (J)=DMAX1(DMIN1(XXU(J),XX(J)),XXL(J))
```

```
      DO 130 I=1,NS
      IF(GGU(I).LE.0.0) GGU(I)=1.0D10    ! Set to infinity for GGU=0
  130 CONTINUE
      WOLD=0.0                           ! Old objective function
      NSH =0                             ! Iteration count
C**   +-------------------------------------------------------+    **
C**   | Call DSM for SLP by Double Secant Method              |    **
C**   | To find X(NXXS) from given XX(NX),DX(NX)              |    **
C**   +-------------------------------------------------------+    **
  200 CALL DSM(XX,DX,XXU,XXL,GGU,GWO,GWN,NX,NS,NE,MGW,ICASE,KKK
     *        ,A,LI,LJ,D,X,NIA,NJA,NJX,B,MI,MJ,IS,IC)
C**   +-------------------------------------------------------+    **
C**   | Update variables XX(NX), move limits DX(NX) & count NSH |    **
C**   +-------------------------------------------------------+    **
      DXMAX=0.0                                  ! For converge. test
      DO 230 J=1,NX
      DXJ=X(2*J-1)-X(2*J)                        ! Movement of var. J
      DXNJ=DABS(DXJ)
      DXMJ=DABS(DXM(J))
      DXMAX=DMAX1(DXMAX,DXNJ/DXMJ)               ! Maximum movement
      IF(DXJ*DXM(J).LT.0.AND.NSH.NE.0) DXNJ=DXNJ/2 ! Reduce move limit
      DX(J)=DMAX1(DMIN1((1+DXNJ/DX(J))*DXNJ,DXMJ)  ! if move dir. change
     *        ,0.25*DX(J),0.04*DXMJ,4.0*EPS*DXMJ)  ! Update DX(J)
      DXM(J)=DSIGN(DXMJ,DXJ)                     ! Save move direction
  230 XX(J)=XX(J)+DXJ                            ! Update XX(J)
      NSH=NSH+1                                  ! Iteration count
C**   +-------------------------------------------------------+    **
C**   | Output intermediate results                           |    **
C**   +-------------------------------------------------------+    **
      CALL OUTMED(XX,DX,X,D,NX,NXXS,NSH,ICASE,KKK)
C**   +-------------------------------------------------------+    **
C**   | Test convergence                                      |    **
C**   +-------------------------------------------------------+    **
      WDIF=WOLD-GWO(NGW)
      WOLD=GWO(NGW)                          ! Save objective function
      IF(NSH.GE.MAXS) GO TO 300              ! Maximum allowed number
      IF(ICASE.NE.0) GO TO 200               ! Infeasible or unbounded
      IF(DABS(WDIF).LE.0.001*DABS(WOLD)) GO TO 300  ! Objective function
      IF(DXMAX.GT.EPS) GO TO 200             ! Max. movement too small
C**   +-------------------------------------------------------+    **
C**   | Output final results & return                         |    **
C**   +-------------------------------------------------------+    **
  300 CALL FFF(NX,NS,NE,XX,GWO(NGW),GWO)
      CALL OUTFNL(NSH,NX,NS,NE,XX,GWO(NGW),GWO)
      RETURN
      END
*************************************************************************
      SUBROUTINE DSM(XX,DX,XXU,XXL,GGU,GWO,GWN,NX,NS,NE,MGW,ICASE,KKK
     *              ,A,LI,LJ,D,X,NIA,NJA,NJX,B,MI,MJ,IS,IC)
C**   =========================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XX(NX),DX(NX),XXU(NX),XXL(NX),GGU(NS)
      DIMENSION GWO(MGW),GWN(MGW)
      DIMENSION A(NIA,NJA),LI(NIA),LJ(NJA),D(NJX),X(NJX)
      DIMENSION B(NIA,NJA),MI(NIA),MJ(NJA)
      DATA SPOS,SNEG/1.0D0,-1.0D0/
C**   =========================================================    **
C**   Double Secant Method for Sequential Linear Programming      **
C**   Setup simplex tableau and call LINEAR for linear programming    **
C**   ---------------------------------------------------------    **
```

```
C**   Input   : NX,NS,NE,MGW,EPS,MAXS,NIA,NJA,NJX,IS,IC               **
C**           : XX(NX),DX(NX),XXU(NX),XXL(NX),GGU(NS)                 **
C**   Working : GWO(MGW),GWN(MGW)                                     **
C**           : A,LI,LJ,D,X,B,MI,MJ : For linear programming          **
C**   ============================================================    **
C**   +----------------------------------------------------------+    **
C**   | Set constants                                            |    **
C**   +----------------------------------------------------------+    **
      NGW =NS+NE+1
      NXX =NX+NX
      NXXB=NXX+1
C**   +----------------------------------------------------------+    **
C**   | Compute GWO(1:NG)=G(I), GWO(NG+1)=F_obj at XX(NX)         |    **
C**   | Setup Simplex Tableau : A(1:NG+1,1:NX+NX)    (NG=NS+NE)  |    **
C**   +----------------------------------------------------------+    **
      CALL FFF(NX,NS,NE,XX,GWO(NGW),GWO)
      CALL GRDGW(XX,DX,GWO,GWN,A(1,1),SPOS,NX,NS,NE,MGW,NIA+NIA)
      CALL GRDGW(XX,DX,GWO,GWN,A(1,2),SNEG,NX,NS,NE,MGW,NIA+NIA)
C**   ------------------------------------------------------------    **
      DO 20 J=1,NXX,2            ! \ . Use \./ if this . /
      DO 20 I=1,NS+NE            !   \.  is feasible  ./      \./ secant
      ASUM=A(I,J)+A(I,J+1)       !     +    region   +
      IF(ASUM.GE.0.0) GO TO 20   !      .\         /.      ... nonlinear
      AVAG=0.5*(A(I,J)-A(I,J+1)) !        .\     /.            constrain
      A(I,J)  = AVAG             ! _________\./__________
      A(I,J+1)=-AVAG             ! Use _._ if this is feasible region
   20 CONTINUE                   ! _._ tangent
C**   +----------------------------------------------------------+    **
C**   | Set upper bound D(J) and variable index LJ(J)            |    **
C**   +----------------------------------------------------------+    **
      J=0
      DO 30 JJ=1,NX
      J=J+1
      LJ(J)=J
      D(J)=MIN(DX(JJ),XXU(JJ)-XX(JJ))
      J=J+1
      LJ(J)=J
   30 D(J)=MIN(DX(JJ),XX(JJ)-XXL(JJ))
C**   +----------------------------------------------------------+    **
C**   | Set upper bound D(NXX+I) of slack var. & basic var. index|    **
C**   | Set Simplex tableau : A(1:NG+1,NX+NX+1) = RHS of constra.|    **
C**   +----------------------------------------------------------+    **
      NXXS=NXX
      DO 40 I=1,NGW
      LI(I)=0
      IF(I.GT.NS) GO TO 40
      NXXS=NXXS+1
      LI(I)=NXXS
      D(NXXS)=GGU(I)
   40 A(I,NXXB)=-GWO(I)
C**   +----------------------------------------------------------+    **
C**   | Perform linear programming                               |    **
C**   +----------------------------------------------------------+    **
      CALL LINEAR
     *(A,LI,LJ,D,X,NGW,NXXB,NXXS,ICASE,KKK,NIA,NJA,B,MI,MJ,IS,IC)
      RETURN
      END
*************************************************************************
      SUBROUTINE GRDGW(XX,DX,GWO,GWN,DGW,S,NX,NS,NE,MGW,NIA2)
C**   ============================================================    **
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XX(NX),DX(NX),GWO(MGW),GWN(MGW),DGW(NIA2,NX)
C**   ============================================================    **
C**   Compute gradients by forward or backward difference             **
C**   ------------------------------------------------------------    **
C*I   XX(NX)    = Values of variables                                 **
C*I   DX(NX)    = Delta Values of variables > 0                       **
C*I   GWO(NG+1) = Constra. G(1:NG) & Obj.Fun. F_obj at XX(NX)         **
C*W   GWN(NG+1) = Constra. G(1:NG) & Obj.Fun. F_obj at XX(NX)+DX(J)   **
C*O   DGW(I,J)  = (G_I(XX(J)+DX(J))-G_I(XX(J)))/DX(J) if S=+1         **
C*O   DGW(I,J)  = (G_I(XX(J)-DX(J))-G_I(XX(J)))/DX(J) if S=-1         **
C*I   NX        = Number of original variables    (NG=NS+NE)          **
C*I   NS,NE     = Number of .LE.,.EQ. constraints (NGW=NG+1)          **
C**   ============================================================    **
      DO 40 J=1,NX
C**   +----------------------------------------------------------+    **
C**   | Compute constraint GWN(1:NS+NE)=G(I) and GWN(NGW)=F_obj  |    **
C**   +----------------------------------------------------------+    **
      XXJ=XX(J)
      XX(J)=XX(J)+DX(J)*S
      DXP=DABS(XX(J)-XXJ)
      CALL FFF(NX,NS,NE,XX,GWN(NS+NE+1),GWN)
      XX(J)=XXJ
C**   +----------------------------------------------------------+    **
C**   | Compute DGW(I,J)=dG(I)/dX(J) and DGW(NGW,J)=dF_obj/dX(J) |    **
C**   +----------------------------------------------------------+    **
      DO 40 I=1,NS+NE+1
   40 DGW(I,J)=(GWN(I)-GWO(I))/DXP
      RETURN
      END
*************************************************************************
      SUBROUTINE OUTMED(XX,DX,X,D,NX,NXXS,NSH,ICASE,KKK)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XX(NX),DX(NX),X(NXXS),D(NXXS)
C**   ============================================================    **
C**   Output intermedeate results                                     **
C**   ============================================================    **
      WRITE(*,'(1X,29(''-''))/'' NSH,ICASE,KKK ='',3I5)') NSH,ICASE,KKK
      WRITE(*,'('' D =''/(1X,1P8E10.3))') (D(J),J=1,NXXS)
      WRITE(*,'('' X =''/(1X,1P8E10.3))') (X(J),J=1,NXXS)
      WRITE(*,'('' DX =''/(1X,1P8E10.3))') (DX(J),J=1,NX)
      WRITE(*,'('' XX =''/(1X,1P8E10.3))') (XX(J),J=1,NX)
      RETURN
      END
*************************************************************************
      SUBROUTINE OUTFNL(NSH,NX,NS,NE,XX,WO,GWO)
C**   ============================================================    **
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XX(NX),GWO(NS+NE)
C**   ============================================================    **
C**   Output final results                                            **
C**   ============================================================    **
      WRITE(*,'(1X,9(''=''))/A,I4,A)') ' After',NSH,' Linear Programming'
      WRITE(*,'(A,1PE20.9)') ' The optimum objective function is',WO
      WRITE(*,'(/'' Values of variables ''//(1X,1P8E10.3))') XX
      WRITE(*,'(/'' Values of constraint''//(1X,1P8E10.3))') GWO
      RETURN
      END
*************************************************************************
```

21.12 準牛頓法之推導

前面所提到的準牛頓法為黑森矩陣之逆矩陣採用DFP公式做修訂。事實上所有之黑森矩陣或其逆矩陣之近似求法均稱為準牛頓法，而其有關公式亦為數不少，故專闢一節做一系統性的推導。本節之符號採用文獻[6]書中之符號。 下標c、下標$+$、下標$++$分別表示前一點、目前點、下一點之函數值，向量或矩陣。向量以小寫粗體表示，矩陣以大寫粗體表示。

首先考慮下列之單變數非線性方程式之二種解法：(1)牛頓法；(2)割線法。

$$f(x)=0 \tag{21.105}$$

(1)牛頓法：由已知之$f(x_+)$與$f'(x_+)$，以一階之近似式$n_+(x)$近似$f(x)$，解$n_+(x)=0$所得之x_{++}做為$f(x)=0$之近似解。

$$f(x)\approx n_+(x) \;=\; f(x_+)+f'(x_+)(x-x_+) \tag{21.106}$$

$$x_{++} \;=\; x_+ - f(x_+)/f'(x_+) \tag{21.107}$$

(2)割線法：由已知之$f(x_+)$與$f(x_c)$，以一階之近似式$m_+(x)$近似$f(x)$，解$m_+(x)=0$所得之x_{++}做為$f(x)=0$之近似解。

$$f(x)\approx m_+(x) \;=\; f(x_+)+a_+(x-x_+) \tag{21.108}$$

$$x_{++} \;=\; x_+ - f(x_+)/a_+ \tag{21.109}$$

上式中之a_+可選用滿足下列二項條件之值：

$$(a)\;: \qquad m_+(x_+) \;=\; f(x_+) \tag{21.110}$$

$$(b)\;: \qquad m_+(x_c) \;=\; f(x_c) \tag{21.111}$$

條件(a)不管a_+選用何值都會滿足；但條件(b)必須a_+選用下式之由式(21.113)所得之值才會滿足：

$$a_+=\frac{f(x_c)-f(x_+)}{x_c-x_+} \tag{21.112}$$

$$f(x_+)+a_+(x_c-x_+)=f(x_c) \tag{21.113}$$

因此即得$f(x)=0$之近似解為得自割線法之下列x_{++}。

$$x_{++}=x_+-f(x_+)(x_c-x_+)/(f(x_c)-f(x_+)) \tag{21.114}$$

現考慮下列之n變數之n元非線性聯立方程式之二種解法：(1)牛頓法；(2)割線法。

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{21.115}$$

(1)牛頓法：由已知之$\mathbf{f}(\mathbf{x}_+)$與$\mathbf{J}(\mathbf{x}_+)$，$\mathbf{J}$稱為賈柯比(Jacobi)矩陣，以一階之近似式$\mathbf{n}_+(\mathbf{x})$近似$\mathbf{f}(\mathbf{x})$，解$\mathbf{n}_+(\mathbf{x}) = \mathbf{0}$所得之$\mathbf{x}_{++}$做為$\mathbf{f}(\mathbf{x}) = \mathbf{0}$之近似解。注意式(21.116)對應於式(21.20)，此時賈柯比矩陣$\mathbf{J}$對應於黑森矩陣$\mathbf{H}$，而為對稱矩陣。

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{n}_+(\mathbf{x}) = \mathbf{f}(\mathbf{x}_+) + \mathbf{J}_+(\mathbf{x} - \mathbf{x}_+) \tag{21.116}$$

$$\mathbf{x}_{++} = \mathbf{x}_+ - \mathbf{J}_+^{-1}\mathbf{f}(\mathbf{x}_+) \tag{21.117}$$

(2)割線法：由已知之$\mathbf{f}(\mathbf{x}_+)$與$\mathbf{f}(\mathbf{x_c})$，以一階之近似式$\mathbf{m}_+(\mathbf{x})$近似$\mathbf{f}(\mathbf{x})$，解$\mathbf{m}_+(\mathbf{x}) = \mathbf{0}$所得之$\mathbf{x}_{++}$做為$\mathbf{f}(\mathbf{x}) = \mathbf{0}$之近似解。

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{m}_+(\mathbf{x}) = \mathbf{f}(\mathbf{x}_+) + \mathbf{A}_+(\mathbf{x} - \mathbf{x}_+) \tag{21.118}$$

$$\mathbf{x}_{++} = \mathbf{x}_+ - \mathbf{A}_+^{-1}\mathbf{f}(\mathbf{x}_+) \tag{21.119}$$

上式中之$\mathbf{A}_+$可選用滿足下列二項條件之值：

$$(a): \quad \mathbf{m}_+(\mathbf{x}_+) = \mathbf{f}(\mathbf{x}_+) \tag{21.120}$$

$$(b): \quad \mathbf{m}_+(\mathbf{x_c}) = \mathbf{f}(\mathbf{x_c}) \tag{21.121}$$

條件(a)不管$\mathbf{A}_+$選用何值都會滿足；但條件(b)必須$\mathbf{A}_+$選用一些特別的值才會滿足。但$\mathbf{A}_+$之決定並不像單變數之a_+那樣單純，因為$\mathbf{A}_+$有n^2個值待定，而條件(b)只有n個方程式。因此僅由條件(b)並無法求得唯一解，而可以用許多種替代方式來決定$\mathbf{A}_+$。為求得唯一解，一些常用的附加條件有下列數種：(1)極小化：(1a) $||\mathbf{A}_+ - \mathbf{A_c}||_\mathbf{F}$；或(1b) $||\mathbf{A}_+^{-1} - \mathbf{A_c^{-1}}||_\mathbf{F}$；或(1c) $||\mathbf{K}_+ - \mathbf{L_c}||_\mathbf{F}$；或(1d) $||\mathbf{K}_+^{-\mathbf{T}} - \mathbf{L_c^{-T}}||_\mathbf{F}$。其中$\mathbf{L_c}$為下三角矩陣且$\mathbf{L_c}\mathbf{L_c^T} = \mathbf{A_c}$，而$\mathbf{K}_+\mathbf{K}_+^\mathbf{T} = \mathbf{A}_+$但$\mathbf{K}_+$不是三角矩陣。(2)限制矩陣差之型式為：(2a) $\mathbf{A}_+ - \mathbf{A_c} = \mathbf{u}\mathbf{v}^\mathbf{T}$；或(2b) $\mathbf{A}_+^{-1} - \mathbf{A_c^{-1}} = \bar{\mathbf{u}}\bar{\mathbf{v}}^\mathbf{T}$；或(2c) $\mathbf{K}_+ - \mathbf{L_c} = \tilde{\mathbf{u}}\tilde{\mathbf{v}}^\mathbf{T}$；或(2d) $\mathbf{K}_+^{-\mathbf{T}} - \mathbf{L_c^{-T}} = \tilde{\tilde{\mathbf{u}}}\tilde{\tilde{\mathbf{v}}}^\mathbf{T}$。(3)限制$\mathbf{u}$，$\mathbf{v}$等為：(3a) $n \times 1$之矩陣且$\mathbf{u} = \mathbf{v}$，稱一階對稱；或(3b) $n \times 1$之矩陣，稱一階不對稱；或(3c) $n \times 2$之矩陣且$\mathbf{u} = \mathbf{v}$，稱二階對稱。

上述各附加條件需用到前一點之 $\mathbf{A_c}$ ，或 $\mathbf{A_c^{-1}}$ ，或 $\mathbf{L_c}$ ，或 $\mathbf{L_c^{-1}}$ 。因此將前一點之 $\mathbf{m_c(x)}$ 列示於式 (21.122) 。同時亦將新點之 $\mathbf{m_+(x)}$ 列示於式 (21.123) ；並按條件 (2) 分別寫成式 (21.124) 至式 (21.127) 。

$$\mathbf{m_c\,(x)} = \mathbf{f(x_c\,)} + \mathbf{A_c\,(x - x_c\,)} = \mathbf{f(x_c\,)} + \mathbf{L_c\;L_c^T\;(x - x_c\,)} \quad (21.122)$$

$$\mathbf{m_+(x)} = \mathbf{f(x_+)} + \mathbf{A_+(x - x_+)} = \mathbf{f(x_+)} + \mathbf{K_+K_+^T(x - x_+)} \quad (21.123)$$

$$= \mathbf{f(x_+)} + (\mathbf{A_c} + \mathbf{uv^T})\,(\mathbf{x - x_+}) \quad (21.124)$$

$$= \mathbf{f(x_+)} + (\mathbf{A_c^{-1}} + \mathbf{\bar{u}\bar{v}^T})^{-1}(\mathbf{x - x_+}) \quad (21.125)$$

$$= \mathbf{f(x_+)} + (\mathbf{L_c} + \mathbf{\tilde{u}\tilde{v}^T})(\mathbf{L_c} + \mathbf{\tilde{u}\tilde{v}^T})^{\mathbf{T}}(\mathbf{x - x_+}) \quad (21.126)$$

$$= \mathbf{f(x_+)} + [(\mathbf{L_c^{-T}} + \mathbf{\tilde{\tilde{u}}\tilde{\tilde{v}}^T})(\mathbf{L_c^{-T}} + \mathbf{\tilde{\tilde{u}}\tilde{\tilde{v}}^T})^{\mathbf{T}}]^{-1}(\mathbf{x - x_+}) \quad (21.127)$$

準牛頓法亦可根據比例調整後之新變數 $\mathbf{\hat{x} = T\,x}$ ，計算對應之黑森矩陣 $\mathbf{\hat{A} = T^{-T}AT^{-1}}$ 。若新變數使 $\mathbf{\hat{A}_+ = T^{-T}A_+T^{-1} = I}$ ，即 $\mathbf{A_+ = T^TT}$ ，則可得下列關係：

$$\mathbf{T^{-T}m_c\,(x)} = \mathbf{T^{-T}f(x_c\,)} + \mathbf{T^{-T}A_c\,T^{-1}(Tx - Tx_c\,)} \quad (21.128)$$

$$\mathbf{T^{-T}m_+(x)} = \mathbf{T^{-T}f(x_+)} + \mathbf{T^{-T}A_+T^{-1}(Tx - Tx_+)} \quad (21.129)$$

$$\mathbf{\hat{m}_+(T^{-1}\hat{x})} = \mathbf{\hat{f}(T^{-1}\hat{x}_+)} + \mathbf{\hat{A}_+(\hat{x} - \hat{x}_+)}$$

$$= \mathbf{\hat{f}(T^{-1}\hat{x}_+)} + (\mathbf{\hat{A}_c} + \mathbf{\hat{u}\hat{v}^T})\,(\mathbf{\hat{x} - \hat{x}_+}) \quad (21.130)$$

$$= \mathbf{\hat{f}(T^{-1}\hat{x}_+)} + (\mathbf{\hat{A}_c^{-1}} + \mathbf{\hat{\bar{u}}\hat{\bar{v}}^T})^{-1}(\mathbf{\hat{x} - \hat{x}_+}) \quad (21.131)$$

由條件 (b) 與 (2a) 之式 (21.121) 與式 (21.124) ，可得：

$$\mathbf{f_c - f_+} = (\mathbf{A_c + uv^T})(\mathbf{x_c - x_+}) \quad (21.132)$$

令 $$\mathbf{y_c} = \mathbf{f_c - f_+}\,, \quad \mathbf{s_c} = \mathbf{x_c - x_+} \quad (21.133)$$

$$\mathbf{r_c} = \mathbf{y_c - A_cs_c} \quad (21.134)$$

得 $$\mathbf{r_c} = \mathbf{uv^T\,s_c} \quad (21.135)$$

由條件 (b) 與 (2b) 之式 (21.121) 與式 (21.125) ，可得：

$$\mathbf{x_c - x_+} = (\mathbf{A_c^{-1} + \bar{u}\bar{v}^T})(\mathbf{f_c - f_+}) \quad (21.136)$$

令 $$\mathbf{s_c} = \mathbf{x_c - x_+}\,, \quad \mathbf{y_c} = \mathbf{f_c - f_+} \quad (21.137)$$

$$\mathbf{\bar{r}_c} = \mathbf{s_c - A_c^{-1}y_c} \quad (21.138)$$

得 $$\mathbf{\bar{r}_c} = \mathbf{\bar{u}\bar{v}^T\,y_c} \quad (21.139)$$

由條件(b)與(2a)或(2b)之式(21.121)與式(21.130)或式(21.131)，可得：

$$\hat{\mathbf{f}}_{\mathbf{c}} - \hat{\mathbf{f}}_{+} = (\hat{\mathbf{A}}_{\mathbf{c}} + \hat{\mathbf{u}}\hat{\mathbf{v}}^{\mathbf{T}})(\hat{\mathbf{x}}_{\mathbf{c}} - \hat{\mathbf{x}}_{+}) \tag{21.140}$$

$$\hat{\mathbf{x}}_{\mathbf{c}} - \hat{\mathbf{x}}_{+} = (\hat{\mathbf{A}}_{\mathbf{c}}^{-1} + \hat{\mathbf{u}}\hat{\mathbf{v}}^{\mathbf{T}})(\hat{\mathbf{f}}_{\mathbf{c}} - \hat{\mathbf{f}}_{+}) \tag{21.141}$$

令

$$\hat{\mathbf{y}}_{\mathbf{c}} = \hat{\mathbf{f}}_{\mathbf{c}} - \hat{\mathbf{f}}_{+} = \mathbf{T}^{-\mathbf{T}}\mathbf{y}_{\mathbf{c}} \tag{21.142}$$

$$\hat{\mathbf{s}}_{\mathbf{c}} = \hat{\mathbf{x}}_{\mathbf{c}} - \hat{\mathbf{x}}_{+} = \mathbf{T}\mathbf{s}_{\mathbf{c}} \tag{21.143}$$

$$\hat{\mathbf{r}}_{\mathbf{c}} = \hat{\mathbf{y}}_{\mathbf{c}} - \hat{\mathbf{A}}_{\mathbf{c}}\hat{\mathbf{s}}_{\mathbf{c}} = \mathbf{T}^{-\mathbf{T}}(\mathbf{y}_{\mathbf{c}} - \mathbf{A}_{\mathbf{c}}\mathbf{s}_{\mathbf{c}}) = \mathbf{T}^{-\mathbf{T}}\mathbf{r}_{\mathbf{c}} \tag{21.144}$$

$$\hat{\bar{\mathbf{r}}}_{\mathbf{c}} = \hat{\mathbf{s}}_{\mathbf{c}} - \hat{\mathbf{A}}_{\mathbf{c}}^{-1}\hat{\mathbf{y}}_{\mathbf{c}} = \mathbf{T}(\mathbf{s}_{\mathbf{c}} - \mathbf{A}_{\mathbf{c}}^{-1}\mathbf{y}_{\mathbf{c}}) = \mathbf{T}\bar{\mathbf{r}}_{\mathbf{c}} \tag{21.145}$$

得

$$\hat{\mathbf{r}}_{\mathbf{c}} = \hat{\mathbf{u}}\hat{\mathbf{v}}^{\mathbf{T}}\hat{\mathbf{s}}_{\mathbf{c}} \tag{21.146}$$

$$\hat{\bar{\mathbf{r}}}_{\mathbf{c}} = \hat{\bar{\mathbf{u}}}\hat{\bar{\mathbf{v}}}^{\mathbf{T}}\hat{\mathbf{y}}_{\mathbf{c}} \tag{21.147}$$

由條件(b)與(2c)之式(21.121)與式(21.126)，可得：

$$\mathbf{f}_{\mathbf{c}} - \mathbf{f}_{+} = (\mathbf{L}_{\mathbf{c}} + \tilde{\mathbf{u}}\tilde{\mathbf{v}}^{\mathbf{T}})(\mathbf{L}_{\mathbf{c}} + \tilde{\mathbf{u}}\tilde{\mathbf{v}}^{\mathbf{T}})^{\mathbf{T}}(\mathbf{x}_{\mathbf{c}} - \mathbf{x}_{+}) \tag{21.148}$$

令

$$\mathbf{y}_{\mathbf{c}} = \mathbf{f}_{\mathbf{c}} - \mathbf{f}_{+}\,, \qquad \mathbf{s}_{\mathbf{c}} = \mathbf{x}_{\mathbf{c}} - \mathbf{x}_{+} \tag{21.149}$$

$$\mathbf{K}_{+} = \mathbf{L}_{\mathbf{c}} + \tilde{\mathbf{u}}\tilde{\mathbf{v}}^{\mathbf{T}} \tag{21.150}$$

及

$$\tilde{\mathbf{s}}_{\mathbf{c}} = \mathbf{K}_{+}^{\mathbf{T}}\mathbf{s}_{\mathbf{c}} \tag{21.151}$$

$$\tilde{\mathbf{r}}_{\mathbf{c}} = \mathbf{y}_{\mathbf{c}} - \mathbf{L}_{\mathbf{c}}\tilde{\mathbf{s}}_{\mathbf{c}} \tag{21.152}$$

得

$$\mathbf{y}_{\mathbf{c}} = \mathbf{K}_{+}\mathbf{K}_{+}^{\mathbf{T}}\mathbf{s}_{\mathbf{c}} = \mathbf{K}_{+}\tilde{\mathbf{s}}_{\mathbf{c}} \tag{21.153}$$

$$\tilde{\mathbf{r}}_{\mathbf{c}} = \tilde{\mathbf{u}}\tilde{\mathbf{v}}^{\mathbf{T}}\tilde{\mathbf{s}}_{\mathbf{c}} \tag{21.154}$$

由條件(b)與(2c)之式(21.121)與式(21.127)，可得：

$$\mathbf{x}_{\mathbf{c}} - \mathbf{x}_{+} = (\mathbf{L}_{\mathbf{c}}^{-\mathbf{T}} + \tilde{\bar{\mathbf{u}}}\tilde{\bar{\mathbf{v}}}^{\mathbf{T}})(\mathbf{L}_{\mathbf{c}}^{-\mathbf{T}} + \tilde{\bar{\mathbf{u}}}\tilde{\bar{\mathbf{v}}}^{\mathbf{T}})^{\mathbf{T}}(\mathbf{f}_{\mathbf{c}} - \mathbf{f}_{+}) \tag{21.155}$$

令

$$\mathbf{s}_{\mathbf{c}} = \mathbf{x}_{\mathbf{c}} - \mathbf{x}_{+}\,, \qquad \mathbf{y}_{\mathbf{c}} = \mathbf{f}_{\mathbf{c}} - \mathbf{f}_{+} \tag{21.156}$$

$$\mathbf{K}_{+}^{-\mathbf{T}} = \mathbf{L}_{\mathbf{c}}^{-\mathbf{T}} + \tilde{\bar{\mathbf{u}}}\tilde{\bar{\mathbf{v}}}^{\mathbf{T}} \tag{21.157}$$

及

$$\tilde{\mathbf{y}}_{\mathbf{c}} = \mathbf{K}_{+}^{-1}\mathbf{y}_{\mathbf{c}} \tag{21.158}$$

$$\tilde{\bar{\mathbf{r}}}_{\mathbf{c}} = \mathbf{s}_{\mathbf{c}} - \mathbf{L}_{\mathbf{c}}^{-\mathbf{T}}\tilde{\mathbf{y}}_{\mathbf{c}} \tag{21.159}$$

得

$$\mathbf{s}_{\mathbf{c}} = \mathbf{K}_{+}^{-\mathbf{T}}\mathbf{K}_{+}^{-1}\mathbf{y}_{\mathbf{c}} = \mathbf{K}_{+}^{-\mathbf{T}}\tilde{\mathbf{y}}_{\mathbf{c}} \tag{21.160}$$

$$\tilde{\bar{\mathbf{r}}}_{\mathbf{c}} = \tilde{\bar{\mathbf{u}}}\tilde{\bar{\mathbf{v}}}^{\mathbf{T}}\tilde{\mathbf{y}}_{\mathbf{c}} \tag{21.161}$$

經附加條件(2)之限制後，求$\mathbf{A}_{+}$之問題即改為求諸$\mathbf{u}$與$\mathbf{v}$之問題。即求式(21.135)、式(21.139)、式(21.146)、式(21.147)（簡稱式(21.135)等四

式），或式(21.154)、式(21.161)（簡稱式(21.154)等二式）之諸$\mathbf{u}$與$\mathbf{v}$。因黑森矩陣為對稱，故條件(3a)或(3c)用於式(21.135)等四式；而條件(3b)則僅用於式(21.154)等二式。若用條件(3b)或(3c)，則未知係數仍有$2n$個，需再加條件(1a)或(1b)。若用條件(3a)，則未知係數僅n個，即可直接由式(21.135)等四式求得諸$\mathbf{u}$。以下將以式(21.135)為代表性範例說明$\mathbf{u}$與$\mathbf{v}$之求法。推導時$\mathbf{r_c}$, $\mathbf{s_c}$, $\mathbf{y_c}$之下標c省略。

條件(3a)之一階對稱情形：因$\mathbf{v}=\mathbf{u}$，式(21.135)可寫成式(21.162)，因式中$(\mathbf{u^T s})$為純量，故$\mathbf{u}$與$\mathbf{r}$互相平行，可令$\mathbf{u}=\alpha\mathbf{r}$代入此式即得式(21.163)，由該式可得式(21.164)之α值，因此即得式(21.165)之$\mathbf{u}$。

$$\mathbf{r} = (\mathbf{u}\mathbf{u}^{\mathbf{T}})\mathbf{s} = \mathbf{u}(\mathbf{u}^{\mathbf{T}}\mathbf{s}) \tag{21.162}$$

$$= \alpha\mathbf{r}(\alpha\mathbf{r}^{\mathbf{T}}\mathbf{s}) = \alpha^{\mathbf{2}}\mathbf{r}(\mathbf{r}^{\mathbf{T}}\mathbf{s}) \tag{21.163}$$

$$\mathbf{u} = \alpha\mathbf{r}\,, \qquad \alpha = \frac{\mathbf{1}}{\sqrt{\mathbf{r}^{\mathbf{T}}\mathbf{s}}} \tag{21.164}$$

$$\mathbf{u} = \frac{\mathbf{r}}{\sqrt{\mathbf{r}^{\mathbf{T}}\mathbf{s}}} \tag{21.165}$$

由上式所得之$\mathbf{A}_+$有下列二種。以比例調整新變數之式(21.146)與式(21.147)所得者轉為原變數後與下式相同，即此種求法不受比例調整之影響。事實上由Sherman-Morrison-Woodbury公式可証明下列二式所得之$\mathbf{A}_+$相同。該式稱為$SR1$（Symmetric rank-one）修訂式。

$$\mathbf{A}_+ = \mathbf{A_c} + \frac{(\mathbf{y_c}-\mathbf{A_c}\mathbf{s_c})(\mathbf{y_c}-\mathbf{A_c}\mathbf{s_c})^{\mathbf{T}}}{(\mathbf{y_c}-\mathbf{A_c}\mathbf{s_c})^{\mathbf{T}}\mathbf{s_c}} \tag{21.166}$$

$$\mathbf{A}_+^{-\mathbf{1}} = \mathbf{A_c^{-1}} + \frac{(\mathbf{s_c}-\mathbf{A_c^{-1}}\mathbf{y_c})(\mathbf{s_c}-\mathbf{A_c^{-1}}\mathbf{y_c})^{\mathbf{T}}}{(\mathbf{s_c}-\mathbf{A_c^{-1}}\mathbf{y_c})^{\mathbf{T}}\mathbf{y_c}} \tag{21.167}$$

條件(3b)之一階不對稱情形：式(21.135)可寫成式(21.168)，因式中$(\mathbf{v^T s})$為純量，故$\mathbf{u}$與$\mathbf{r}$互相平行，可令$\mathbf{u}=\alpha\mathbf{r}$代入此式即得式(21.169)，由該式可得式(21.170)之α值，因此即得式(21.171)之$\mathbf{u}$。

$$\mathbf{r} = (\mathbf{u}\mathbf{v}^{\mathbf{T}})\mathbf{s} = \mathbf{u}(\mathbf{v}^{\mathbf{T}}\mathbf{s}) \tag{21.168}$$

$$= \alpha\mathbf{r}(\mathbf{v}^{\mathbf{T}}\mathbf{s}) \tag{21.169}$$

$$\mathbf{u} = \alpha\mathbf{r}\,, \qquad \alpha = \frac{\mathbf{1}}{\mathbf{v}^{\mathbf{T}}\mathbf{s}} \tag{21.170}$$

$$\mathbf{u} = \frac{\mathbf{r}}{\mathbf{v}^{\mathbf{T}}\mathbf{s}}\,, \quad \mathbf{v}\text{ 可任選但須不垂直於 }\mathbf{s} \tag{21.171}$$

$$\mathbf{v} = \mathbf{s} \qquad \text{則可以使附加條件(1)滿足} \tag{21.172}$$

式(21.172)之條件(1)，即$\mathbf{v}=\mathbf{s}$可極小化$||\mathbf{A}_+-\mathbf{A_c}||_\mathbf{F}$，証明如下：

$$\mathbf{A}_+-\mathbf{A_c} = \mathbf{u}\mathbf{v}^\mathbf{T} = \frac{\mathbf{r}\mathbf{v}^\mathbf{T}}{\mathbf{v}^\mathbf{T}\mathbf{s}} \tag{21.173}$$

$$||\mathbf{A}_+-\mathbf{A_c}||_\mathbf{F}^\mathbf{2} = \frac{\sum_i\sum_j(r_iv_j)^2}{(\sum v_ks_k)^2} = \frac{(\sum r_i^2)(\sum v_j^2)}{(\sum v_ks_k)^2} \tag{21.174}$$

$$\text{由}\quad \frac{\partial||\mathbf{A}_+-\mathbf{A_c}||_\mathbf{F}^\mathbf{2}}{\partial v_i} = (\sum r_i^2)\frac{2(\sum v_ks_k)v_i-2(\sum v_j^2)s_i}{(\sum v_ks_k)^3} = 0 \tag{21.175}$$

$$\text{可得}\quad v_i = s_i(\frac{\sum v_j^2}{\sum v_ks_k}) = s_i(\frac{\mathbf{v}^\mathbf{T}\mathbf{v}}{\mathbf{v}^\mathbf{T}\mathbf{s}}) \tag{21.176}$$

亦即 $\mathbf{v}$ // $\mathbf{s}$ 可使$||\mathbf{A}_+-\mathbf{A_c}||_\mathbf{F}^\mathbf{2}$為最小

上述結果用於式(21.135)可得Broyden好修訂式，用於式(21.139)可得Broyden壞修訂式，因不對稱故僅適用於解非線性方程式。上述結果用於式(21.154)可知下列之$\tilde{\mathbf{u}}$與$\tilde{\mathbf{v}}$可極小化$||\mathbf{K}_+-\mathbf{L_c}||_\mathbf{F}^\mathbf{2}$。

$$\tilde{\mathbf{u}} = \frac{\tilde{\mathbf{r}}_\mathbf{c}}{\tilde{\mathbf{s}}_\mathbf{c}^\mathbf{T}\tilde{\mathbf{s}}_\mathbf{c}}, \qquad \tilde{\mathbf{v}} = \tilde{\mathbf{s}}_\mathbf{c} \tag{21.177}$$

代入式(21.150)及由式(21.151)可得式(21.178)及式(21.179)。式(21.179)中括號內為一純量，該項移至等號左邊後可看出$\tilde{\mathbf{s}}_\mathbf{c}$與$\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s}_\mathbf{c}$成比例，即式(21.180)所示。將式(21.180)代入式(21.179)可得式(21.181)，進而得式(21.182)之α^2。再由式(21.180)得式(21.183)之$\tilde{\mathbf{s}}_\mathbf{c}$，及由式(21.178)得式(21.184)之$\mathbf{K}_+$，最後得式(21.185)之$\mathbf{A}_+$，稱為$BFGS$（Broydon–Fletcher–Goldfarb–Shanno）修訂式。 用類似做法於式(21.161)可得式(21.186)之$\mathbf{A}_+^{-\mathbf{1}}$，稱為$DFP$修訂式。一般情形$BFGS$較優於$DFP$修訂式，與Broyden修訂式之好壞類似。

$$\mathbf{K}_+ = \mathbf{L_c}+\frac{\tilde{\mathbf{r}}_\mathbf{c}\tilde{\mathbf{s}}_\mathbf{c}^\mathbf{T}}{\tilde{\mathbf{s}}_\mathbf{c}^\mathbf{T}\tilde{\mathbf{s}}_\mathbf{c}} = \mathbf{L_c}+\frac{(\mathbf{y_c}-\mathbf{L_c}\tilde{\mathbf{s}}_\mathbf{c})\tilde{\mathbf{s}}_\mathbf{c}^\mathbf{T}}{\tilde{\mathbf{s}}_\mathbf{c}^\mathbf{T}\tilde{\mathbf{s}}_\mathbf{c}} \tag{21.178}$$

$$\tilde{\mathbf{s}}_\mathbf{c} = \mathbf{K}_+^\mathbf{T}\mathbf{s_c} = \mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c}+\tilde{\mathbf{s}}_\mathbf{c}(\frac{(\mathbf{y_c}-\mathbf{L_c}\tilde{\mathbf{s}}_\mathbf{c})^\mathbf{T}\mathbf{s_c}}{\tilde{\mathbf{s}}_\mathbf{c}^\mathbf{T}\tilde{\mathbf{s}}_\mathbf{c}}) \tag{21.179}$$

$$\tilde{\mathbf{s}}_\mathbf{c} = \alpha\,\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c} \tag{21.180}$$

$$\alpha\,\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c} = \mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c}+\alpha\,\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c}(\frac{\mathbf{y}_\mathbf{c}^\mathbf{T}\mathbf{s_c}-\alpha\,\mathbf{s}_\mathbf{c}^\mathbf{T}\mathbf{L_c}\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c}}{\alpha^2\mathbf{s}_\mathbf{c}^\mathbf{T}\mathbf{L_c}\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c}}) \tag{21.181}$$

$$\alpha^\mathbf{2} = \frac{\mathbf{y}_\mathbf{c}^\mathbf{T}\mathbf{s_c}}{\mathbf{s}_\mathbf{c}^\mathbf{T}\mathbf{L_c}\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c}} = \frac{\mathbf{y}_\mathbf{c}^\mathbf{T}\mathbf{s_c}}{\mathbf{s}_\mathbf{c}^\mathbf{T}\mathbf{A_c}\mathbf{s_c}} \tag{21.182}$$

$$\tilde{\mathbf{s}}_\mathbf{c} = \left(\frac{\mathbf{y}_\mathbf{c}^\mathbf{T}\mathbf{s_c}}{\mathbf{s}_\mathbf{c}^\mathbf{T}\mathbf{A_c}\mathbf{s_c}}\right)^{\mathbf{1/2}}\mathbf{L}_\mathbf{c}^\mathbf{T}\mathbf{s_c} \tag{21.183}$$

$$\begin{aligned}
\mathbf{K}_+ &= \mathbf{L_c} + \frac{(\mathbf{y_c} - \alpha \mathbf{L_c}\mathbf{L_c^T}\mathbf{s_c})\alpha \mathbf{s_c^T}\mathbf{L_c}}{\alpha^2 \mathbf{s_c^T}\mathbf{L_c}\mathbf{L_c^T}\mathbf{s_c}} && (21.184)\\
\mathbf{A}_+ &= \mathbf{K}_+\mathbf{K}_+^{\mathbf{T}} \\
&= \mathbf{L_c}\mathbf{L_c^T} + \frac{(\mathbf{y_c} - \alpha \mathbf{A_c}\mathbf{s_c})\alpha \mathbf{s_c^T}\mathbf{A_c}}{\alpha^2 \mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c}} + \frac{\alpha \mathbf{A_c^T}\mathbf{s_c}(\mathbf{y_c} - \alpha \mathbf{A_c}\mathbf{s_c})^{\mathbf{T}}}{\alpha^2 \mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c}} \\
&\qquad + \frac{(\mathbf{y_c} - \alpha \mathbf{A_c}\mathbf{s_c})\alpha \mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c}\alpha(\mathbf{y_c} - \alpha \mathbf{A_c}\mathbf{s_c})^{\mathbf{T}}}{(\alpha^2 \mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c})^2} \\
&= \mathbf{A_c} + \frac{\mathbf{y_c}\mathbf{y_c^T}}{\mathbf{y_c^T}\mathbf{s_c}} - \frac{\mathbf{A_c}\mathbf{s_c}\mathbf{s_c^T}\mathbf{A_c}}{\mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c}} && (21.185)\\
\mathbf{A}_+^{-1} &= \mathbf{A_c^{-1}} + \frac{\mathbf{s_c}\mathbf{s_c^T}}{\mathbf{s_c^T}\mathbf{y_c}} - \frac{\mathbf{A_c^{-1}}\mathbf{y_c}\mathbf{y_c^T}\mathbf{A_c^{-1}}}{\mathbf{y_c^T}\mathbf{A_c^{-1}}\mathbf{y_c}} && (21.186)
\end{aligned}$$

條件(3c)之二階對稱情形：將式(21.135)寫成式(21.187)，注意其中$\mathbf{uv^T}$係由$\mathbf{rr^T}$、$\mathbf{rs^T} + \mathbf{sr^T}$、$\mathbf{ss^T}$等一階之矩陣之線性組合，係根據上述一階之結果而設定。令向量$\mathbf{r}$、$\mathbf{s}$之長分別為τ、σ，二向量之夾角為ϕ。則式中$\mathbf{r^T s} = \tau\sigma\cos\phi$、$\mathbf{s^T s} = \sigma^2$均為純量，故得式(21.188)及式(21.189)，由此可將α, β表為γ之函數如式(21.190)所示。代入式(21.191)可得式(21.192)。式(21.192)中向量$\mathbf{s}/\sigma$為單位長，在式(21.193)中以$\mathbf{h}$代之。向量$\frac{\mathbf{r}}{\tau\cos\phi}$投影至$\mathbf{s}$正好為單位長之向量$\mathbf{h}$，令$\mathbf{t} = \frac{\mathbf{r}}{\tau\cos\phi} - \mathbf{h}$，則向量$\mathbf{t}$垂直於向量$\mathbf{h}$。因此式(21.193)中以$(\mathbf{h} + \mathbf{t})$取代式(21.192)之$\frac{\mathbf{r}}{\tau\cos\phi}$。經簡化為式(21.194)後，可用以計算$||\mathbf{A}_+ - \mathbf{A_c}||_{\mathbf{F}}^{\mathbf{2}}$如式(21.195)，由該值對$\gamma$之微分等於零可得$1 - \gamma\sigma^2 = 0$，因此得式(21.199)之$\alpha$, β, γ等值，代入式(21.191)即得式(21.201)，稱為PSB修訂式(Powell Symmetric Broyden update)。注意式(21.197)中含$h_i t_i$或$h_j t_j$之項為零，因向量$\mathbf{t}$垂直於向量$\mathbf{h}$。用類似做法於式(21.139)可得式(21.203)之$\mathbf{A}_+^{-1} - \mathbf{A_c^{-1}}$，稱為$G$（Greenstadt）修訂式。

$$\begin{aligned}
\left\{ \mathbf{r} \right\} &= \left[\mathbf{r} \quad \mathbf{s} \right] \begin{bmatrix} \alpha & \gamma \\ \gamma & \beta \end{bmatrix} \begin{bmatrix} \mathbf{r^T} \\ \mathbf{s^T} \end{bmatrix} \left\{ \mathbf{s} \right\} && (21.187)\\
&= \left[\mathbf{r} \quad \mathbf{s} \right] \begin{bmatrix} \alpha & \gamma \\ \gamma & \beta \end{bmatrix} \begin{Bmatrix} \tau\sigma\cos\phi \\ \sigma^2 \end{Bmatrix} \\
&= \left[\mathbf{r} \quad \mathbf{s} \right] \begin{Bmatrix} \alpha\tau\sigma\cos\phi + \gamma\sigma^2 \\ \gamma\tau\sigma\cos\phi + \beta\sigma^2 \end{Bmatrix} && (21.188)
\end{aligned}$$

$$\alpha\tau\sigma\cos\phi + \gamma\sigma^2 = 1\,, \qquad \gamma\tau\sigma\cos\phi + \beta\sigma^2 = 0 \tag{21.189}$$

$$\alpha \;=\; \frac{1}{\tau\sigma\cos\phi} - \frac{\sigma}{\tau\cos\phi}\,\gamma\;, \qquad \beta \;=\; -\frac{\tau\cos\phi}{\sigma}\,\gamma \tag{21.190}$$

$$\mathbf{A_+ - A_c} \;=\; \left[\begin{array}{cc} \mathbf{r} & \mathbf{s} \end{array}\right] \left[\begin{array}{cc} \alpha & \gamma \\ \gamma & \beta \end{array}\right] \left[\begin{array}{c} \mathbf{r^T} \\ \mathbf{s^T} \end{array}\right] \tag{21.191}$$

$$= \; (\frac{1}{\tau\sigma\cos\phi} - \frac{\sigma}{\tau\cos\phi}\,\gamma)\mathbf{r}\mathbf{r^T} - (\frac{\tau\cos\phi}{\sigma}\,\gamma)\mathbf{s}\mathbf{s^T} + \gamma(\mathbf{r}\mathbf{s^T} + \mathbf{s}\mathbf{r^T})$$

$$= \; \frac{\tau\cos\phi}{\sigma}(\frac{\mathbf{r}\mathbf{r^T}}{\tau^2\cos^2\phi} - \gamma\sigma^2(\frac{\mathbf{r}\mathbf{r^T}}{\tau^2\cos^2\phi} + \frac{\mathbf{s}\mathbf{s^T}}{\sigma^2} - \frac{\mathbf{r}\mathbf{s^T} + \mathbf{s}\mathbf{r^T}}{\tau\sigma\cos\phi})) \tag{21.192}$$

$$= \; \frac{\tau\cos\phi}{\sigma}((\mathbf{h}+\mathbf{t})(\mathbf{h}+\mathbf{t})^\mathbf{T}$$
$$-\gamma\sigma^2((\mathbf{h}+\mathbf{t})(\mathbf{h}+\mathbf{t})^\mathbf{T} + \mathbf{h}\mathbf{h^T} - (\mathbf{h}+\mathbf{t})\mathbf{h^T} - \mathbf{h}(\mathbf{h}+\mathbf{t})^\mathbf{T}) \tag{21.193}$$

$$= \; \frac{\tau\cos\phi}{\sigma}((\mathbf{h}+\mathbf{t})(\mathbf{h}+\mathbf{t})^\mathbf{T} - \gamma\sigma^\mathbf{2}(\mathbf{t}\mathbf{t^T}))$$

$$= \; \frac{\tau\cos\phi}{\sigma}(\mathbf{h}\mathbf{h^T} + \mathbf{h}\mathbf{t^T} + \mathbf{t}\mathbf{h^T} + (\mathbf{1} - \gamma\sigma^\mathbf{2})\,\mathbf{t}\mathbf{t^T}) \tag{21.194}$$

$$||\mathbf{A_+ - A_c}||_\mathbf{F}^\mathbf{2}$$
$$= \; (\frac{\tau\cos\phi}{\sigma})^2 \sum_i \sum_j (h_i h_j + h_i t_j + t_i h_j + (1-\gamma\sigma^2)\,t_i t_j)^2 \tag{21.195}$$

$$-\frac{d\,(||\mathbf{A_+ - A_c}||_\mathbf{F}^\mathbf{2})}{2\tau^2\cos^2\phi\;d\,\gamma}$$
$$= \; \sum_i \sum_j (h_i h_j + h_i t_j + t_i h_j + (1-\gamma\sigma^2)\,t_i t_j)\,t_i t_j \tag{21.196}$$

$$= \; \sum_i \sum_j (t_i h_i h_j t_j + t_i h_i t_j t_j + t_i t_i h_j t_j + (1-\gamma\sigma^2)\,t_i^2 t_j^2) \tag{21.197}$$

$$= \; (1-\gamma\sigma^2)\sum_i t_i^2 \sum_j t_j^2 = 0 \tag{21.198}$$

$$\gamma = \frac{1}{\sigma^2} = \frac{1}{\mathbf{s^T s}}\;, \qquad \beta = -\frac{\tau\sigma\cos\phi}{\sigma^4} = -\frac{\mathbf{r^T s}}{(\mathbf{s^T s})^\mathbf{2}}\;, \qquad \alpha = 0 \tag{21.199}$$

$$\mathbf{A_+ - A_c} \;=\; \left[\begin{array}{cc} \mathbf{r} & \mathbf{s} \end{array}\right] \left[\begin{array}{cc} 0 & \mathbf{1/s^T s} \\ \mathbf{1/s^T s} & -\mathbf{r^T s}/(\mathbf{s^T s})^\mathbf{2} \end{array}\right] \left[\begin{array}{c} \mathbf{r^T} \\ \mathbf{s^T} \end{array}\right] \tag{21.200}$$

$$= \; \frac{\mathbf{r}\mathbf{s^T} + \mathbf{s}\mathbf{r^T}}{\mathbf{s^T s}} - \frac{(\mathbf{r^T s})\mathbf{s}\mathbf{s^T}}{(\mathbf{s^T s})^\mathbf{2}} \tag{21.201}$$

$$\mathbf{A_+^{-1} - A_c^{-1}} \;=\; \left[\begin{array}{cc} \bar{\mathbf{r}} & \mathbf{y} \end{array}\right] \left[\begin{array}{cc} 0 & \mathbf{1/y^T y} \\ \mathbf{1/y^T y} & -\bar{\mathbf{r}}^\mathbf{T}\mathbf{y}/(\mathbf{y^T y})^\mathbf{2} \end{array}\right] \left[\begin{array}{c} \bar{\mathbf{r}}^\mathbf{T} \\ \mathbf{y^T} \end{array}\right] \tag{21.202}$$

$$= \frac{\bar{\mathbf{r}}\mathbf{y}^{\mathbf{T}} + \mathbf{y}\bar{\mathbf{r}}^{\mathbf{T}}}{\mathbf{y}^{\mathbf{T}}\mathbf{y}} - \frac{(\bar{\mathbf{r}}^{\mathbf{T}}\mathbf{y})\mathbf{y}\mathbf{y}^{\mathbf{T}}}{(\mathbf{y}^{\mathbf{T}}\mathbf{y})^{\mathbf{2}}} \tag{21.203}$$

用類似做法於式(21.146)及式(21.147)可最小化$||\mathbf{T}^{-\mathbf{T}}(\mathbf{A}_+ - \mathbf{A_c})\mathbf{T}^{-\mathbf{1}}||_{\mathbf{F}}$或$||\mathbf{T}(\mathbf{A}_+^{-\mathbf{1}} - \mathbf{A_c^{-1}})\mathbf{T}^{\mathbf{T}}||_{\mathbf{F}}$。而所得結果分別為$DFP$修訂式與$BFGS$修訂式。其逆矩陣等於由(3b)條件所得之式(21.186)與式(21.185)之矩陣。注意上列二修訂式會受比例調整之影響；而下列二修訂式則不受影響。

$$\begin{aligned}
\hat{\mathbf{A}}_+ - \hat{\mathbf{A}}_{\mathbf{c}} &= \frac{\hat{\mathbf{r}}\hat{\mathbf{s}}^{\mathbf{T}} + \hat{\mathbf{s}}\hat{\mathbf{r}}^{\mathbf{T}}}{\hat{\mathbf{s}}^{\mathbf{T}}\hat{\mathbf{s}}} - \frac{(\hat{\mathbf{r}}^{\mathbf{T}}\hat{\mathbf{s}})\hat{\mathbf{s}}\hat{\mathbf{s}}^{\mathbf{T}}}{(\hat{\mathbf{s}}^{\mathbf{T}}\hat{\mathbf{s}})^{\mathbf{2}}} \\
\mathbf{A}_+ - \mathbf{A}_{\mathbf{c}} &= \mathbf{T}^{\mathbf{T}}\frac{\mathbf{T}^{-\mathbf{T}}\mathbf{r}\mathbf{s}^{\mathbf{T}}\mathbf{T}^{\mathbf{T}} + \mathbf{T}\mathbf{s}\mathbf{r}^{\mathbf{T}}\mathbf{T}^{-\mathbf{1}}}{\mathbf{s}^{\mathbf{T}}\mathbf{T}^{\mathbf{T}}\mathbf{T}\mathbf{s}}\mathbf{T} - \mathbf{T}^{\mathbf{T}}\frac{(\mathbf{r}^{\mathbf{T}}\mathbf{T}^{-\mathbf{1}}\mathbf{T}\mathbf{s})\mathbf{T}\mathbf{s}\mathbf{s}^{\mathbf{T}}\mathbf{T}^{\mathbf{T}}}{(\mathbf{s}^{\mathbf{T}}\mathbf{T}^{\mathbf{T}}\mathbf{T}\mathbf{s})^{\mathbf{2}}}\mathbf{T} \\
&= \frac{\mathbf{r}\mathbf{s}^{\mathbf{T}}\mathbf{A}_+ + \mathbf{A}_+\mathbf{s}\mathbf{r}^{\mathbf{T}}}{\mathbf{s}^{\mathbf{T}}\mathbf{A}_+\mathbf{s}} - \frac{(\mathbf{r}^{\mathbf{T}}\mathbf{s})\mathbf{A}_+\mathbf{s}\mathbf{s}^{\mathbf{T}}\mathbf{A}_+}{(\mathbf{s}^{\mathbf{T}}\mathbf{A}_+\mathbf{s})^{\mathbf{2}}} \\
&= \frac{\mathbf{r}\mathbf{y}^{\mathbf{T}} + \mathbf{y}\mathbf{r}^{\mathbf{T}}}{\mathbf{y}^{\mathbf{T}}\mathbf{s}} - \frac{(\mathbf{r}^{\mathbf{T}}\mathbf{s})\mathbf{y}\mathbf{y}^{\mathbf{T}}}{(\mathbf{y}^{\mathbf{T}}\mathbf{s})^{\mathbf{2}}}
\end{aligned} \tag{21.204}$$

$$\begin{aligned}
\hat{\mathbf{A}}_+^{-\mathbf{1}} - \hat{\mathbf{A}}_{\mathbf{c}}^{-\mathbf{1}} &= \frac{\hat{\bar{\mathbf{r}}}\hat{\mathbf{y}}^{\mathbf{T}} + \hat{\mathbf{y}}\hat{\bar{\mathbf{r}}}^{\mathbf{T}}}{\hat{\mathbf{y}}^{\mathbf{T}}\hat{\mathbf{y}}} - \frac{(\hat{\bar{\mathbf{r}}}^{\mathbf{T}}\hat{\mathbf{y}})\hat{\mathbf{y}}\hat{\mathbf{y}}^{\mathbf{T}}}{(\hat{\mathbf{y}}^{\mathbf{T}}\hat{\mathbf{y}})^{\mathbf{2}}} \\
\mathbf{A}_+^{-\mathbf{1}} - \mathbf{A}_{\mathbf{c}}^{-\mathbf{1}} &= \mathbf{T}^{-\mathbf{1}}\frac{\mathbf{T}\bar{\mathbf{r}}\mathbf{y}^{\mathbf{T}}\mathbf{T}^{-\mathbf{1}} + \mathbf{T}^{-\mathbf{T}}\mathbf{y}\bar{\mathbf{r}}^{\mathbf{T}}\mathbf{T}^{\mathbf{T}}}{\mathbf{y}^{\mathbf{T}}\mathbf{T}^{-\mathbf{1}}\mathbf{T}^{-\mathbf{T}}\mathbf{y}}\mathbf{T}^{-\mathbf{T}} \\
&\qquad -\mathbf{T}^{-\mathbf{1}}\frac{(\bar{\mathbf{r}}^{\mathbf{T}}\mathbf{T}^{\mathbf{T}}\mathbf{T}^{-\mathbf{T}}\mathbf{y})\mathbf{T}^{-\mathbf{T}}\mathbf{y}\mathbf{y}^{\mathbf{T}}\mathbf{T}^{-\mathbf{1}}}{(\mathbf{y}^{\mathbf{T}}\mathbf{T}^{-\mathbf{1}}\mathbf{T}^{-\mathbf{T}}\mathbf{y})^{\mathbf{2}}}\mathbf{T}^{-\mathbf{T}} \\
&= \frac{\bar{\mathbf{r}}\mathbf{y}^{\mathbf{T}}\mathbf{A}_+^{-\mathbf{1}} + \mathbf{A}_+^{-\mathbf{1}}\mathbf{y}\bar{\mathbf{r}}^{\mathbf{T}}}{\mathbf{y}^{\mathbf{T}}\mathbf{A}_+^{-\mathbf{1}}\mathbf{y}} - \frac{(\bar{\mathbf{r}}^{\mathbf{T}}\mathbf{y})\mathbf{A}_+^{-\mathbf{1}}\mathbf{y}\mathbf{y}^{\mathbf{T}}\mathbf{A}_+^{-\mathbf{1}}}{(\mathbf{y}^{\mathbf{T}}\mathbf{A}_+^{-\mathbf{1}}\mathbf{y})^{\mathbf{2}}} \\
&= \frac{\bar{\mathbf{r}}\mathbf{s}^{\mathbf{T}} + \mathbf{s}\bar{\mathbf{r}}^{\mathbf{T}}}{\mathbf{s}^{\mathbf{T}}\mathbf{y}} - \frac{(\bar{\mathbf{r}}^{\mathbf{T}}\mathbf{y})\mathbf{s}\mathbf{s}^{\mathbf{T}}}{(\mathbf{s}^{\mathbf{T}}\mathbf{y})^{\mathbf{2}}}
\end{aligned} \tag{21.205}$$

利用下列之Sherman–Morrison–Woodbury公式可証明上列之矩陣分別為式(21.186)與式(21.185)者之逆矩陣。下式中$\mathbf{S}$為$m \times m$之方陣，而$m \leq n$。

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^{\mathbf{T}})^{-\mathbf{1}} = \mathbf{A}^{-\mathbf{1}} - \frac{\mathbf{A}^{-\mathbf{1}}\mathbf{u}\mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}}{\mathbf{1} + \mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}\mathbf{u}} \tag{21.206}$$

$$(\mathbf{A} + \mathbf{U}\mathbf{S}\mathbf{V}^{\mathbf{T}})^{-\mathbf{1}} = \mathbf{A}^{-\mathbf{1}} - \mathbf{A}^{-\mathbf{1}}\mathbf{U}(\mathbf{S}^{-\mathbf{1}} + \mathbf{V}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}\mathbf{U})^{-\mathbf{1}}\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}} \tag{21.207}$$

上列二公式可由下列之乘積為單位矩陣証明之：

$$\begin{aligned}
&(\mathbf{A} + \mathbf{u}\mathbf{v}^{\mathbf{T}})(\mathbf{A}^{-\mathbf{1}} - \frac{\mathbf{A}^{-\mathbf{1}}\mathbf{u}\mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}}{\mathbf{1} + \mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}\mathbf{u}}) \\
&= \mathbf{I} + \mathbf{u}\mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}} - \frac{(\mathbf{A}\mathbf{A}^{-\mathbf{1}})\mathbf{u}\mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}} + \mathbf{u}(\mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}\mathbf{u})\mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}}{\mathbf{1} + \mathbf{v}^{\mathbf{T}}\mathbf{A}^{-\mathbf{1}}\mathbf{u}} = \mathbf{I}
\end{aligned}$$

$$
\begin{aligned}
&(\mathbf{A}+\mathbf{USV}^{\mathbf{T}})(\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{U}(\mathbf{S}^{-1}+\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1})\\
&= \mathbf{I} \quad + \quad \mathbf{USV}^{\mathbf{T}}\mathbf{A}^{-1}\\
&\quad -(\mathbf{AA}^{-1})\mathbf{US}(\mathbf{S}^{-1})(\mathbf{S}^{-1}+\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\\
&\qquad -\mathbf{US}(\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})(\mathbf{S}^{-1}+\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1} \;=\; \mathbf{I}
\end{aligned}
$$

以下列之$BFGS$修訂式(21.185)為例，利用上列公式求其對應之逆矩陣。

$$
\mathbf{A}_+-\mathbf{A}_\mathbf{c}=\frac{\mathbf{y_c}\mathbf{y_c^T}}{\mathbf{y_c^T}\mathbf{s_c}}-\frac{\mathbf{A_c}\mathbf{s_c}\mathbf{s_c^T}\mathbf{A_c}}{\mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c}}=\begin{bmatrix}\mathbf{y_c} & \mathbf{A_c}\mathbf{s_c}\end{bmatrix}\begin{bmatrix}\frac{1}{\mathbf{y_c^T}\mathbf{s_c}} & 0\\ 0 & \frac{-1}{\mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c}}\end{bmatrix}\begin{bmatrix}\mathbf{y_c^T}\\ \mathbf{s_c^T}\mathbf{A_c}\end{bmatrix}
$$

$$
\begin{aligned}
&-(\mathbf{S}^{-1}+\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})^{-1}\\
&\qquad=\left(\begin{bmatrix}-\mathbf{y_c^T}\mathbf{s_c} & 0\\ \mathbf{0} & \mathbf{s_c^T}\mathbf{A_c}\mathbf{s_c}\end{bmatrix}-\begin{bmatrix}\mathbf{y_c^T}\\ \mathbf{s_c^T}\mathbf{A_c}\end{bmatrix}\mathbf{A_c^{-1}}\begin{bmatrix}\mathbf{y_c} & \mathbf{A_c}\mathbf{s_c}\end{bmatrix}\right)^{-1}\\
&\qquad=\begin{bmatrix}-\mathbf{y_c^T}(\mathbf{s_c}+\mathbf{A_c^{-1}}\mathbf{y_c}) & -\mathbf{y_c^T}\mathbf{s_c}\\ -\mathbf{s_c^T}\mathbf{y_c} & 0\end{bmatrix}^{-1}\\
&\qquad=\begin{bmatrix}0 & -\mathbf{1}/\mathbf{y_c^T}\mathbf{s_c}\\ -\mathbf{1}/\mathbf{s_c^T}\mathbf{y_c} & \mathbf{y_c^T}(\mathbf{s_c}+\mathbf{A_c^{-1}}\mathbf{y_c})/(\mathbf{y_c^T}\mathbf{s_c})^{\mathbf{2}}\end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{A}_+^{-1}-\mathbf{A}_\mathbf{c}^{-1} &= -\mathbf{A}^{-1}\mathbf{U}(\mathbf{S}^{-1}+\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\\
&=\begin{bmatrix}\mathbf{A_c^{-1}}\mathbf{y_c} & s_c\end{bmatrix}\begin{bmatrix}-1 & 0\\ 1 & 1\end{bmatrix}\\
&\quad\begin{bmatrix}-1 & 0\\ 1 & 1\end{bmatrix}\begin{bmatrix}0 & -\mathbf{1}/\mathbf{y_c^T}\mathbf{s_c}\\ -\mathbf{1}/\mathbf{s_c^T}\mathbf{y_c} & \mathbf{y_c^T}(\mathbf{s_c}+\mathbf{A_c^{-1}}\mathbf{y_c})/(\mathbf{y_c^T}\mathbf{s_c})^{\mathbf{2}}\end{bmatrix}\begin{bmatrix}-1 & 0\\ 1 & 1\end{bmatrix}\\
&\quad\begin{bmatrix}-1 & 0\\ 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{y_c^T}\mathbf{A_c^{-1}}\\ \mathbf{s_c^T}\end{bmatrix}\\
&=\begin{bmatrix}\bar{\mathbf{r}}_\mathbf{c} & \mathbf{s_c}\end{bmatrix}\begin{bmatrix}0 & \mathbf{1}/\mathbf{y_c^T}\mathbf{s_c}\\ \mathbf{1}/\mathbf{s_c^T}\mathbf{y_c} & -\mathbf{y_c^T}(\mathbf{s_c}-\mathbf{A_c^{-1}}\mathbf{y_c})/(\mathbf{y_c^T}\mathbf{s_c})^{\mathbf{2}}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{r}}_\mathbf{c}^{\mathbf{T}}\\ \mathbf{s_c^T}\end{bmatrix}\\
&=\frac{\bar{\mathbf{r}}_\mathbf{c}\mathbf{s_c^T}+\mathbf{s_c}\bar{\mathbf{r}}_\mathbf{c}^{\mathbf{T}}}{\mathbf{y_c^T}\mathbf{s_c}}-\frac{(\mathbf{y_c^T}\bar{\mathbf{r}}_\mathbf{c})\mathbf{s_c}\mathbf{s_c^T}}{(\mathbf{y_c^T}\mathbf{s_c})^{\mathbf{2}}} \qquad (21.205)
\end{aligned}
$$

習題

1. 試用Sherman–Morrison–Woodbury公式由式(21.166)導出式(21.167)。

2. 試由下列$\mathbf{A}_+$之*DFP*修訂式利用Sherman–Morrison–Woodbury公式推導下列$\mathbf{A}_+^{-1}$之*DFP*修訂式。

$$\mathbf{A}_+ = \mathbf{A_c} + \frac{\mathbf{r_c y_c^T} + \mathbf{y_c r_c^T}}{\mathbf{y_c^T s_c}} - \frac{(\mathbf{r_c^T s_c})\mathbf{y_c y_c^T}}{(\mathbf{y_c^T s_c})^2} \tag{21.204}$$

$$\mathbf{A}_+^{-1} = \mathbf{A_c^{-1}} + \frac{\mathbf{s_c s_c^T}}{\mathbf{s_c^T y_c}} - \frac{\mathbf{A_c^{-1} y_c y_c^T A_c^{-1}}}{\mathbf{y_c^T A_c^{-1} y_c}} \tag{21.186}$$

3. 試寫一套程式，利用拉格蘭治乘數法以極大化極小值 (Max.min) 問題解有限制條件之最佳化問題。

參考文獻

1. Fletcher, R. and Reeves, C. M., "Function Minimizatiom by Conjugate Gradients," Computer J., Vol.7, 1964, pp.149-154.

2. Fletcher, R. and Powell, M. J. D., "A Rapidly Convergent Decent Method for Minimization," Computer J., Vol.6, 1963, pp.163-168.

3. Zangwill, Willard I., *Nonlinear Programming - A Unified Approach,* Englewood Cliffs N.J., Prentice-Hall, 1969.

4. Bazaraa, Mokhtar S. and Shetty, C. M., *Nonlinear Programming - Theory and Applications,* New York, Wiley, 1979.

5. Rao, S. S., *Optimization : Theory and Applications,* 2nd Ed., New Delhi, Wiley Eastern, 1984.

6. Dennis, J.E., Jr. and Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice-Hall Inc., Englewood Chiffs, 1983.

7. Lin, Tsung-Wu, "Double Secant Method for Nonlinear Mathematical Programming," Proceedings of the 2nd IFIP WG7.5 Conference on Reliability and Optimization of Structural Systems, 163-171, London, U.K., 1988.

8. 林聰悟，"Well Behaved Penalty Functions for Constrained Optimization," Journal of the Chinese Instiitute of Engineers, Vol.13, No.2, 157-166, Taipei, 1990.

9. 林聰悟，鐘銘輝，黃金田，"Solving Optimization Problems by Searching Along a Curve, Journal of the Chinese Institute of Engineers, Vol.15, No.4, 427-434, Taipei, 1992.

10. 林聰悟，黃金田，鐘銘輝，"Adaptive Modificaiton of Objective Function for Lagrange Multiplier Method in Constrained Optimization Problems," Journal of the Chinese Institute of Engineers, Vol.16, No.2, Taipei, 1992.

11. 鐘銘輝，『以懲罰函數法及機動調整拉格蘭治法求解最佳化問題之方法研究與程式建立』，國立臺灣大學土木工程研究所博士論文，1993.

第二十二章

結構系統參數識別

22.1 前言

系統之反應分析為計算一已知系統(即描述該系統之參數已知)在受某一輸入函數作用時之輸出函數。 系統之識別分析則試著由系統之輸入函數與輸出函數反求(即識別出)該系統之參數。任何工程系統只要其輸入與輸出關係依照某種可預測之組成律，即可用系統識別法求出該系統之參數。在地震工程領域，系統識別法應用於實際結構以研究結構受強烈地表震動之地震反應與動力特性。根據所量測得之地表震動為輸入函數，與所量測得之結構反應為輸出函數，其結構系統之數學模式之特性參數可由動力反應分析識別出。

本章擬介紹Beck之模態最小化法 (Modal minimization method)。該法利用最小化輸出誤差函數以估計線性結構動力系統之最佳參數。其輸出誤差函數為數學模式分析得之輸出函數與量測得之輸出函數之差之平方和。該誤差函數可代表輸出函數之吻合度。結構系統之數學模式根據結構動力系統之運動方程式，並經模態分離使每一模態為獨立之運動方程式以簡化分析步驟及識別過程。輸入函數採用地表震動 (excitation)之加速度、速度與位移歷時 (time history) ；輸出函數採用結構量測點之結構反應 (response)之加速度歷時。識別之系統參數主要包括各模態之自然頻率 (natural frequency) 、阻尼比 (damping ratio)與結構之模態向量， 其他附帶需識別者為各模態之參與係數與初始位移及速度。

求系統參數使誤差函數為最小與曲線近似法之原理類似，但因誤差函數對參數之微分等於零之方程式並非為線性方程式，故須以一般函數之極值問題求解。

22.2 系統參數識別法

對一工程系統，一般可做三種不同型態之分析：
(1) 預測分析：由已知之系統參數與輸入函數計算系統之輸出函數。
(2) 量測分析：由已知之系統參數與輸出函數計算系統之輸入函數。
(3) 識別分析：由已知之系統輸入與輸出函數計算系統之行為參數。
因此識別分析可提供工程師有關系統之行為參數與其基本特性之重要資料。

系統識別分析之主要目的為求得系統參數$\{\beta\}$以將系統模式化。利用已知之系統輸入函數$\{\ddot{u}_g(t)\}$為模式輸入函數，其計算所得之模式輸出函數$\{\ddot{x}(t)\}$必須與量測所得之系統輸出函數$\{\ddot{u}(t)\}$相符合。因此，將代表輸出函數吻合度之誤差函數最小化，可求得系統參數。其誤差函數

$$J(\{\beta\}) = \int ||\{\ddot{u}(t)\} - \{\ddot{x}(t;\{\beta\})\}||^2 \, dt \tag{22.1}$$

為計算所得之模式輸出函數與量測所得之系統輸出函數之差之平方對時間之積分。系統識別分析之主要結果為代表系統行為之數學模式之最佳系統參數。

22.3 模態最小化法

模態最小化法基本上只針對單一模態反應之誤差函數最小化以求得該模態之參數。對於多自由度線性結構動力系統，假設其阻尼可使模態分離，即可分別計算每一模態之反應，而不受其他模態之影響。如此結構之總反應可表示成各別模態之反應之線性組合。

在模態最小化法中，初步假設量測所得之總反應全部為第一模態之反應。將第一模態反應之誤差函數最小化，可得第一模態之模態參數之最佳值，同時可得第一模態之反應。其次假設量測所得之總反應為第一模態及第二模態之反應之和。因此量測所得之總反應減去計算所得之第一模態之反應即為第二模態之反應。將第二模態反應之誤差函數最小化，可得第二模態之模態參數之最佳值，同時可得第二模態之反應。依此類推，即可求得較高模態之參數及反應，直至高模態之反應已小至可以忽略之程度為止。此外，若高模態之反應與量測之誤差相當時亦無法求

得可靠之參數。一般情況計算四到六個模態即可得到相當吻合之輸出反應。

注意在第一次循環求得各模態之參數後，應再由第一模態開始重新將第一模態反應之誤差函數最小化。不過此時第一模態之反應為總反應減其他模態之反應，因其較最初之假設正確，故可得較正確之模態參數，其他模態亦同樣可得較正確之模態參數。理論上，應反覆計算數個循環直至各模態之參數不再改變或改變量很小為止。通常約需做五到八個循環。

22.4 結構動力分析

為了做系統識別分析，本節首先簡要說明多自由度結構系統之運動方程式及有關之分析方法。考慮一多自由度結構動力系統，為了能分析多支承之輸入函數，將結構自由度分為二部分：一部分為支承自由度，以下標 g 標示；其餘部分為結構自由度。由結點之動力平衡，考慮慣性力、阻尼力與彈性力，可得下列之運動方程式

$$[m]\{\ddot{u}^t(t)\}+[m_g]\{\ddot{u}_g(t)\}+[c]\{\dot{u}^t(t)\}+[c_g]\{\dot{u}_g(t)\}+[k]\{u^t(t)\}+[k_g]\{u_g(t)\}=\{0\} \tag{22.2}$$

與下列之初始條件

$$\{u^t(t_o)\} = \{u_o^t\}\,, \qquad \{\dot{u}^t(t_o)\} = \{\dot{u}_o^t\} \tag{22.3}$$

式中 $[m],\ [m_g]$ 為質量矩陣，$[c],\ [c_g]$ 為阻尼矩陣，$[k],\ [k_g]$ 為勁度矩陣；$\{\ddot{u}_g(t)\}, \{\dot{u}_g(t)\}, \{u_g(t)\}$ 為已知 N_g 個支承自由度之加速度、速度與位移；$\{\ddot{u}^t(t)\}, \{\dot{u}^t(t)\}, \{u^t(t)\}$ 為 N 個結構自由度之總加速度、總速度與總位移；而 $\{\dot{u}_o^t\}, \{u_o^t\}$ 為結構自由度之初始總速度與初始總位移。將結構自由度之總位移 $\{u^t(t)\}$ 視為動態位移 $\{u(t)\}$ 與由支承位移 $\{u_g(t)\}$ 之靜態作用所產生之擬靜態位移 $\{u^s(t)\}$ 之和，即

$$\{u^t(t)\} = \{u(t)\} + \{u^s(t)\} = \{u(t)\} + [i]\{u_g(t)\} \tag{22.4}$$

但因在系統識別分析時，其中之影響矩陣 $[i]$ 為未知參數，且其元素數量頗多，識別不易，本章將改用已知之擬影響矩陣 $[\hat{i}]\}$ 取代之。因控制系統識別分析之誤差函數 J 係根據輸出函數之差值計算，由下式可知其以三種不同輸出函數 $\{\ddot{u}^t(t)\}$ 、$\{\ddot{u}(t)\}$ 或 $\{\hat{\ddot{u}}(t)\}$ 計算時，所得之結果相同。因此不論

用未知影響矩陣或已知擬影響矩陣計算，所得之誤差函數J之值並不受影響。

$$\begin{aligned}
J &= \int ||\{\ddot{u}_n(t)\}-\{\ddot{x}_n(t)\}||^2\,dt = \int ||(\{\ddot{u}^t_n(t)\}-\{\ddot{u}^s_n(t)\})-(\{\ddot{x}^t_n(t)\}-\{\ddot{u}^s_n(t)\})||^2\,dt \\
&= \int ||(\{\ddot{u}^t_n(t)\}-[i]\{\ddot{u}_g(t)\})-(\{\ddot{x}^t_n(t)\}-[i]\{\ddot{u}_g(t)\})||^2\,dt \\
&= \int ||\{\ddot{u}^t_n(t)\}-\{\ddot{x}^t_n(t)\}||^2\,dt \\
&= \int ||(\{\hat{\ddot{u}}_n(t)\}+[\hat{i}]\{\ddot{u}_g(t)\})-(\{\hat{\ddot{x}}_n(t)\}+[\hat{i}]\{\ddot{u}_g(t)\})||^2\,dt \\
&= \int ||\{\hat{\ddot{u}}_n(t)\}-\{\hat{\ddot{x}}_n(t)\}||^2\,dt
\end{aligned} \tag{22.5}$$

現將結構總位移以已知擬影響矩陣表示成

$$\{u^t(t)\} = \{\hat{u}(t)\} + \{\hat{u}^s(t)\} = \{\hat{u}(t)\} + [\hat{i}]\{u_g(t)\} \tag{22.6}$$

將上式代入式(22.2)，即得

$$\begin{aligned}
[m]\{\hat{\ddot{u}}(t)\}+[c]\{\hat{\dot{u}}(t)\}+[k]\{\hat{u}(t)\} = &-[[m][\hat{i}] + [m_g]]\{\ddot{u}_g(t)\}-[[c][\hat{i}] + [c_g]]\{\dot{u}_g(t)\} \\
&-[[k][\hat{i}] + [k_g]]\{u_g(t)\}
\end{aligned} \tag{22.7}$$

或簡寫成

$$[m]\{\hat{\ddot{u}}(t)\}+[c]\{\hat{\dot{u}}(t)\}+[k]\{\hat{u}(t)\} = -[\hat{m}_g]\{\ddot{u}_g(t)\}-[\hat{c}_g]\{\dot{u}_g(t)\}-[\hat{k}_g]\{u_g(t)\} \tag{22.8}$$

現將擬動態位移$\{\hat{u}(t)\}$寫成各模態位移之線性組合

$$\{\hat{u}(t)\} = \{\phi_1\}Y_1(t) + \cdots + \{\phi_N\}Y_N(t) = \sum_{n=1}^{N}\{\phi_n\}Y_n(t) = \sum_{n=1}^{N}\{\hat{u}_n(t)\} \tag{22.9}$$

式中$\{\phi_n\}$為第n模態之$N\times 1$模態向量，$Y_n(t)$為第n模態之廣義正規化坐標，$\{\hat{u}_n(t)\}$為第n模態之擬模態位移。將式(22.9)代入式(22.8)後，前乘$\{\phi_n\}^T$，並引用模態向量之正交性（即當$i\neq j$時：$\{\phi_i\}^T[m]\{\phi_j\}=0$，$\{\phi_i\}^T[c]\{\phi_j\}=0$，$\{\phi_i\}^T[k]\{\phi_j\}=0$），即得下式之第$n$模態之獨立模態方程式，可用以求得擬模態位移。

$$\begin{aligned}
\{\hat{\ddot{u}}_n(t)\} + 2\xi_n\omega_n\{\hat{\dot{u}}_n(t)\} + \omega_n^2\{\hat{u}_n(t)\} = -\{\phi_n\}\Big(&\langle L_{nm}\rangle\{\ddot{u}_g(t)\} + \langle L_{nc}\rangle\{\dot{u}_g(t)\} \\
&+\langle L_{nk}\rangle\{u_g(t)\}\Big)
\end{aligned} \tag{22.10}$$

及初始擬模態位移與速度

$$\{\hat{u}_n(t_o)\} = \hat{u}_{no} = \{\phi_n\}Y_{no}\,, \qquad \{\hat{\dot{u}}_n(t_o)\} = \hat{\dot{u}}_{no} = \{\phi_n\}\dot{Y}_{no} \tag{22.11}$$

式中之第n模態之自然頻率ω_n與阻尼比ξ_n之定義為

$$\omega_n^2 = \frac{\{\phi_n\}^T[k]\{\phi_n\}}{\{\phi_n\}^T[m]\{\phi_n\}} \tag{22.12}$$

$$2\xi_n\omega_n = \frac{\{\phi_n\}^T[c]\{\phi_n\}}{\{\phi_n\}^T[m]\{\phi_n\}} \tag{22.13}$$

式中之支承加速度、速度與位移對第n模態之參與係數$\langle L_{nm}\rangle, \langle L_{nc}\rangle, \langle L_{nk}\rangle$之定義為

$$\begin{aligned}
\langle L_{nm}\rangle &= \langle L_{nm1}, \ldots, L_{nmN_g}\rangle = \frac{\{\phi_n\}^T[\hat{m}_g]}{\{\phi_n\}^T[m]\{\phi_n\}} = \frac{\{\phi_n\}^T([m][\hat{i}]+[m_g])}{\{\phi_n\}^T[m]\{\phi_n\}} \\
\langle L_{nc}\rangle &= \langle L_{nc1}, \ldots, L_{ncN_g}\rangle = \frac{\{\phi_n\}^T[\hat{c}_g]}{\{\phi_n\}^T[m]\{\phi_n\}} = \frac{\{\phi_n\}^T([c][\hat{i}]+[c_g])}{\{\phi_n\}^T[m]\{\phi_n\}} \\
\langle L_{nk}\rangle &= \langle L_{nk1}, \ldots, L_{nkN_g}\rangle = \frac{\{\phi_n\}^T[\hat{k}_g]}{\{\phi_n\}^T[m]\{\phi_n\}} = \frac{\{\phi_n\}^T([k][\hat{i}]+[k_g])}{\{\phi_n\}^T[m]\{\phi_n\}}
\end{aligned} \tag{22.14}$$

注意此處因採用擬影響矩陣而比一般動力分析多了兩種(速度與位移)參與係數。而初始模態位移與初始模態速度之廣義正規化坐標可由$\{\hat{u}_o\}$與$\{\hat{\dot{u}}_o\}$求得

$$Y_n(t_o) = Y_{no} = \frac{\{\phi_n\}^T[m]\{\hat{u}_o\}}{\{\phi_n\}^T[m]\{\phi_n\}}, \quad \dot{Y}_n(t_o) = \dot{Y}_{no} = \frac{\{\phi_n\}^T[m]\{\hat{\dot{u}}_o\}}{\{\phi_n\}^T[m]\{\phi_n\}} \tag{22.15}$$

22.5 單一模態之識別

現以模態最小化法，利用分離之獨立模態方程式，可一次分析一個模態之反應，而將一個模態之參數識別出。因此結構系統之參數識別即可簡化為單一模態之參數識別。地震工程上之結構動力系統可用下列數學模式表示

$$\begin{aligned}
\{\hat{\ddot{x}}_n(t)\} + 2\xi_n\omega_n\{\hat{\dot{x}}_n(t)\} + \omega_n^2\{\hat{x}_n(t)\} = -\{\phi_n\}(&\langle L_{nm}\rangle\{\ddot{u}_g(t)\} + \langle L_{nc}\rangle\{\dot{u}_g(t)\} \\
&+\langle L_{nk}\rangle\{u_g(t)\})
\end{aligned} \tag{22.16}$$

及初始擬模態位移與速度

$$\{\hat{x}_n(t_o)\} = \hat{x}_{no} = \{\phi_n\}Y_{no}, \qquad \{\hat{\dot{x}}_n(t_o)\} = \hat{\dot{x}}_{no} = \{\phi_n\}\dot{Y}_{no} \tag{22.17}$$

式中之輸入函數為量測所得之地表加速度$\{\ddot{u}_g(t)\}$、速度$\{\dot{u}_g(t)\}$與位移$\{u_g(t)\}$；其輸出函數為計算所得之結構量測點之加速度$\{\hat{\ddot{x}}(t)\}$、速度$\{\hat{\dot{x}}(t)\}$與位移$\{\hat{x}(t)\}$。需識別之系統參數共有$2+N+(3N_g+2)$個，將其分為三組：(1)第一組為模態自然頻率ω_n與模態阻尼比ξ_n之2個參數；(2)第二組為模態向量$\{\phi_n\}$之N個元素；(3)第三組為模態參與係數$\langle L_{nm}\rangle, \langle L_{nc}\rangle, \langle L_{nk}\rangle$之$3N_g$個元素及2個模態廣義一般化初始位移與初始速度$Y_{no}, \dot{Y}_{no}$。

若固定第一組與第二組參數，則模態之輸出函數與第三組參數值均為線性關係。亦即模態之反應輸出函數$\{\hat{\ddot{x}}_n(t)\}$可寫成下列線性關係

$$\{\hat{\ddot{x}}_n(t)\}=\{\phi_n\}\langle\beta_n\rangle\{\ddot{S}_n(t)\}=\{\phi_n\}\langle\ \langle L_{nm}\rangle\ \ \langle L_{nc}\rangle\ \ \langle L_{nk}\rangle\ \ Y_{no}\ \ \dot{Y}_{no}\ \rangle\begin{Bmatrix}\{\ddot{S}_{nm}(t)\}\\ \{\ddot{S}_{nc}(t)\}\\ \{\ddot{S}_{nk}(t)\}\\ \ddot{S}_{nd}(t)\\ \ddot{S}_{nv}(t)\end{Bmatrix}$$
$$=\{\phi_n\}(\langle L_{nm}\rangle\{\ddot{S}_{nm}(t)\}+\langle L_{nc}\rangle\{\ddot{S}_{nc}(t)\}+\langle L_{nk}\rangle\{\ddot{S}_{nk}(t)\}+Y_{no}\ddot{S}_{nd}(t)+\dot{Y}_{no}\ddot{S}_{nv}(t)) \tag{22.18}$$

式中之$\{\ddot{S}_n(t)\}$為下列運動方程式之反應值。注意前三式之初始值為0，後二式之作用力為0，即為自由振動。

$$\begin{aligned}
&\{\ddot{S}_{nm}(t)\}+2\xi_n\omega_n\{\dot{S}_{nm}(t)\}+\omega_n^2\{S_{nm}(t)\}=-\{\ddot{u}_g(t)\},\ \ \{S_{nm}(t_o)\}=\{\dot{S}_{nm}(t_o)\}=\{0\}\\
&\{\ddot{S}_{nc}(t)\}+2\xi_n\omega_n\{\dot{S}_{nc}(t)\}+\omega_n^2\{S_{nc}(t)\}=-\{\dot{u}_g(t)\},\ \ \{S_{nc}(t_o)\}=\{\dot{S}_{nc}(t_o)\}=\{0\}\\
&\{\ddot{S}_{nk}(t)\}+2\xi_n\omega_n\{\dot{S}_{nk}(t)\}+\omega_n^2\{S_{nk}(t)\}=-\{u_g(t)\},\ \ \{S_{nk}(t_o)\}=\{\dot{S}_{nk}(t_o)\}=\{0\}\\
&\ddot{S}_{nd}(t)+2\xi_n\omega_n\dot{S}_{nd}(t)+\omega_n^2 S_{nd}(t)=0\ ,\quad S_{nd}(t_o)=1\ ,\quad \dot{S}_{nd}(t_o)=0\\
&\ddot{S}_{nv}(t)+2\xi_n\omega_n\dot{S}_{nv}(t)+\omega_n^2 S_{nv}(t)=0\ ,\quad S_{nv}(t_o)=0\ ,\quad \dot{S}_{nv}(t_o)=1
\end{aligned}\tag{22.19}$$

此外若固定第一組與第三組參數，則模態之輸出函數在結構量測點之值與模態向量對應之值亦成正比之線性關係。亦即在固定第一組之ω_n與ξ_n2個參數時，可交替固定第二組或第三組參數，以計算另一組參數之最佳值。此時因輸出函數與參數間之線性關係，該二組模態參數之最佳值可由線性聯立方程式直接求得，有關計算式詳次節之推導。但因輸出函數與第一組參數ω_n或ξ_n間並不存在簡單的線性關係，該2參數之最佳值僅能利用非線性關係式求解。

式(22.19)之反應分析可採用各種方法，以下列示所附程式使用之計算式，為Beck與Dowling根據Duhamel積分式所導出。該法對只求一種反

應值(此處為加速度)時運算量最少。

$$\ddot{u}(t)+2\xi\omega\dot{u}(t)+\omega^2u(t)=p(t),\quad u(t_o)=u_o,\quad \dot{u}_o(t_o)=v_o \tag{22.20}$$

上列運動方程式之解以等時間距Δt之值表示，令$t_i=t_o+i\Delta t$，$p_i=p(t_i)$，假設二時段間之$p(t)$為直線，則其加速度$a_i=\ddot{u}(t_i)$可由下式求得

$$a_o = p_o-\omega^2u_o-2\xi\omega v_o \tag{22.21}$$

$$a_1 = A_3u_o+A_4v_o+B_1p_1+B_2p_o \tag{22.22}$$

$$a_i = A_1a_{i-1}+A_2a_{i-2}+B_1(p_i-2p_{i-1}+p_{i-2}),\quad i>1 \tag{22.23}$$

上式中之下列係數與時間t_i無關可先行求出

$$A_1 = 2e^{-\xi\omega\Delta t}\cos\omega_d\Delta t \tag{22.24}$$

$$A_2 = -e^{-2\xi\omega\Delta t} \tag{22.25}$$

$$A_3 = \omega^2e^{-\xi\omega\Delta t}\Big(\frac{\xi\omega\Delta t}{\omega_d\Delta t}\sin\omega_d\Delta t-\cos\omega_d\Delta t\Big) \tag{22.26}$$

$$A_4 = -\omega e^{-\xi\omega\Delta t}\Big(\frac{(1-2\xi^2)\omega\Delta t}{\omega_d\Delta t}\sin\omega_d\Delta t+2\xi\cos\omega_d\Delta t\Big) \tag{22.27}$$

$$B_1 = \frac{e^{-\xi\omega\Delta t}}{\omega_d\Delta t}\sin\omega_d\Delta t \tag{22.28}$$

$$B_2 = e^{-\xi\omega\Delta t}\Big(-\frac{1+\xi\omega\Delta t}{\omega_d\Delta t}\sin\omega_d\Delta t+\cos\omega_d\Delta t\Big) \tag{22.29}$$

$$\omega_d = \sqrt{1-\xi^2}\,\omega \tag{22.30}$$

22.6 線性模態參數之計算

首先考慮第三組參數之計算：將誤差函數寫成第三組參數之函數

$$\begin{aligned}&J(\langle L_{nm}\rangle,\langle L_{nc}\rangle,\langle L_{nk}\rangle,Y_{no},\dot{Y}_{no})\\&=\sum_{i=1}^{N}\frac{1}{V_{ni}}\int\Big(\hat{\ddot{u}}_{ni}(t)-\hat{\ddot{x}}_{ni}(t;\langle L_{nm}\rangle,\langle L_{nc}\rangle,\langle L_{nk}\rangle,Y_{no},\dot{Y}_{no})\Big)^2\,dt\\&=\sum_{i=1}^{N}\frac{1}{\int\hat{\ddot{u}}_{ni}^{\,2}(t)\,dt}\int\Big(\hat{\ddot{u}}_{ni}(t)-\phi_{ni}\langle\beta_n\rangle\{\ddot{S}_n(t)\}\Big)^2\,dt\end{aligned} \tag{22.31}$$

使J為最小之條件為$\{\nabla J\}=\{\frac{\partial J}{\partial\beta_j}\}=\{0\}$，即其分量為

$$\frac{\partial J}{\partial\beta_j}=2\sum_{i=1}^{N}\frac{1}{\int\hat{\ddot{u}}_{ni}^{\,2}(t)\,dt}\int\Big(\hat{\ddot{u}}_{ni}(t)-\phi_{ni}\langle\beta_n\rangle\{\ddot{S}_n(t)\}\Big)\Big(-\phi_{ni}\ddot{S}_{nj}(t)\Big)\,dt=0 \tag{22.32}$$

將含未知參數$\langle\beta_n\rangle$項移至等式左邊，其餘各項移至等式右邊可得

$$\sum_{i=1}^{N}\frac{\phi_{ni}^2}{\int\hat{\ddot{u}}_{ni}^2(t)\,dt}\int\ddot{S}_{nj}(t)\langle\beta_n\rangle\{\ddot{S}_n(t)\}\,dt=\sum_{i=1}^{N}\frac{\phi_{ni}}{\int\hat{\ddot{u}}_{ni}^2(t)\,dt}\int\ddot{S}_{nj}(t)\hat{\ddot{u}}_{ni}(t)\,dt$$

或

$$\int\ddot{S}_{nj}(t)\langle\beta_n\rangle\{\ddot{S}_n(t)\}\,dt=\frac{1}{\sum_{i=1}^{N}\frac{\phi_{ni}^2}{\int\hat{\ddot{u}}_{ni}^2(t)\,dt}}\sum_{i=1}^{N}\frac{\phi_{ni}}{\int\hat{\ddot{u}}_{ni}^2(t)\,dt}\int\ddot{S}_{nj}(t)\hat{\ddot{u}}_{ni}(t)\,dt\quad(22.33)$$

其詳細矩陣式為

$$\begin{bmatrix}\int\ddot{S}_{n1}(t)\ddot{S}_{n1}(t)dt & \int\ddot{S}_{n1}(t)\ddot{S}_{n2}(t)dt & \cdots & \int\ddot{S}_{n1}(t)\ddot{S}_{n(3N_g+2)}(t)dt\\ & \int\ddot{S}_{n2}(t)\ddot{S}_{n2}(t)dt & \cdots & \int\ddot{S}_{n2}(t)\ddot{S}_{n(3N_g+2)}(t)dt\\ & & \ddots & \vdots\\ sym. & & & \int\ddot{S}_{n(3N_g+2)}(t)\ddot{S}_{n(3N_g+2)}(t)dt\end{bmatrix}\begin{Bmatrix}\beta_1\\ \beta_2\\ \vdots\\ \beta_{3N_g+2}\end{Bmatrix}$$

$$=\frac{1}{\sum_{i=1}^{N}\frac{\phi_{ni}^2}{V_{ni}}}\begin{Bmatrix}\frac{\phi_{n1}}{V_{n1}}\int\ddot{S}_{n1}(t)\hat{\ddot{u}}_{n1}(t)dt+\cdots+\frac{\phi_{nN}}{V_{nN}}\int\ddot{S}_{n1}(t)\hat{\ddot{u}}_{nN}(t)dt\\ \frac{\phi_{n1}}{V_{n1}}\int\ddot{S}_{n2}(t)\hat{\ddot{u}}_{n1}(t)dt+\cdots+\frac{\phi_{nN}}{V_{nN}}\int\ddot{S}_{n2}(t)\hat{\ddot{u}}_{nN}(t)dt\\ \vdots\\ \frac{\phi_{n1}}{V_{n1}}\int\ddot{S}_{n(3N_g+2)}(t)\hat{\ddot{u}}_{n1}(t)dt+\cdots+\frac{\phi_{nN}}{V_{nN}}\int\ddot{S}_{n(3N_g+2)}(t)\hat{\ddot{u}}_{nN}(t)dt\end{Bmatrix}$$

其次考慮第二組參數之計算：將誤差函數寫成第二組參數之函數

$$\begin{aligned}J(\{\phi_n\})&=\sum_{i=1}^{N}\frac{1}{V_{ni}}\int\left(\hat{\ddot{u}}_{ni}(t)-\hat{\ddot{x}}_{ni}(t;\phi_{ni})\right)^2dt\\&=\sum_{i=1}^{N}\frac{1}{\int\hat{\ddot{u}}_{ni}^2(t)\,dt}\int\left(\hat{\ddot{u}}_{ni}(t)-\phi_{ni}\langle\beta_n\rangle\{\ddot{S}_n(t)\}\right)^2dt\end{aligned}\quad(22.34)$$

使J為最小之條件為$\{\nabla J\}=\{\frac{\partial J}{\partial\phi_{nj}}\}=\{0\}$，即其分量為

$$\frac{\partial J}{\partial\phi_{nj}}=2\frac{1}{\int\hat{\ddot{u}}_{nj}^2(t)\,dt}\int\left(\hat{\ddot{u}}_{nj}(t)-\phi_{nj}\langle\beta_n\rangle\{\ddot{S}_n\}\right)\left(-\langle\beta_n\rangle\{\ddot{S}_n(t)\}\right)dt=0\quad(22.35)$$

因結構模態反應輸出函數之第j分量僅與模態向量之第j分量有關，上式每一條件式僅含一未知參數，即模態向量之一分量。因此各模態之分量可由下列單一關係式求得

$$\phi_{nj}=\frac{\int\hat{\ddot{u}}_{nj}(t)\langle\beta_n\rangle\{\ddot{S}_n(t)\}\,dt}{\int\left(\langle\beta_n\rangle\{\ddot{S}_n(t)\}\right)^2dt}\quad(22.36)$$

在將單一模態之誤差函數最小化時，須由非線性最佳化法交替調整第一組參數之ω_n與ξ_n直至收斂為止。而於每次改變第一組參數ω_n或ξ_n後，即須反覆以上列關係式交替計算第三組參數與第二組參數直至收斂為止。如此纔完成單一模態之參數識別。

22.7 系統識別分析步驟

以下說明模態最小化法識別第一組參數之步驟(詳程式Optiminze)：

(1)第一層迴圈：控制Mf=Mi:NumM。NumM為欲識別之模態總數。Mi=1時為以漸近方式，先識別出一個模態之參數後再增加一個模態繼續識別參數，直至識別NumM個模態為止。當Mi=NumM時，則直接對NumM個模態之參數識別。一般情形，取Mi=NumM時計算時間會較少，但如初始值與最佳值相差太大可能不會收斂。

(2)第二層迴圈：控制Swep=1:10。該層迴圈每做一次循環，即對Mf個模態，自第一模態至第Mf模態依次做單一模態之最小化識別。當每一模態參數改變量均很小時，該迴圈會提早結束。

(3)第三層迴圈：控制Mode=1:Mf。此迴圈即對第Mode模態做參數識別。此時，其他模態之參數保持不變。

以下迴圈詳副程式MinimizeErr。此副程式主要在求第Mode模態之ω_n與ξ_n使誤差函數為最小。對該二參數一次改變其中一個做線性搜尋。

(4)第四層迴圈：控制Count=1:12。此迴圈控制ω_n與ξ_n交替調整之次數。

(5)第五層迴圈：控制I=1:2。此迴圈決定對那一參數做調整：I=1時調整ω_n，I=2時調整ξ_n。

(6)第六層迴圈：控制Step=1:16。此迴圈控制線性搜尋之過程及次數。

以下說明對第二組與第三組參數交替調整之有關計算步驟(詳副程式ComputeAmM)：

(1)迴圈10與20：計算式(22.33)左邊矩陣之係數Sjk，與右邊向量中之積分項SjA，該積分值針對各結構量測點分列成數行(NumS為式(22.33)中之N)。

(2)迴圈30與40：計算式(22.33)右邊向量之係數RHS，該值由迴圈20算得之積分值SjA乘對應係數後累加而得。

(3)解式(22.33)即得B，該值為第三組參數$\langle\beta_n\rangle$。
(4)迴圈60：計算BSB，該值為式(22.36)之分母。
(5)迴圈80：計算BSA，該值為式(22.36)之分子。
(6)迴圈70：由式(22.36)求模態向量對應各量測點之值，即為第二組參數$\{\phi_n\}$。
(7)迴路95：控制第三組參數與第二組參數交替計算之次數。

22.8 結構系統參數識別程式

［表一］ 結構系統參數識別程式

```
**********************************************************************
**    MAIN PROGRAM  BridgeIdent3D                                   **
**  ================================================================**
**    System Identification analysis for Multiple Supported Inputs  **
**    of Bridge Structures by Modal Minimization Method ( Beck,J.L., **
**    "Determining Models of Structures from Earthquake Records",    **
**    CalTech, CA, 1978 )                                           **
**  ================================================================**
      CALL  InputData
      CALL  Optimize
      CALL  OutputResult
      STOP
      END
**********************************************************************
      SUBROUTINE  InputData
**  ================================================================**
**    UserInput   computational parameters, initial guess of modal  **
**                parameters, pseudo influence matrix               **
**    ReadData    of recorded ground and structure responses        **
**    Initialize  recorded and computed responses, normalization c  **
**  ================================================================**
      CALL  UserInput
      CALL  ReadData
      CALL  Initialize
      RETURN
      END
**********************************************************************
      SUBROUTINE  UserInput
**  ================================================================**
**    READ  NumM,NumF,NumG,NumS,NumT,Mi,Ti,Tf                       **
**    READ  Omg,Ksi    INITIALIZE  Phi    READ  Inf                 **
**    ---------------------------------------------------------     **
**    NumM           = Number of Modes to be analyized              **
**    NumF           = Number of support Forcing inputs   = 1,2,3   **
**    NumG           = Number of instrumented Ground support DOFs   **
**    NumS           = Number of instrumented Superstructure DOFs   **
**    NumT           = Number of Time steps in input data file      **
**    Mi             = Initial number of Modes of the analysis      **
**    Ti,Tf          = Initial and Final Time step of the analysis  **
```

```
**    .............................................................    **
**    Omg(NumM)      = estimated modal frequencies                      **
**    Ksi(NumM)      = estimated modal damping coefficients             **
**    Phi(NumS,NumM) = estimated mode  shapes                  = 1     **
**    Inf(NumS,NumG) = pseudo Influence matrix                          **
**  ================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumM,NumF,NumG,NumS,NumT,Mi,Ti,Tf , Mode,Grnd,Stru
      REAL*8     Omg(MaxM),Ksi(MaxM),Phi(MaxS,MaxM),Inf(MaxS,MaxG)
      COMMON  /M[]/  NumM
      COMMON  /F[]/  NumF
      COMMON  /G[]/  NumG
      COMMON  /S[]/  NumS
      COMMON  /T[]/  NumT
      COMMON  /Tif/  Ti,Tf
      COMMON  /Mii/  Mi
      COMMON  /X[]/  Omg,Ksi
      COMMON  /Y[]/  Phi
      COMMON  /I[]/  Inf
**  ================================================================  **
      WRITE(*,100)
 100  FORMAT(' Bridge Identification by ''Modal Minimization Method'' ')
      OPEN (1,FILE='BI3D1.In',FORM='FORMATTED',ACCESS='SEQUENTIAL')
**    +-------------------------------------------------------------+    **
**    |  READ  NumM,NumF,NumG,NumS,NumT,Mi,Ti,Tf                    |    **
**    +-------------------------------------------------------------+    **
      READ (1,'(8I5)')  NumM,NumF,NumG,NumS,NumT,Mi,Ti,Tf
      IF (NumM.GT.MaxM)           STOP ' UserInput : NumM > MaxM '
      IF (NumF.GT.MaxF)           STOP ' UserInput : NumF > MaxF '
      IF (NumG.GT.MaxG)           STOP ' UserInput : NumG > MaxG '
      IF (NumS.GT.MaxS)           STOP ' UserInput : NumS > MaxS '
      IF (NumT.GT.MaxT)           STOP ' UserInput : NumT > MaxT '
      IF (Mi.LT.1.OR.Mi.GT.NumM)  Mi = NumM
      IF (Ti.LT.1)                Ti = 1
      IF (Tf.GT.NumT)             Tf = NumT
      Tf = Ti + (Tf-Ti)/2*2
**    .............................................................    **
*      OPEN (6,FILE='BI3D1.Deb',FORM='FORMATTED')
      WRITE(*,'('' Max M F G S T = '',5I5)')  MaxM,MaxF,MaxG,MaxS,MaxT
      WRITE(*,'('' Num M F G S T = '',5I5)')  NumM,NumF,NumG,NumS,NumT
      WRITE(*,'(''     Mi  Ti Tf = '',3I5)')  Mi,Ti,Tf
**    +-------------------------------------------------------------+    **
**    |  READ  Omg(NumM),Ksi(NumM)                                  |    **
**    +-------------------------------------------------------------+    **
      DO 10  Mode = 1,NumM
          READ (1,'(2F5.0)')  Omg(Mode),Ksi(Mode)
          WRITE(*,'('' Omg Ksi = '',2F8.4)')  Omg(Mode),Ksi(Mode)
  10  CONTINUE
**    +-------------------------------------------------------------+    **
**    |  INITIALIZE  Phi(NumS,NumM)                                 |    **
**    +-------------------------------------------------------------+    **
      DO 20  Mode = 1,NumM
      DO 20  Stru = 1,NumS
          Phi(Stru,Mode) = 1.D0
  20  CONTINUE
**    +-------------------------------------------------------------+    **
**    |  READ  Inf(NumS,NumG)                                       |    **
**    +-------------------------------------------------------------+    **
      DO 30  Grnd = 1,NumG
          READ (1,'(16F5.0)')  ( Inf(Stru,Grnd) , Stru=1,NumS )
```

```
          WRITE(*,'('' Inf = '')')
          WRITE(*,'(10F8.4)')  ( Inf(Stru,Grnd) , Stru=1,NumS )
  30  CONTINUE
      CLOSE(1)
      RETURN
      END
*************************************************************************
      SUBROUTINE  ReadData
**  ================================================================  **
**    READ  Ag,At                                                     **
**    ------------------------------------------------------------    **
**    Ag(NumT,NumG,NumF) = recorded Ground support Acce,velo,displ    **
**    At(NumT,NumS)      = recorded Total superstructure Accel.       **
**  ================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumF,NumG,NumS,NumT , Forc,Grnd,Stru,Time
      REAL*8     Ag(MaxT,MaxG,MaxF),At(MaxT,MaxS)
      REAL*4     Temp(MaxT)
      COMMON  /F[]/  NumF
      COMMON  /G[]/  NumG
      COMMON  /S[]/  NumS
      COMMON  /T[]/  NumT
      COMMON  /Ag[]/ Ag
      COMMON  /At[]/ At
**  ================================================================  **
      OPEN (3,FILE='BI3D1.Rec',FORM='FORMATTED',ACCESS='SEQUENTIAL')
**    +-------------------------------------------------------+    **
**    |  READ  Ag(NumT,NumG,NumF)                             |    **
**    +-------------------------------------------------------+    **
      DO 10  Grnd = 1,NumG
      DO 10  Forc = 1,NumF
          READ (3,*)  ( Temp(Time) , Time=1,NumT )
      DO 10  Time = 1,NumT
          Ag(Time,Grnd,Forc) = Temp(Time)
  10  CONTINUE
**    +-------------------------------------------------------+    **
**    |  READ  At(NumT,NumS)                                  |    **
**    +-------------------------------------------------------+    **
      DO 20  Stru = 1,NumS
          READ (3,*)  ( Temp(Time) , Time=1,NumT )
      DO 20  Time = 1,NumT
          At(Time,Stru) = Temp(Time)
  20  CONTINUE
      CLOSE(3)
      RETURN
      END
*************************************************************************
      SUBROUTINE  Initialize
**  ================================================================  **
**    COMPUTE  Ar    INITIALIZE  Am    COMPUTE  Norm                  **
**    ------------------------------------------------------------    **
**    Ar(T,NumS) = recorded Relative Acceleration      = At-Inf*Ag    **
**    Am(T,NumM) = computed Modal relative Acceleration               **
**    Norm(NumS) = Normalization constant of Ar      = 1/Int(Ar^2)    **
**  ================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumM,NumG,NumS,Ti,Tf , Mode,Grnd,Stru,Time
      REAL*8     Ag(MaxT,MaxG,MaxF),At(MaxT,MaxS)
      REAL*8     Ar(MaxT,MaxS),Am(MaxT,MaxM),Norm(MaxS)
      REAL*8     Inf(MaxS,MaxG),InfAg,Intg,dT/0.02D0/
```

```
      COMMON  /M[]/  NumM
      COMMON  /G[]/  NumG
      COMMON  /S[]/  NumS
      COMMON  /Tif/  Ti,Tf
      COMMON  /Ag[]/ Ag
      COMMON  /At[]/ At
      COMMON  /Ar[]/ Ar
      COMMON  /Am[]/ Am
      COMMON  /I[]/  Inf
      COMMON  /No[]/ Norm
**  ================================================================  **
**    +-------------------------------------------------------+    **
**    |  COMPUTE  Ar(T,NumS) = At(T,NumS)                     |    **
**    |                      - Inf(NumS,NumG)*Ag(T,NumG,1)    |    **
**    +-------------------------------------------------------+    **
      DO 10  Stru = 1,NumS
      DO 10  Time = Ti,Tf
          InfAg = 0.D0
          DO 20  Grnd = 1,NumG
              InfAg = InfAg + Inf(Stru,Grnd)*Ag(Time,Grnd,1)
  20      CONTINUE
          Ar(Time,Stru) = At(Time,Stru) - InfAg
  10  CONTINUE
**    +-------------------------------------------------------+    **
**    |  INITIALIZE  Am(T,NumM) = 0                           |    **
**    +-------------------------------------------------------+    **
      DO 30  Mode = 1,NumM
      DO 30  Time = Ti,Tf
          Am(Time,Mode) = 0.D0
  30  CONTINUE
**    +-------------------------------------------------------+    **
**    |  COMPUTE  Norm(NumS) = 1/Int(Ar(T,NumS)^2)            |    **
**    +-------------------------------------------------------+    **
      DO 40  Stru = 1,NumS
          Norm(Stru) = 1 / Intg ( Ar(Ti,Stru),Ar(Ti,Stru),Ti,Tf,dT )
  40  CONTINUE
      WRITE(*,'('' Norm = '',1P8F10.4)')  ( Norm(Stru) , Stru=1,NumS )
      RETURN
      END
**********************************************************************
      FUNCTION  Intg ( X,Y,Ti,Tf,dT )
**  ================================================================  **
**    COMPUTE  the numerical integration  Intg  of function  X*Y      **
**        over the time interval  Ti~Tf  with the fixed step size     **
**        dT  by using the Simpson's rule ( 1 4 2 4 2 ... 4 1 )       **
**    !!  (Tf-Ti) must be an even number which is multiple of 2       **
**  ================================================================  **
      INTEGER    Ti,Tf,Time
      REAL*8     Intg,X(Ti:Tf),Y(Ti:Tf),dT,A,B
**  ================================================================  **
      A=0.D0
      B=0.D0
      DO 10  Time = Ti+1,Tf,2
          A = A + X(Time)*Y(Time)
          B = B + X(Time+1)*Y(Time+1)
  10  CONTINUE
      Intg = (X(Ti)*Y(Ti)-X(Tf)*Y(Tf)+4.D0*A+2.D0*B) / 3.D0 * dT
      RETURN
      END
**********************************************************************
```

```
      SUBROUTINE  Optimize
**  ================================================================  **
**    For each Swep of Mode = 1,NumM  :  ComputeArM  MinimizeErr      **
**    ----------------------------------------------------------    **
**    Err  = Error function to be minimized                           **
**    OmgO = Old Omg before MinimizeErr                               **
**    KsiO = Old Ksi before MinimizeErr                               **
**    Diff = relative Difference between OmgO,KsiO and Omg,Ksi        **
**  ================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumM,Mi,Mf , Mode,Swep
      REAL*8     Omg(MaxM),Ksi(MaxM),Err,OmgO,KsiO,Diff
      COMMON  /M[]/  NumM
      COMMON  /Mii/  Mi
      COMMON  /X[]/  Omg,Ksi
**  ================================================================  **
      DO 10  Mf = Mi,NumM
      DO 20  Swep = 1,10
          Diff = 0.D0
          DO 30  Mode = 1,Mf
**    +-------------------------------------------------------+    **
**    |  STORE  OmgO,KsiO                                     |    **
**    +-------------------------------------------------------+    **
              OmgO = Omg(Mode)
              KsiO = Ksi(Mode)
**    +-------------------------------------------------------+    **
**    |  OPTIMIZE  Omg,Ksi    by    MINIMIZE  Err             |    **
**    +-------------------------------------------------------+    **
              CALL  ComputeArM ( Mode )
              CALL  MinimizeErr ( Mode,Omg(Mode),Ksi(Mode),Err )
**    +-------------------------------------------------------+    **
**    |  COMPUTE  Diff                                        |    **
**    +-------------------------------------------------------+    **
              Diff = DMAX1 (DABS(OmgO/Omg(Mode)-1.D0)
     &                     ,DABS(KsiO/Ksi(Mode)-1.D0),Diff)
**    .........................................................    **
              WRITE(*,'('' Omg Ksi Err Diff OmgD KsiD = '',6F10.5)')
     &                    Omg(Mode),Ksi(Mode),Err,Diff,
     &                        1.D0-OmgO/Omg(Mode),1.D0-KsiO/Ksi(Mode)
  30      CONTINUE
          IF (Diff.LT.0.001D0)  GOTO 10
  20  CONTINUE
  10  CONTINUE
      RETURN
      END
**********************************************************************
      SUBROUTINE  ComputeArM ( M )
**  ================================================================  **
**    COMPUTE  ArM                                                    **
**    ----------------------------------------------------------    **
**    ArM(T,NumS) = recorded Relative Modal Accel = Ar-Sum(Am*Phi)    **
**  ================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumM,NumS,Ti,Tf , M , Mode,Stru,Time
      REAL*8     Ar(MaxT,MaxS),Am(MaxT,MaxM),Phi(MaxS,MaxM)
      REAL*8     ArM(MaxT,MaxS)
      COMMON  /M[]/   NumM
      COMMON  /S[]/   NumS
      COMMON  /Tif/   Ti,Tf
      COMMON  /Ar[]/  Ar
```

```
      COMMON  /Am[]/  Am
      COMMON  /ArM[]/ ArM
      COMMON  /Y[]/   Phi
**  ================================================================  **
**    +-------------------------------------------------------------+   **
**    |  COMPUTE  ArM(T,NumS) = Ar(T,NumS)                          |   **
**    |                       - Sum(Am(T,NumM)*Phi(NumS,NumM))      |   **
**    +-------------------------------------------------------------+   **
      DO 10  Stru = 1,NumS
      DO 10  Time = Ti,Tf
          ArM(Time,Stru) = Ar(Time,Stru) + Am(Time,M)*Phi(Stru,M)
      DO 10  Mode = 1,NumM
          ArM(Time,Stru) = ArM(Time,Stru) - Am(Time,Mode)*Phi(Stru,Mode)
  10  CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE  MinimizeErr ( M,OmgM,KsiM,Err )
**  ================================================================  **
**    MINIMIZE  Err  to find Optimal OmgM,KsiM by using T.W.Lin's     **
**        nonlinear minimization procedures and  Xmin                 **
**    ..........................................................      **
**    ComputeErr : COMPUTE  Err  for given values of OmgM and KsiM    **
**    Xmin       : FIND  Optimum values of OmgM or KsiM               **
**    ----------------------------------------------------------      **
**    Err       = Error function to be minimized                      **
**    OmgM,KsiM = variables of error function to be optimized         **
**    ..........................................................      **
**    X(1:2) = initial and optimum values of variables OmgM,KsiM      **
**    F*     = Function values of X                        = Err      **
**    TypX   = Typical values of X                                    **
**    TolX   = Tolerance value for X/TypX                             **
**    TolF   = Tolerance value for dF(X)/dX                           **
**    dX     = new step size of X              = Max(|Xm-Xi|/TypX)    **
**           = maximum relative total moving distance                 **
**    Count  = number of Count over minimizing OmgM and KsiM <= 12    **
**    Step   = number of Step in minimizing OmgM or KsiM     <= 16    **
**  ================================================================  **
      INTEGER    M,I,Count,Step,Valley
      REAL*8     OmgM,KsiM,Err,X(2),TypX(2),TolX,TolF,dX,dXOld,Xmin
      REAL*8     Xo,Fo,Xx,Fx,Xm,Fm,Xn,Xi,Rate,Buf(14)
**  ================================================================  **
**    +-------------------------------------------------------------+   **
**    |  INITIALIZE  X(2),TypX(2),TolX,dX                           |   **
**    +-------------------------------------------------------------+   **
      X(1)    = OmgM
      X(2)    = KsiM
      TypX(1) = 5.D0
      TypX(2) = 0.05D0
      TolX    = 0.0001D0
      dX      = 0.1D0
**    +-------------------------------------------------------------+   **
**    |  ITERATE  minimizing OmgM and KsiM up to Count=12 times     |   **
**    +-------------------------------------------------------------+   **
      Count = 0
  95  Count = Count + 1
      dXOld = dX
      dX    = 0.D0
      DO 10  I = 1,2
**    +-------------------------------------------------------------+   **
```

```
**    |  INITIALIZE  current Xx,Fx , optimum Xm,Fm , next Xn     |    **
**    |  SAVE        initial Xi                                  |    **
**    +--------------------------------------------------------+    **
          Xx = 1.D30
          Fx = 1.D30
          Xm = Xx
          Fm = Fx
          Xn = X(I)
          Xi = X(I)
**    +--------------------------------------------------------+    **
**    |  ITERATE  minimizing OmgM or KsiM up to Step=16 times  |    **
**    +--------------------------------------------------------+    **
          Step = 0
  85      Step = Step + 1
**    +--------------------------------------------------------+    **
**    |  SAVE  previous Xo,Fo                                  |    **
**    +--------------------------------------------------------+    **
          Xo = Xx
          Fo = Fx
**    +--------------------------------------------------------+    **
**    |  USE  next Xn from Xmin as current Xx,X for ComputeErr |    **
**    +--------------------------------------------------------+    **
          Xx   = Xn
          X(I) = Xx
          CALL  ComputeErr ( M,X(1),X(2),Fx )
**    +--------------------------------------------------------+    **
**    |  SAVE  optimum Xm,Fm during iteration Steps            |    **
**    +--------------------------------------------------------+    **
          IF (Fx.LT.Fm)  THEN
              Xm = Xx
              Fm = Fx
          ENDIF
**    +--------------------------------------------------------+    **
**    |  OBTAIN  next Xn from Xmin for next iteration Step     |    **
**    +--------------------------------------------------------+    **
          Xn = Xmin ( Xx,Fx,dXOld*TypX(I),Step,Valley,Buf )
**    +--------------------------------------------------------+    **
**    |  CHECK  convergence of X,F                             |    **
**    +--------------------------------------------------------+    **
          IF (Step.EQ.2)  TolF = TolX * DABS((Fx-Fo)/(Xx-Xo))
          IF (Step.LT.5 .AND. Valley.EQ.0)  GOTO 85
          IF (Step.LT.16 .AND.
     &        DABS(Xx-Xo).GT.TolX*TypX(I) .AND.
     &        DABS(Fx-Fo).GT.TolF*DABS(Xx-Xo))  GOTO 85
**    +--------------------------------------------------------+    **
**    |  UPDATE  X , dX = Max(|Xm-Xi|/TypX)                    |    **
**    +--------------------------------------------------------+    **
          X(I) = Xm
          Rate = 0.5 * DABS(Xm-Xi)/TypX(I)
          IF (Rate.GT.dX)  dX = Rate
          WRITE(*,'(I4,F7.4)')  Step,Rate
  10  CONTINUE
**    +--------------------------------------------------------+    **
**    |  CHECK  convergence of OmgM or KsiM                    |    **
**    +--------------------------------------------------------+    **
      IF (dX.GT.dXOld)  dX = dXOld
      IF (dX.GT.TolX.AND.Count.LT.12)  GOTO 95
**    +--------------------------------------------------------+    **
**    |  RETURN  the minimum Err and the optimum OmgM,KsiM     |    **
**    +--------------------------------------------------------+    **
```

```
      OmgM = X(1)
      KsiM = X(2)
      CALL  ComputeErr ( M,OmgM,KsiM,Err )
      RETURN
      END
*************************************************************************
      SUBROUTINE  ComputeErr ( M,OmgM,KsiM,Err )
**  ===================================================================  **
**    COMPUTE  AmM,PhiM,Err                                              **
**    -------------------------------------------------------------    **
**    AmM(T)     = computed Modal Acceleration of the Mth mode           **
**    PhiM(NumS) = computed mode shapes of the Mth mode                  **
**    Err        = Error function = Sum(Norm*Int((ArM-AmM*PhiM)^2))      **
**    PF         = Penalty Function to Err for computed OmgM,KsiM        **
**  ===================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumS,Ti,Tf , M , Stru,Time
      REAL*8     ArM(MaxT,MaxS),Am(MaxT,MaxM),Phi(MaxS,MaxM),Norm(MaxS)
      REAL*8     Err,OmgM,KsiM,Diff(MaxT),Intg,dT/0.02D0/,PF
      COMMON  /S[]/   NumS
      COMMON  /Tif/   Ti,Tf
      COMMON  /ArM[]/ ArM
      COMMON  /Am[]/  Am
      COMMON  /Y[]/   Phi
      COMMON  /No[]/  Norm
**  ===================================================================  **
**    +-------------------------------------------------------------+    **
**    |  COMPUTE  AmM(T),PhiM(NumS)                                 |    **
**    +-------------------------------------------------------------+    **
      CALL  ComputeAmM ( OmgM,KsiM,Phi(1,M),Am(1,M) )
**    +-------------------------------------------------------------+    **
**    |  COMPUTE  Err = Sum ( Norm(NumS) * Int ( (ArM(T,NumS)       |    **
**    |               - AmM(T)*PhiM(NumS))^2 ) )                    |    **
**    +-------------------------------------------------------------+    **
      Err = 0.D0
      DO 10  Stru = 1,NumS
          DO 20  Time = Ti,Tf
              Diff(Time) = ArM(Time,Stru) - Am(Time,M) * Phi(Stru,M)
  20      CONTINUE
          Err = Err + Norm(Stru) * Intg (Diff(Ti),Diff(Ti),Ti,Tf,dT)
  10  CONTINUE
**    +-------------------------------------------------------------+    **
**    |  ADD  PF to Err for OmgM<0 and KsiM<0 and KsiM>0.5          |    **
**    +-------------------------------------------------------------+    **
      Err = Err + PF(    OmgM,100.D0,0.001D0,1.D0,1.D0,1)
      Err = Err + PF(    KsiM,100.D0,0.001D0,1.D0,1.D0,1)
      Err = Err + PF(0.5-KsiM,100.D0,0.001D0,1.D0,1.D0,1)
      WRITE(*,'('' OmgM,KsiM,Err = '',2F10.5,F7.3)')  OmgM,KsiM,Err
      RETURN
      END
*************************************************************************
      SUBROUTINE  ComputeAmM ( OmgM,KsiM,PhiM,AmM )
**  ===================================================================  **
**    COMPUTE  AmM,PhiM  by iteration of linear modal minimization       **
**        for linear parameter PhiM and participation factor B           **
**    ...............................................................    **
**    COMPUTE S,Sjk,SjA << COMPUTE RHS , SOLVE B , COMPUTE BSB,BSA       **
**        , COMPUTE PhiM , CHECK convergence >> NORMALIZE PhiM,B ,       **
**        COMPUTE AmM                                                    **
**    -------------------------------------------------------------    **
```

```
**    ArM(T,NumS) = recorded Relative Modal Acceleration              **
**    AmM(T)      = computed Modal Acceleration         = Sum(B*S)    **
**    OmgM,KsiM   = Modal frequency & Modal damping ratio             **
**    PhiM(NumS)  = Mode shapes  = Int(SumBS*ArM)/Int(SumBS*SumBS)    **
**    Norm(NumS)  = Normalization constant of Ar                      **
**    ..........................................................      **
**    B(NumV)        = participation factor    = soln of Sjk*B=RHS    **
**    S(T,NumV)      = unit accel response of SDOF linear oscil.      **
**    Sjk(NumV,NumV) = Int(Sj*Sk)                                     **
**    SjA(NumV,NumS) = Int(Sj*ArM)                                    **
**    RHS(NumV)      = Right Hand Side of linear equations            **
**                   = Sum(Norm*PhiM*SjA)/Sum(Norm*PhiM^2)            **
**    BSB            = Bj*Sjk*Bk = Int(Sum(B*S)*Sum(B*S))             **
**    BSA            = Bj*SjA = Int(Sum(B*S)*ArM)                     **
**    NumV           = Number of Variables of system = NumG*NumF+2    **
**  ================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumS,Ti,Tf,NumV , Stru,Time , J,K,Count
      REAL*8     Ag(MaxT,MaxG,MaxF),ArM(MaxT,MaxS),AmM(MaxT)
      REAL*8     OmgM,KsiM,PhiM(MaxS),Norm(MaxS),PhiMax,PhiOld
      REAL*8     B(MaxV),S(MaxT,MaxV),Sjk(MaxV,MaxV),SjA(MaxV,MaxS)
      REAL*8     RHS(MaxV),BSB,BSA,Intg,dT/0.02D0/,Sum
      COMMON  /S[]/   NumS
      COMMON  /Tif/   Ti,Tf
      COMMON  /V[]/   NumV
      COMMON  /Ag[]/  Ag
      COMMON  /ArM[]/ ArM
      COMMON  /No[]/  Norm
**  ================================================================  **
**    +-------------------------------------------------------+     **
**    |  COMPUTE  S(T,NumV)                                   |     **
**    +-------------------------------------------------------+     **
      CALL  ComputeS ( OmgM,KsiM,Ag,S )
**    +-------------------------------------------------------+     **
**    |  COMPUTE  Sjk = Skj = Int(Sj*Sk) , SjA = Int(Sj*ArM)  |     **
**    +-------------------------------------------------------+     **
      DO 10  J = 1,NumV
      DO 10  K = 1,J
          Sjk(J,K) = Intg ( S(Ti,J),S(Ti,K),Ti,Tf,dT )
          Sjk(K,J) = Sjk(J,K)
  10  CONTINUE
      DO 20  J = 1,NumV
      DO 20  Stru = 1,NumS
          SjA(J,Stru) = Intg ( S(Ti,J),ArM(Ti,Stru),Ti,Tf,dT )
  20  CONTINUE
**    ---------------------------------------------------------     **
      Count = 0
  95  Count = Count + 1
**    +-------------------------------------------------------+     **
**    |  COMPUTE  RHS = Sum(Norm*PhiM*SjA) / Sum(Norm*PhiM^2) |     **
**    +-------------------------------------------------------+     **
      Sum = 0.D0
      DO 30  Stru = 1,NumS
          Sum = Sum + Norm(Stru) * PhiM(Stru)**2
  30  CONTINUE
      DO 40  J = 1,NumV
          RHS(J) = 0.D0
          DO 50  Stru = 1,NumS
              RHS(J) = RHS(J) + Norm(Stru) * PhiM(Stru) * SjA(J,Stru)
  50      CONTINUE
```

```
          RHS(J) = RHS(J) / Sum
  40  CONTINUE
**    +--------------------------------------------------------+    **
**    |  SOLVE  B from linear equations  Sjk * B = RHS         |    **
**    +--------------------------------------------------------+    **
      CALL  SolveSystem ( Sjk,RHS,B,NumV )
**    +--------------------------------------------------------+    **
**    |  COMPUTE  BSB = Bj*Sjk*Bk = Int(Sum(B*S)*Sum(B*S))     |    **
**    +--------------------------------------------------------+    **
      BSB =  0.D0
      DO 60  J = 1,NumV
      DO 60  K = 1,NumV
          BSB = BSB + B(J) * Sjk(J,K) * B(K)
  60  CONTINUE
**    ---------------------------------------------------------    **
      PhiMax = 0.D0
      Sum = 0.D0
      DO 70  Stru = 1,NumS
          PhiOld = PhiM(Stru)
**    +--------------------------------------------------------+    **
**    |  COMPUTE  BSA = Bj*SjA = Int(Sum(B*S)*ArM)             |    **
**    +--------------------------------------------------------+    **
          BSA = 0.D0
          DO 80  J = 1,NumV
              BSA = BSA + B(J) * SjA(J,Stru)
  80      CONTINUE
**    +--------------------------------------------------------+    **
**    |  COMPUTE  PhiM = BSA / BSB                             |    **
**    +--------------------------------------------------------+    **
          PhiM(Stru) = BSA / BSB
          IF (DABS(PhiM(Stru)).GT.PhiMax)  PhiMax = DABS(PhiM(Stru))
          Sum = Sum + DABS(PhiM(Stru)-PhiOld)
  70  CONTINUE
**    +--------------------------------------------------------+    **
**    |  CHECK  convergence of PhiM                            |    **
**    +--------------------------------------------------------+    **
      IF (Sum.GT.PhiMax*NumS*0.001D0.AND.Count.LT.10)  GOTO 95
**    +--------------------------------------------------------+    **
**    |  NORMALIZE  PhiM,B                                     |    **
**    +--------------------------------------------------------+    **
      DO 90  Stru = 1,NumS
          PhiM(Stru) = PhiM(Stru) / PhiMax
  90  CONTINUE
      DO 100  J = 1,NumV
          B(J) = B(J) * PhiMax
 100  CONTINUE
*      WRITE(*,'('' PhiM = '',10F8.4)')  ( PhiM(Stru) , Stru=1,NumS )
*      WRITE(*,'('' B = '',10F8.4)')  ( B(J) , J=1,NumV )
**    +--------------------------------------------------------+    **
**    |  COMPUTE  AmM = Sum(B*S)                               |    **
**    +--------------------------------------------------------+    **
      DO 110  Time = Ti,Tf
          AmM(Time) = 0.D0
          DO 110  J = 1,NumV
              AmM(Time) = AmM(Time) + B(J) * S(Time,J)
 110  CONTINUE
      RETURN
      END
**********************************************************************
      SUBROUTINE ComputeS ( Omg,Ksi,G,S )
```

```
**  ================================================================  **
**    COMPUTE  unit acceleration response S(T,NumV) of SDOF linear    **
**        oscillator  S + 2*Ksi*Omg*V + Omg^2*X = - G  by using       **
**        Duhamel's integral over the time interval Ti~Tf ( Beck,     **
**        J.L. & Dowling,M.J. , " Quick Algorithms for Computing      **
**        Either Displacement, Velocity or Acceleration of an         **
**        Oscillator " , Earthquake Engineering and Structural        **
**        Dynamics, Vol.16, P.245-253, 1988 )                         **
**    ------------------------------------------------------------    **
**    S(T,1:NumV-2) = response of 'forced vibration' due to ground    **
**                    accel,veloc,displ with X(To)=V(To)=0            **
**    S(T,NumV-1)   = response of 'free vibration' with X(To)=1       **
**    S(T,NumV)     = response of 'free vibration' with V(To)=1       **
**    ............................................................    **
**    Omg,Ksi        = natural frequency & damping ratio of oscil.    **
**    G(T,NumG,NumF) = recorded Ground support Accel,Veloc,Displ      **
**                   = Ag(T,NumG), Vg(T,NumG), Xg(T,NumG)             **
**    NumV           = Number of response Variables  = NumG*NumF+2    **
**  ================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumF,NumG,Ti,Tf,NumV , F,I,T,V
      REAL*8     Omg,Ksi,G(MaxT,MaxG,MaxF),S(MaxT,MaxV),dT/0.02D0/
      REAL*8     A1,A2,A3,A4,B1,B2,B3,OmgT,KsiT,SKsiT,e,eSin,eCos
      COMMON  /F[]/   NumF
      COMMON  /G[]/   NumG
      COMMON  /Tif/   Ti,Tf
      COMMON  /V[]/   NumV
**  ================================================================  **
**    +------------------------------------------------------------+    **
**    |  COMPUTE  the required repeat constants                    |    **
**    +------------------------------------------------------------+    **
      OmgT = Omg * dT
      KsiT = Ksi * OmgT
      SKsiT = DSQRT(1-Ksi**2) * OmgT
      e = DEXP(-KsiT)
      eSin = e * DSIN(SKsiT) / SKsiT
      eCos = e * DCOS(SKsiT)
**    +------------------------------------------------------------+    **
**    |  COMPUTE  the time independent coefficients As & Bs        |    **
**    +------------------------------------------------------------+    **
      A1 = 2.D0 * eCos
      A2 = - e**2
      A3 = Omg**2 * ( KsiT*eSin - eCos )
      A4 = - Omg * ( (1.D0-2.D0*Ksi**2)*OmgT*eSin + 2.D0*Ksi*eCos )
      B1 = eSin
      B2 = eCos - (1.D0+KsiT)*eSin
      B3 = 1.D0
**    +------------------------------------------------------------+    **
**    |  COMPUTE  responses of forced vibration  S(T,1:NumV-2)     |    **
**    +------------------------------------------------------------+    **
      V = 0
      DO 10  F = 1,NumF
          DO 20  I = 1,NumG
              V = V + 1
              S(Ti,V)   = - B3*G(Ti,I,F)
              S(Ti+1,V) = - B1*G(Ti+1,I,F) - B2*G(Ti,I,F)
          DO 20  T = Ti+2,Tf
              S(T,V) = A1*S(T-1,V)                  + A2*S(T-2,V)
     &               - B1*(G(T,I,F) - 2*G(T-1,I,F) + G(T-2,I,F))
  20      CONTINUE
```

```
**    +-------------------------------------------------------+    **
**    |  UPDATE  Bs  from G=Ag to G=2*Ksi*Omg*Vg and G=Omg^2*Xg  |    **
**    +-------------------------------------------------------+    **
          B1 = B1 * Omg
          B2 = B2 * Omg
          B3 = B3 * Omg
  10  CONTINUE
**    +-------------------------------------------------------+    **
**    |  COMPUTE  responses of free vibration  S(T,NumV-1:NumV)  |    **
**    +-------------------------------------------------------+    **
      S(Ti,V+1) = - Omg**2
      S(Ti,V+2) = - 2.D0*Ksi*Omg
      S(Ti+1,V+1) = A3
      S(Ti+1,V+2) = A4
      DO 30  T = Ti+2,Tf
          S(T,V+1) = A1*S(T-1,V+1) + A2*S(T-2,V+1)
          S(T,V+2) = A1*S(T-1,V+2) + A2*S(T-2,V+2)
  30  CONTINUE
      NumV = V + 2
      RETURN
      END
************************************************************************
      SUBROUTINE  SolveSystem ( A,B,X,N )
**  ==================================================================  **
**    SOLVE  system of N linear equations  A(N,N)*X(N)=B(N)  of X     **
**    ..................................................    **
**    LDLTDec : Lower-Diagonal-Lower-Transpose Decomposition for A    **
**              A = LDLT    AX = B    ->    LDLTX = B                 **
**    FDBSub  : Forward-Diagonal-Backward Substitution to solve X     **
**              LTX = Y    DY = Z    LZ = B                          **
**    ------------------------------------------------------    **
**    A    : must be a symmetric positive definite square matrix      **
**    MaxV = Maximum dimension of matrix A                    >= N    **
**  ==================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    N
      REAL*8     A(MaxV,MaxV),B(MaxV),X(MaxV)
      REAL*8     L(MaxV,MaxV),D(MaxV)
**  ==================================================================  **
      DO 10 J=1,N
      X(J)=B(J)
      DO 10 I=1,J
   10 L(I,J)=A(I,J)
      CALL  CBDECP (L,N,N,MaxV)
      CALL  CBSOLX (L,X,N,N,MaxV)
      RETURN
      END
************************************************************************
      SUBROUTINE  OutputResult
**  ==================================================================  **
**    WRITE  Ar,Ac    WRITE  Omg,Ksi,Phi                              **
**    ------------------------------------------------------------    **
**    Ar(T,NumS)     = Recorded relative Acceleration                 **
**    Ac(T,NumS)     = Computed relative Acceleration = Sum(Am*Phi)   **
**    Omg(NumM)      = computed optimal modal frequencies             **
**    Ksi(NumM)      = computed optimal modal damping coefficients    **
**    Phi(NumS,NumM) = computed optimal mode shapes                   **
**  ==================================================================  **
$     INCLUDE : 'BI3D1.Inc'
      INTEGER    NumM,NumS,Ti,Tf , Mode,Stru,Time
```

```
      REAL*8     Ar(MaxT,MaxS),Am(MaxT,MaxM),Ac
      REAL*8     Omg(MaxM),Ksi(MaxM),Phi(MaxS,MaxM)
      COMMON  /M[]/  NumM
      COMMON  /S[]/  NumS
      COMMON  /Tif/  Ti,Tf
      COMMON  /Ar[]/ Ar
      COMMON  /Am[]/ Am
      COMMON  /X[]/  Omg,Ksi
      COMMON  /Y[]/  Phi
**  ================================================================  **
**    +-------------------------------------------------------+     **
**    |  WRITE  Ar(T,NumS)                                    |     **
**    |         Ac(T,NumS) = Sum(Am(T,NumM)*Phi(NumS,NumM))   |     **
**    +-------------------------------------------------------+     **
      OPEN (4,FILE='BI3D1.Res',FORM='FORMATTED')
      DO 10  Stru = 1,NumS
      DO 10  Time = Ti,Tf
          Ac = 0.D0
          DO 20  Mode = 1,NumM
              Ac = Ac + Am(Time,Mode)*Phi(Stru,Mode)
  20      CONTINUE
          WRITE(4,110)  Stru,Time,Ar(Time,Stru),Ac
 110      FORMAT(2I5,1P2E12.4)
  10  CONTINUE
**    +-------------------------------------------------------+     **
**    |  WRITE  Omg(NumM),Ksi(NumM),Phi(NumS,NumM)            |     **
**    +-------------------------------------------------------+     **
      OPEN (2,FILE='BI3D1.Out',FORM='FORMATTED')
      DO 30  Mode = 1,NumM
          WRITE(2,120)  Mode,Omg(Mode),Mode,Ksi(Mode)
 120      FORMAT(2X,'Omg(',I1,') =',F12.6,10X,'Ksi(',I1,') =',F12.6)
      DO 30  Stru = 1,NumS
          WRITE(2,130)  Stru,Mode,Phi(Stru,Mode)
 130      FORMAT(2X,'Phi(',I2,',',I1,') =',F12.6)
  30  CONTINUE
      RETURN
      END
**********************************************************************
**    BI3D1.Inc                                                     **
**********************************************************************
**    INCLUDE FILE  for  BI3D1.For                                  **
**  ================================================================  **
**    DEFINE  Maximum dimensions : MaxM,MaxF,MaxG,MaxS,MaxT,MaxV    **
**            for MAIN PROGRAM and SUBROUTINEs of BI3D1.For         **
**    BI3D1.Dat  must has NumT*NumF*NumG+NumT*NumS lines of data    **
**    ------------------------------------------------------------  **
**    MaxM = Maximum number of Modes can be analyized      >= NumM  **
**    MaxF = Maximum number of support Forcing inputs      >= NumF  **
**    MaxG = Maximum number of Ground support DOFs         >= NumG  **
**    MaxS = Maximum number of superStructure DOFs         >= NumS  **
**    MaxT = Maximum number of Time steps in Ag and At     >= NumT  **
**    MaxV = Maximum number of Variables for solving Am    >= NumV  **
**  ================================================================  **
      PARAMETER  (
     &    MaxM = 5,
     &    MaxF = 3,
     &    MaxG = 9,
     &    MaxS = 15,
     &    MaxT = 2000,
     &    MaxV = MaxG * MaxF + 2  )
```

****

參考文獻

1. Beck, J. L., "Determining Models of Structures from Earthquake Records", Doctoral Thesis, California Institute of Technology, Pasadena, CA, 1978.

2. Beck, J. L., and Dowling, M. J., "Quick Algorithms for Computing Either Displacement, Velocity or Acceleration of an Oscillator", Earthquake Engineering and Structural Dynamics, Vol. 16, 1988.

3. Beck, J. L., and Jennings, P. C., "Structural Identification Using Linear Models and Earthquake Records", Earthquake Engineering and Structural Dynamics, Vol. 8, 1979.

4. Chopra, A. K., Dynamics of Structures : Theory and Applications to Earthquake Engineering, Prentice Hall, NJ, 1995.

5. Clough, R. W., and Penzien, J., Dynamics of Structures, McGraw-Hill, NY, 1975.

6. Lin, T. W., "Well Behaved Penalty Function for Constrained Optimization", Journal of the Chinese Institute of Engineers, Vol. 13, No. 2, 1990.

7. Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., Numerical Recipes - The Art of Scientific Computing, Cambridge University Press, NY, 1992.

第二十三章

網路獨立迴路之建立

23.1 前言

本章介紹以計算機尋求網路獨立迴路之有效方法與程式。在管線網路分析模式中，節點法在求解過程中對節點水頭（即流體在節點處之壓力）差值趨近於零之管線造成不穩定之數值問題，而迴路法則較為穩定有效。以迴路法分析時，建立迴路為重要步驟，因此本章特別介紹一種尋求網路獨立迴路之方法。利用網路組成之特牲，將網路以逐漸加入管線之方式組成該網路以尋求獨立迴路，即在組成網路過程中，尋求互相連接之管線，將其頭尾節點與已組成之獨立網路相連接，則新加入管線及已組成之網路中之管線所連接成之新迴路必與已組成之網路中之各迴路互相獨立。在尋找獨立迴路之過程中，利用標記與掃描的技巧，可有效的尋找出新的獨立迴路。

許多工程問題經常以網路模式表示，如管線網路，交通網路，電力網路等。構成網路之基本元素為節點 (node) 與弧 (arc)，節點與弧在各種問題裏具有不同之意義，例如在交通網路裏節點代表轉運站，弧代表轉運站間之聯絡道路，在管線網路裏代表自來水或其它流體管線，節點代表管線之交點或儲蓄池，幫浦之位置。管線網路分析模式可分為二類主要分析方法：一為節點法；一為迴路法。節點法以網路之節點水頭為未知數，利用流量平衡（即流入與流出節點之水量相等）為等式進行求解，但因流量平衡式之一次導式含有水頭差值在分母之情況，故在求解過程中對水頭差值趨近於零之管線將造成不穩定之數值問題。迴路法則以管線流量為未知數，再以流量平衡式 及迴路水頭平衡（即迴路中管線二端點之水頭差之和為零）為等式求解，此方法較為穩定有效，故一般較為分析者所採

用。早期Kesavan等利用圖形理論 (graph theory) 尋找迴路：首先找出一個樹狀圖形，其節點為網路之全部節點，其管線數等於節點數減一，任何節點間只有一條通路，亦即無迴路存在於樹狀圖形，全網路中屬於此樹狀圖形之網路之管線之集合稱樹集 (tree)，其餘不屬於樹狀網路之管線之集合稱餘樹集 (cotree)。餘樹集內之管線數等於全網路之迴路數，取餘樹集內之某一管線與樹集內之管線可得一迴路稱基本迴路。因這些基本迴路各僅含一餘樹集內之管線，故均為獨立之迴路。此法之最大缺點為所得迴路路徑較長，即包含之管線較多，這將使網路分析所需之運算量增加。本章將介紹另一方法以尋找迴路，其效率不遜於以樹集與餘樹集尋找迴路之方法，且所得迴路路徑較短，因其允許新迴路經過已組成迴路之所有管線，相當於每一迴路不限於僅路徑樹集內之管線。Nielsen則以圖形理論利用矩陣操作直接求出迴路水頭平衡式之係數矩陣。以下先略為介紹該法，並附帶介紹有關之平衡式(以下說明可略過)。

將流量平衡式 (即流入與流出節點之水量相等) 寫為：

$$\sum_{i=1}^{N_p} a_{ij} q_i = Q_j \ , \qquad j = 1, 2, \ldots, N_n - 1 \tag{23.1}$$

式中i為管線編號，j為節點編號，N_p為管線總數，N_n為節點總數，a_{ij}係數有1，-1，0三種值，若i管線之正向流流入j節點，則$a_{ij}=1$；若i管線之正向流流出j節點，則$a_{ij}=-1$；若i管線與j節點不相連，則$a_{ij}=0$。q_i為i管線之流量，Q_j為j節點之流出水量，前式可寫成矩陣式如下：

$$[A]^T \{q\} = \{Q\} \tag{23.2}$$

式中$[A]$為$N_p \times (N_n - 1)$矩陣，$\{q\}$為N_p元向量，$\{Q\}$為$(N_n - 1)$元向量。將流量$\{q\}$分為兩種流量之和：

$$\{q\} = \{q_c\} + [C]\{u\} = [\ I\ \mid\ C\ ] \left\{ \begin{array}{c} q_c \\ u \end{array} \right\} \tag{23.3}$$

且$[C]$矩陣滿足下列二式

$$[A]^T [C] = 0 \tag{23.4}$$

$$[C] \neq 0 \tag{23.5}$$

式中$\{q_c\}$為式(23.2)之任意解，$\{u\}$為$(N_p - N_n + 1)$元向量，為迴路之流量，$[C]$為$N_p \times (N_p - N_n + 1)$矩陣，則$[C]$矩陣為迴路水頭平衡式之係數矩陣，

$[C]$矩陣可由下列過程找出：首先對$[A]^T$矩陣作行調換，即：

$$[A]^T[P] = [\ F\ \mid\ G\ ] \tag{23.6}$$

使$[F]$為$(N_n-1)\times(N_n-1)$之非奇異方陣（例如$[F]$之各行可選為對應於Kesavan之樹集內之管線i之係數a_{ij}），再令

$$\{q_c\} = [P]\begin{Bmatrix} [F]^{-1}\{Q\} \\ \{0\} \end{Bmatrix} \tag{23.7}$$

$$[C] = [P]\begin{bmatrix} -[F]^{-1}[G] \\ [I] \end{bmatrix} \tag{23.8}$$

則$[C]$矩陣即為閉迴路係數矩陣，其元素有0，-1，1三種值。

Nielsen法佔用之程式記憶空間為$(N_n-1)\times N_p$，對大型網路而言，管線數目可以大至數百或數千，使用較為不便，下節即介紹另一種建立獨立迴路之法。

23.2 建立獨立迴路之原理

由管線相連形成之網路具有下式基本特性：

$$N_l = N_p - N_n + 1 \tag{23.9}$$

其中N_l為獨立閉迴路(closed loop)之總個數，N_p為管線個數，N_n為節點個數。

任何網路既為管線及節點所組成，必可將網路以逐漸加入新管線及新節點之方式組成該網路。在組成網路之過程中，加入新管線及新節點之方式是尋找一串相連接之新管線及其間新節點，其頭尾節點再與已組成之網路中之舊管線及其間舊節點相連而構成一新迴路。注意該串新管線不得單獨構成迴路，且新管線之節點須為新節點，僅該串新管線之頭尾兩端節點為舊節點。則：(1)新加入之管線及已組成之網路中之管線所連接成之新迴路必與已組成之網路中之各獨立迴路互相獨立；(2)組成第一個迴路之管線個數與節點個數相等；(3)新增一迴路時，其新加入之新管線個數必較新節點個數多1。故在逐次尋得新迴路之各階段，其已組成之網路組中之獨立迴路個數，管線個數及節點個數必符合下式：

$$N_{l,i} = N_{p,i} - N_{n,i} + 1 \tag{23.10}$$

其中$N_{l,i}$，$N_{p,i}$，$N_{n,i}$分別為第i階段之迴路個數，管線個數及節點個數。

比較式(23.9)與式(23.10)二式亦可知當有新的迴路產生時，所須新增加之管線個數為新增加節點個數加1。且因新增加之迴路具有新加入之管線，故該迴路與上一階段之所有迴路之關係必屬獨立迴路(或稱不相依迴路)。

23.3 建立獨立迴路之技巧

為說明方便，首先定義一些術語：在加入新管線以組成網路而尋找獨立迴路之過程中，當管線在前階段已加入網路中，稱其為已使用過之管線(used pipe)，稱該管線之兩端節點為已使用過之節點(used node)，當節點之相鄰管線均未使用過，稱其為未使用過之節點(即節點尚未加入網路中)，當節點之相鄰管線均已使用過，稱其為完全使用過之節點(即節點之所有相鄰管線均已加入網路中)，若節點之部分相鄰管線已使用過，稱其為未完全使用過之節點。稱待尋找之新迴路起始點為出發節點(start node)，若以出發節點之某相鄰管線開始尋找獨立迴路，稱該管線為出發管線 (start pipe)，稱出發管線之相對於出發節點之另端節點為根節點 (root node)。

建立獨立迴路之技巧可利用網路學裏的掃描(scan)與標記 (label)。在網路學裏有多種問題可利用掃描與標記的技巧，例如最短路徑，最長排程，最大流量等(第十九章亦利用此技巧)。基本上，當甲節點欲將訊息傳入乙節點時，必須透過相連之管線，因此透過管線以尋找節點，稱為掃描。其相鄰節點合於條件者則被標記，稱甲節點為掃描節點，乙節點為被標記節點。被標記之節點並依序列入一稱為標記節點序列之集合裏，並等待做為掃描節點。任何時候節點之狀態必為下列三種之一：(1)為未標記，(2)為已標記待掃描，(3)為已標記已掃描。若有必要，亦可將已標記已掃描之節點(第三種狀態)重新列入已標記待掃描(第二種狀態)之狀態，如最短，最長排程。雖然各種問題所使用之標記條件不一，但其精神則大同小異。應用以上所述方法尋找獨立迴路時，當某節點被相鄰之掃瞄節點掃描而標記，即表示可能經由掃瞄節點至被標記節點之管線而構成新迴路，因此在掃描與標記之過程中，被標記節點須符合下列條件：(1)若掃描節點為已使用過，所經過之管線亦須為已使用過(即只能透過已使用過管線掃瞄)。(2)被標記節點原為未標記節點。

在開始尋找獨立迴路之前，首先將網路中所有節點及管線均列為未使用過之節點及管線，並任選一節點為出發節點，其相鄰未使用過之管線為出發管線，出發管線之另一端節點為根節點，之後重覆下述尋找新迴路之過程：

(1) 首先將所有節點列為未標記節點，將根節點標記後列於標記節點序列之首。之後重覆下述動作直至尋得新的迴路或該序列裏沒有待掃描節點存在為止：

(1a) 自標記節點序列中依序取出一待掃描節點做為掃描節點，以該掃描節點透過其相鄰管線掃瞄其周圍相鄰之節點，被掃描之節點若合乎上述標記之條件則被標記，被標記之節點若為出發節點，則尋得獨立的新迴路，續做步驟(2a)。

(1b) 否則，將此新標記之節點列入標記節點序列裏，並記錄此新標記節點之掃描節點及經過之管線號碼以供反向退回迴路之出發節點之用。續做步驟(1a)。

(2a) 若尋得新迴路，則由出發節點依所紀錄之管線號碼逐次反向退回經過之節點至出發節點，所經管線即為新迴路之管線。將該迴路之所有管線列為使用過。續做步驟(3)。

(2b) 若標記節點序列裏沒有待掃描節點存在，表示從出發管線出發無法尋得新迴路。將出發管線及兩端節點列為使用過。續做步驟(3)。

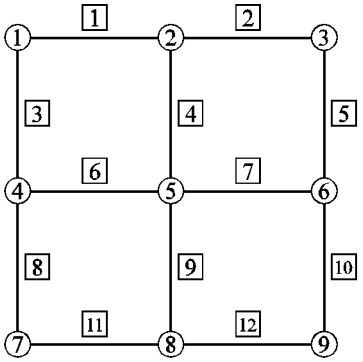
(3) 重新選定一未完全使用過之節點作為出發節點，重覆步驟(1)以尋找新迴路；若所有節點均為完全使用過，則已尋得一相連網路之所有迴路。續做步驟(4)。

(4) 若網路為多區不相連網路，則於一區管線及節點均已列為完全使用過之節點後，須再尋找他區之未使用過節點做為出發節點，並選定出發管線與根節點後重覆步驟(1)。

開迴路 (open loop) 亦可用下列方式求得：虛設一管線於開迴路之頭尾節點，當尋找其它迴路時，禁止通過此虛設管線，以此虛設管線為出發管線，所尋得之迴路去除虛設管線，即為所需之開迴路。

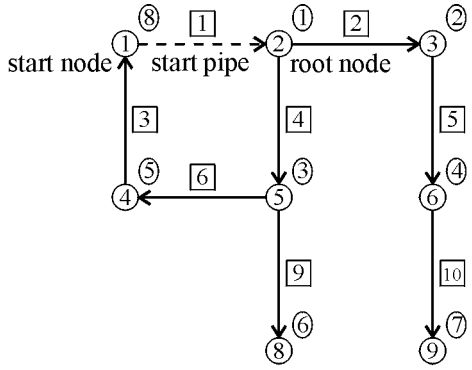
23.4 範例

圖一之網路有9個節點，12支管線，4個獨立迴路，以圖二(a)至(d)依序標示尋找4個獨立迴路之過程。

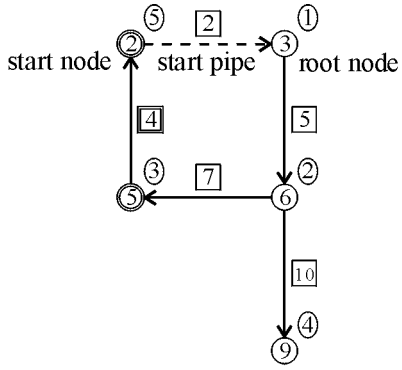


○：節點編號(未使用)
□：管線編號(未使用)
◎：節點編號(已使用)
▣：管線編號(已使用)
○：節點之標記順序

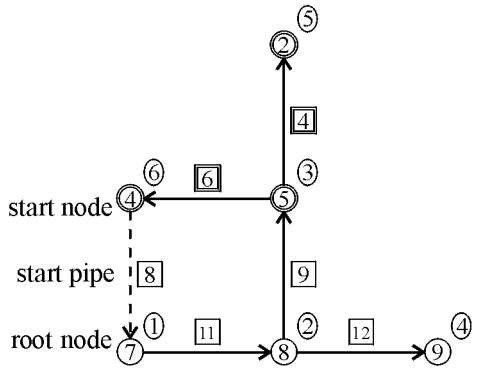
圖一 節點與管線編號



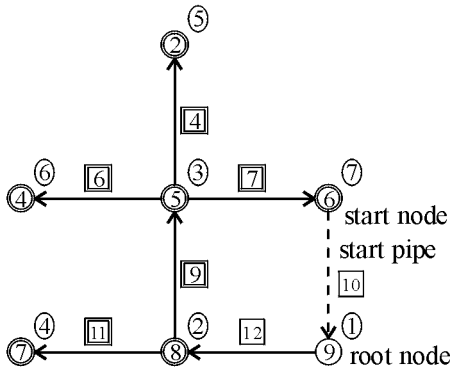
(a) LOOP 1



(b) LOOP 2



(c) LOOP 3



(d) LOOP 4

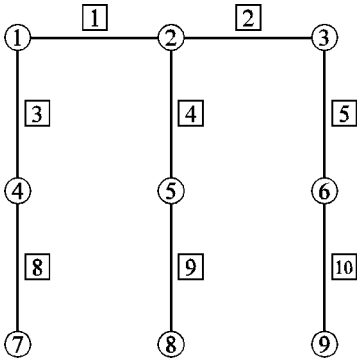
圖二 迴路之標記順序

在圖二(a)中，節點1為出發節點，管線1為出發管線，節點2為根節點，節點之被標記順序為(2, 3, 5, 6, 4, 8, 9, 1)。在找到出發節點1時，即可確定尋得新的獨立迴路，之後以節點1依反向路徑可得該迴路之管線號碼，依序為(3, 6, 4, 1)，之後將迴路上之管線列為已使用過。

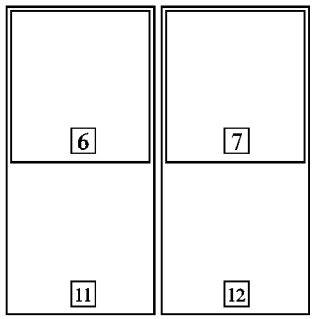
在圖二(b)中，節點2為出發節點，管線2為出發管線，節點3為根節點，節點之被標記順序為(3, 6, 5, 9, 2)。注意當以節點5為掃描節點時，因節點5為已使用過之節點，故僅能透過已使用過之管線4掃瞄而標記節點2，而不能透過未使用過之管線9掃瞄到節點8，其餘之作法同上述。

圖二(c)及圖二(d)分別尋找另外二個獨立迴路，之後因所有管線均已列為已使用過，表示全部獨立迴路均已找得。

若以Kesavan法取圖三之樹集，則所得之各獨立迴路之管線號碼為(6, 4, 1, 3)，(7, 5, 2, 4)，(11, 9, 4, 1, 3, 8)，(12, 10, 5, 2, 4, 9)，見圖四。其迴路之管線數目明顯的比圖二所得迴路之管線數目多。事實上由圖二所得之獨立迴路幾乎為最短迴路。



圖三 樹集管線與節點



圖四 餘樹集管線與基本迴路

23.5 建立獨立迴路之程式

本文提供上節方法所建立之程式，程式中各變數意義如下：NP，NN，NL分別代表網路管線數目，節點數目，迴路數目，NOPP(1:2,I)分別為管線之頭尾節點編號。指標PONP(J)之內容為使K=PONP(J)+I–1值對應到PONS(K)之儲存順序號碼K，即以PONS(PONP(J)+I–1)儲存節點J周圍之第I相鄰管線號碼。指標POLP(L)之內容為使K=POLP(L)+I–1值對應到POLS(K)之儲存順序號碼K，即以POLS(POLP(L)+I–1)儲存迴路L所屬之第I相連管線號碼。以前節之網路為例，上述各值如下列所示：

管線號碼J	1	2	3	4	5	6	7	8	9	10	11	12
NOPP(1,J)	1	2	1	2	3	4	5	4	5	6	7	8
NOPP(2,J)	2	3	4	5	6	5	6	7	8	9	8	9

節點號碼J	管線號碼	管線數	PONP(J)	
1	–1 –3	2	1	
2	1 –2 –4	3	3	=2+1
3	2 –5	2	6	=3+3
4	3 –6 –8	3	8	=2+6
5	4 6 –7 –9	4	11	=3+8
6	5 7 –10	3	15	=4+11
7	8 –11	2	18	=3+15
8	9 11 –12	3	20	=2+18
9	10 12	2	23	=3+20
			25	=2+23

$PONS(K)$	–1	–3	1	–2	–4	2	–5	3	–6	–8
K	1	2	3	4	5	6	7	8	9	10

4	6	–7	–9	5	7	–10	8	–11	9	11	–12	10	12	
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

迴路號碼L	管線號碼	管線數	POLP(L)	
1	3 6 –4 –1	4	1	
2	4 7 –5 –2	4	5	=4+1
3	6 9 –11 –8	4	9	=4+5
4	–7 9 12 –10	4	13	=4+9
			17	=4+13

$POLS(K)$	3	6	–4	–1	4	7	–5	–2
K	1	2	3	4	5	6	7	8

6	9	–11	–8	–7	9	12	–10	
9	10	11	12	13	14	15	16	17

程式另外以PONU(J)記錄J節點之使用狀況：PONU(J)=PONP(J+1)表示節點未使用；PONP(J)<PONU(J)<PONP(J+1)表示節點已使用但未完全使用；PONU(J)=PONP(J)表示節點已完全使用。為了節省程式記憶空間，以PONP(J)取負號代表J節點已標記，以NOPP(1,IP)取負號代表管線IP已使用，以NOPP(2,IP)取負號代表虛設管線。以LVSN(1:JL)序列儲存被標記節點號碼，其第1至IL–1序號內之節點為已掃描節點，第IL序號之節點為掃描節點，第IL+1至JL序號節點為待掃描節點。對應位置之LVSB(JL)及LVSP(JL)分別記錄使節點LVSN(JL)被標記之掃描節點的序號及所透過之管線之號碼。以IR代表出發節點，IP代表出發管線，JR代表根節點，以下為程式之處理步驟：

(1)形成PONP(1:NP)，PONS(*)，PONU(1:NN)，令出發節點IR等於1。

(2)由出發節點IR選出發管線IP及根節點JR。

(3)標記出發管線之根節點JR，令IL=1，將根節點存入標記節點序列之第1位置，即令JL=1，儲存被標記節點號碼及對應資料LVSN(JL)=JR，LVSP(JL)=IP，LVSB(JL)=0。

(4)取出標記節點序列中之第IL序號之節點IN=LVSN(IL)做為掃描節點，依序透過其相鄰管線JP以掃瞄節點JN。若某節點JN合於標記條件，則繼續步驟(5)。若節點IN之掃瞄節點均不合標記條件，則跳至步驟(6)。

(5)標記節點JN，即令JL=JL+1，儲存標記序列資料LVSN(JL)=JN，LVSP(JL)=JP，LVSB(JL)=IL，若節點JN為出發節點，則尋得新的獨立迴路，跳至步驟(7)，否則繼續步驟(6)。

(6)令IL=IL+1，若IL≤JL則跳回步驟4；若IL>JL則表示標記序列中已無待掃描節點，令節點IR，JR及管線IP為已使用過，跳至步驟(8)。

(7)由LVSN，LVSP，LVSB之資料找出新獨立迴路之節點及管線，並令其為已使用過，繼續步驟(8)。

(8)尋找同一區網路之未使用過之節點做為新的出發節點IR，尋得則跳回步驟(2)；否則繼續步驟(9)。

(9)尋找另一區網路之未使用過之節點做為新的出發節點IR，尋得則跳回步驟(2)；否則結束本程式。

本程式因尋找迴路所需之記憶空間為$4 * N_p$，較Nielsen所提出之方法所需之$N_p * (N_n - 1)$大為節省，並曾用於臺北市之自來水管網分析中，其$N_p = 448$, $N_n = 239$, $N_l = 210$。

［表一］ 建立獨立迴路之副程式

```
************************************************************************
      SUBROUTINE MKLOOP(NOPP,PONP,PONS,POLP,POLS
     *                 ,PONU,LVSN,LVSP,LVSB,NP,NN,NL)
C**   ================================================================    **
      INTEGER NOPP(2,NP),PONP(1),PONS(1),POLP(1),POLS(1)
      INTEGER PONU(NN),LVSN(NN),LVSP(NN),LVSB(NN)
C**   ================================================================    **
C**   Input  : NOPP(2,NP),NP,NN                                            **
C**   Oouput : PONP(NN+1),PONS(2*NP),POLP(NL+1),POLS(POLP(NL+1)-1),NL **
C**   Working: PONU(NN),LVSN(NN),LVSP(NN),LVSB(NN)                       **
C**   ----------------------------------------------------------------    **
C**   NP,NN,NL   = Numbers of pipes, nodes, loops                         **
C**   NOPP(2,IP) = Positive pipe flow                                     **
C**                from node NOPP(1,IP) to NOPP(2,IP)                     **
C**   PONP,PONS  = Pipe list of nodes                                     **
C**   POLP,POLS  = Pipe list of loops                                     **
C**   PONU(IN)   = Index for the usage of pipes connected to node IN  **
C**              = PONP(IN+1) : No   pipe  used : unused node         **
C**              = PONP(IN)   : All  pipes used : used   node         **
C**              = In between : Some pipes used : used   node         **
C**   PONP(IN)   < 0 : To mark node IN as labeled                         **
C**   NOPP(1,IP) < 0 : To mark pipe IP as used                            **
C**   NOPP(2,IP) < 0 : Input value for a dummy pipe btwn 2 reservoirs **
C**                    It is used in the only loop rooted by itself   **
C**   LVSN,LVSP,LVSB = Node, pipe, backward_pntr in level structure   **
C**   ----------------------------------------------------------------    **
C**   Connected pipes of node IN : PONS( PONP(IN) : PONP(IN+1)-1) )   **
C**   Connected pipes of loop IL : POLS( POLP(IL) : POLP(IL+1)-1) )   **
C**   ================================================================    **
C**   +--------------------------------------------------------------+    **
C**   | Count the no. of pipes connecting to node IN : PONU(IN)  |    **
C*1   +--------------------------------------------------------------+    **
      DO 100 IN=1,NN
  100 PONU(IN)=0
      DO 120 IP=1,NP
      DO 120 K=1,2
      IN=IABS(NOPP(K,IP))
  120 PONU(IN)=PONU(IN)+1
C**   +--------------------------------------------------------------+    **
C**   | Setup pointers to the pipe list for the nodes            |    **
C**   +--------------------------------------------------------------+    **
      PONP(1)=1
      DO 140 IN=1,NN
      PONP(IN+1)=PONP(IN)+PONU(IN)
  140 PONU(IN)=PONP(IN)
C**   +--------------------------------------------------------------+    **
C**   | Setup pipe list of nodes                                 |    **
C**   +--------------------------------------------------------------+    **
      DO 160 IP=1,NP
      DO 160 K=1,2
      IN=IABS(NOPP(K,IP))
      IT=PONU(IN)
      PONS(IT)=IP*(2*K-3)
  160 PONU(IN)=IT+1
C**   +--------------------------------------------------------------+    **
C**   | Initialize a loop array                                  |    **
C**   +--------------------------------------------------------------+    **
      NL=0
```

```
      POLP(NL+1)=1
      ML=NP-NN+1
C**   +--------------------------------------------------------+    **
C**   | Setup a level structure for an unused pipe: NOPP(1,IP)>0 |    **
C**   | The pipe must has at least one  used  node: node IR      |    **
C**   | Node IR is a used node but not labeled as a root node    |    **
C*2   +--------------------------------------------------------+    **
      IR=1
  200 IBG=PONP(IR)
      IED=PONP(IR+1)-1
      DO 700 IT=IBG,IED
      IP=IABS(PONS(IT))
      IF(NOPP(1,IP).LE.0) GO TO 700      ! Skip used pipes
C**   +--------------------------------------------------------+    **
C**   | Make IP as a root pipe and JR as a root node : JL = 1    |    **
C**   | Mark node JR as labeled : PONP(JR) < 0                   |    **
C*3   +--------------------------------------------------------+    **
      JR=NOPP(1,IP)+IABS(NOPP(2,IP))-IR
      PONP(JR)=-PONP(JR)
      JL=1
      LVSN(JL)=JR
      LVSP(JL)=PONS(IT)
      LVSB(JL)=0
      IL=1
C*4   ----------------------------------------------------------    **
  300 IN=LVSN(IL)
      JBG=IABS(PONP(IN))
      JED=IABS(PONP(IN+1))-1
      DO 400 JT=JBG,JED
      JP=IABS(PONS(JT))
C**   +--------------------------------------------------------+    **
C**   | Never go through unused pipe from used node              |    **
C**   +--------------------------------------------------------+    **
C**   | PONU(IN)=PONP(IN+1): unused node: OK for all unused pipe |    **
C**   | PONU(IN)=PONP(IN)  : used   node: OK for all  used  pipe |    **
C**   | PONU(IN)=In between: used   node: OK for used pipes only |    **
C**   +--------------------------------------------------------+    **
      IF(PONU(IN).LE.JED.AND.NOPP(1,JP).GT.0) GO TO 400
      IF(NOPP(2,JP).LT.0) GO TO 400
      JN=IABS(NOPP(1,JP))+NOPP(2,JP)-IN
      IF(PONP(JN).LT.0) GO TO 400
      IF(JP.EQ.IP) GO TO 400
C*5   ----------------------------------------------------------    **
      PONP(JN)=-PONP(JN)
      JL=JL+1
      LVSN(JL)=JN
      LVSP(JL)=PONS(JT)
      LVSB(JL)=IL
      IF(JN.EQ.IR) GO TO 500
  400 CONTINUE
C*6   ----------------------------------------------------------    **
      IL=IL+1
      IF(IL.LE.JL) GO TO 300
C**   +--------------------------------------------------------+    **
C**   | Reset index for labeled nodes                            |    **
C*6   | Always mark the root pipe and its end nodes as used      |    **
C*7   +--------------------------------------------------------+    **
  500 DO 550 IL=1,JL
      IN=LVSN(IL)
  550 PONP(IN)=IABS(PONP(IN))
```

```
C**   -------------------------------------------------------------    **
      PONU(IR)=PONU(IR)-1
      PONU(JR)=PONU(JR)-1
      NOPP(1,IP)=-NOPP(1,IP)
C**   -------------------------------------------------------------    **
      IF(IN.NE.IR) GO TO 700
C**   +-----------------------------------------------------------+    **
C**   | Trace back the level structure for the pipes of new loop |    **
C*7   +-----------------------------------------------------------+    **
      NL=NL+1
      KT=POLP(NL)
C**   -------------------------------------------------------------    **
  600 JP=IABS(LVSP(JL))
      IF(NOPP(1,JP).GT.0) THEN
         IN=NOPP(1,JP)
         JN=NOPP(2,JP)
         PONU(IN)=PONU(IN)-1
         PONU(JN)=PONU(JN)-1
         NOPP(1,JP)=-IN
      ENDIF
      POLS(KT)=LVSP(JL)
      KT=KT+1
      JL=LVSB(JL)
      IF(JL.GT.0) GO TO 600
C**   -------------------------------------------------------------    **
      POLP(NL+1)=KT
  700 CONTINUE
C**   +-----------------------------------------------------------+    **
C**   | Find a used node which has at least one unused pipe : IR |    **
C**   |   or an unused node for another separate district   : IY |    **
C**   +-----------------------------------------------------------+    **
C**   | PONU(IR)=PONP(IR+1): unused node: NG for all unused pipe |    **
C**   | PONU(IR)=PONP(IR)  : used   node: NG for all  used  pipe |    **
C*8   | PONU(IR)=In between: used   node: OK for any unused pipe |    **
C*9   +-----------------------------------------------------------+    **
      IX=IR
      IY=NN+1
  750 IR=IR+1
      IF(IR.GT.NN) IR=1
      IF(IR.EQ.IX) THEN
         IF(IY.GT.NN) GO TO 900
         ML=ML+1
         IR=IY
         GO TO 200
      ENDIF
      IF(PONU(IR).EQ.PONP(IR)) GO TO 750
      IF(PONU(IR).NE.PONP(IR+1)) GO TO 200
      IY=MIN0(IR,IY)
      GO TO 750
C**   -------------------------------------------------------------    **
  900 IF(NL.NE.ML) WRITE(*,*) ' ERROR IN MKLOOP '
      DO 950 IP=1,NP
  950 NOPP(1,IP)=IABS(NOPP(1,IP))
      RETURN
      END
*************************************************************************
```

［表二］ 建立獨立迴路之試用程式

```
************************************************************************
      PROGRAM MKMAIN
C**   ================================================================    **
      INTEGER A(10000)
      NDIM=10000
C**   ================================================================    **
  100 READ (*,'(/I5,2I5)') NP,NN,ND
      IF(NP.EQ.0.OR.NN.EQ.0) STOP
      IF(ND.LE.0) ND=1
      WRITE(*,'(I6)')
      NL=NP-NN+ND
C**   ----------------------------------------------------------------    **
      N1=1
      N2=N1+2*NP        ! NOPP(2,NP)
      N3=N2+NN+1        ! PONP(NN+1)
      N4=N3+2*NP        ! PONS(2*NP)
      N5=N4+NL+1        ! POLP(NL+1)
      N6=NDIM-4*NN+1    ! POLS(*) unknown
      N7=N6+NN          ! PONU(NN)
      N8=N7+NN          ! LVSN(NN)
      N9=N8+NN          ! LVSP(NN)
      NA=N9+NN          ! LVSB(NN)
C**   ----------------------------------------------------------------    **
      CALL MKINP (A(N1),NP)
      CALL MKLOOP(A(N1),A(N2),A(N3),A(N4),A(N5)
     *           ,A(N6),A(N7),A(N8),A(N9),NP,NN,NL)
      CALL MKOUT (A(N2),A(N3),A(N4),A(N5),NN,NL)
      IF(N5+A(N4+NL).GT.N6) WRITE(*,*) ' ARRAY TOO SMALL '
      GO TO 100
      END
************************************************************************
      SUBROUTINE MKINP(NOPP,NP)
C**   ================================================================    **
      INTEGER NOPP(2,NP)
C**   ================================================================    **
      DO 200 IP=1,NP
      READ (*,'(5X,2I5)') NOPP(1,IP),NOPP(2,IP)
  200 CONTINUE
      RETURN
      END
************************************************************************
      SUBROUTINE MKOUT(PONP,PONS,POLP,POLS,NN,NL)
C**   ================================================================    **
      INTEGER PONP(1),PONS(1),POLP(1),POLS(1)
C**   ================================================================    **
      DO 100 IN=1,NN
      IPBG=PONP(IN)
      IPED=PONP(IN+1)-1
  100 WRITE(*,'(I6,14I5)') IN,(PONS(IP),IP=IPBG,IPED)
C**   ----------------------------------------------------------------    **
      DO 200 IL=1,NL
      IPBG=POLP(IL)
      IPED=POLP(IL+1)-1
  200 WRITE(*,'(I6,14I5)') IL,(POLS(IP),IP=IPBG,IPED)
      RETURN
      END
************************************************************************
```

參考文獻

1. Chin, K.K., Gay, R.K., Lchua, S.H. and Ho, S.Y., "Solution of Water Networks by Sparse Matrix Methods," International Journal for Numerical Methods in Engineering, Vol.12, pp.1261–1277, 1978.

2. Wood, D.J. and Rayes, A.G., "Reliability of Algorithms for Pipe Network Analysis," Journal of Hydraulic Engineering, ACSE, Vol.107, No.10, pp.1145–1161, 1981.

3. Kesavan, H.K. and Chandrashekar, M., "Graph Theory Models for Pipe Network Analysis," Journal of Hydraulic Engineering, ACSE, Vol.98, pp.345–364, 1972.

4. Nielsen, H.B., "Methods for Analyzing Pipe Networks," Journal of Hydraulic Engineering, ACSE, Vol.115, No.2, pp.139–157, 1989.

5. 林聰悟，鐘銘輝，『建立網路獨立迴路之新方法』，中國土木水利工程學刊，第四卷，第四期，pp.319–324，1992。

索引

五畫

六畫

七畫

八畫

九畫

十畫

十一畫

十二畫

十三畫

十四畫

十五畫

十六畫

十七畫

十八畫

十九畫

二十畫

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

國家圖書館出版品預行編目資料

數值方法與程式/林聰悟，林佳慧著.--初版.
--臺北市：陳紫芬，民86
面：　公分
含參考書目
ISBN 957-97240-3-2 (精裝)

1. 數值分析　2. 工程數學-資料處理

310.107　86012075

中華民國八十八年十月初版再刷

數值方法與程式

定價：800元

著　者：林聰悟　林佳慧
電　話：(02)2363-3214
發行人：陳紫芬
印刷所：圖文技術服務有限公司
地址：台北市辛亥路二段一七一巷三號七樓
電話：(02)2735-0607
傳真：(02)2735-0619
郵政劃撥帳號：01165759
帳　　戶：陳紫芬