National Taiwan University **Finite Element Method**

Department of Civil Engineering Instructors: C.-S. CHEN

Lab Assignment 5, 05/02/2019, 1800 -- 2000

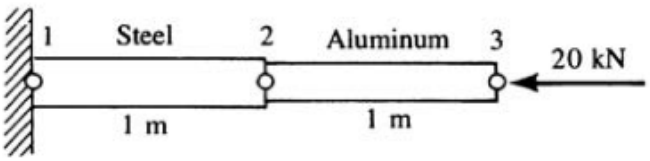
Due 2000

Lab Grading Policy: Attendance 20%, Score 100%

The Lab repeats the exercises you did in Midterm. You can use or modify your Midterm MATLAB codes and are encouraged to complete these exercises by 2000. You should upload them before 2100 on Saturday and no penalty will apply. **No late submission is permitted**. We will in general post the reference solutions by **Sunday**.

Download the `Lab5Dist.zip` **package from the ceiba course website. Once unzipping the file, you will find a folder** `r07XXXXXX`**, four empty sub-folders** `Prob1`**,** `Prob1`**,** `Prob3`**, and** `Prob4` **and a sub-folder** `Supplement`**. Change the folder name to your ID** `(r07XXXXXX to your ID)` **and write your solution directly in the sub-folder for each problem listed below. Consult and use any** `.m` **files in** `Supplement` **as you see proper.**

1. **(20%)** Consider a bar shown below and determine the nodal displacements, the reactions and the force in each element. Be aware of unit consistency.



$E_{st} = 200$ GPa
$A_{st} = 4 \times 10^{-4}$ m^2
$E_{al} = 70$ GPa
$A_{al} = 2 \times 10^{-4}$ m^2

Below are sample outputs:

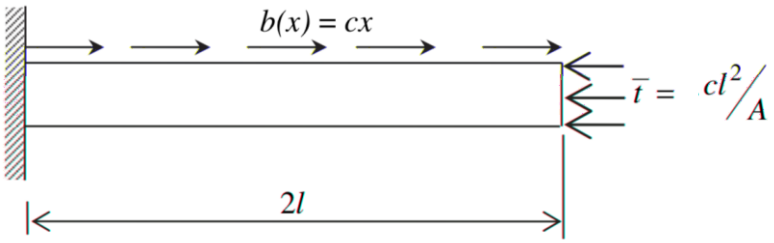
```
Displacements
node      displacements
 1:         0.0000e+00
 2:        -2.5000e-04
 3:        -1.6786e-03
Reactions
node        reactions
 1:         2.0000e+04
Element Forces
element   node_I   node_J     force I       force J
   1        1        2      2.0000e+04   -2.0000e+04
   2        2        3      2.0000e+04   -2.0000e+04
```

```
% 1D Simple Bar
```

```
% clear memory
clear all
%%
% E: modulus of elasticity
% A: area of cross section
% L: length of bar
E=[200e9 70e9]; A=[4e-4 2e-4]; L=[1 1];

% numberElements: number of elements
numberElements=2;
% numberNodes: number of nodes
numberNodes=3;
% generation of coordinates and connectivities
elementNodes=[1 2;2 3];
nodeCoordinates=[0 1 2];
% for structure:
   % displacements: displacement vector
   % force : force vector
   % stiffness: stiffness matrix
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);
% applied load at node 2
force(3)=-20000.0;
%%
% computation of the system stiffness matrix
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  k(e)=E(e)*A(e)/L(e) ;
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+k(e)*[1 -1;-1 1]
end
% boundary conditions and solution
% prescribed dofs
prescribedDof=[1];
% solution
GDof=numberNodes;
displacements=solution(GDof,prescribedDof,stiffness,force);
% output displacements/reactions
outputDisplacementsReactions(displacements,stiffness,GDof,prescribedDof,force,nu
mberElements,elementNodes,E,A,L)

%.............................................................

function outputDisplacementsReactions...

(displacements,stiffness,GDof,prescribedDof,force,numberElements,elementNodes,E,
A,L)

% output of displacements and reactions in
% tabular form

% GDof: total number of degrees of freedom of
% the problem

disp('Displacements')
fprintf('node    displacements\n')
% displacements
for jj=1:GDof
    fprintf('%2.0f:       %10.4e\n', jj, displacements(jj))
end
```

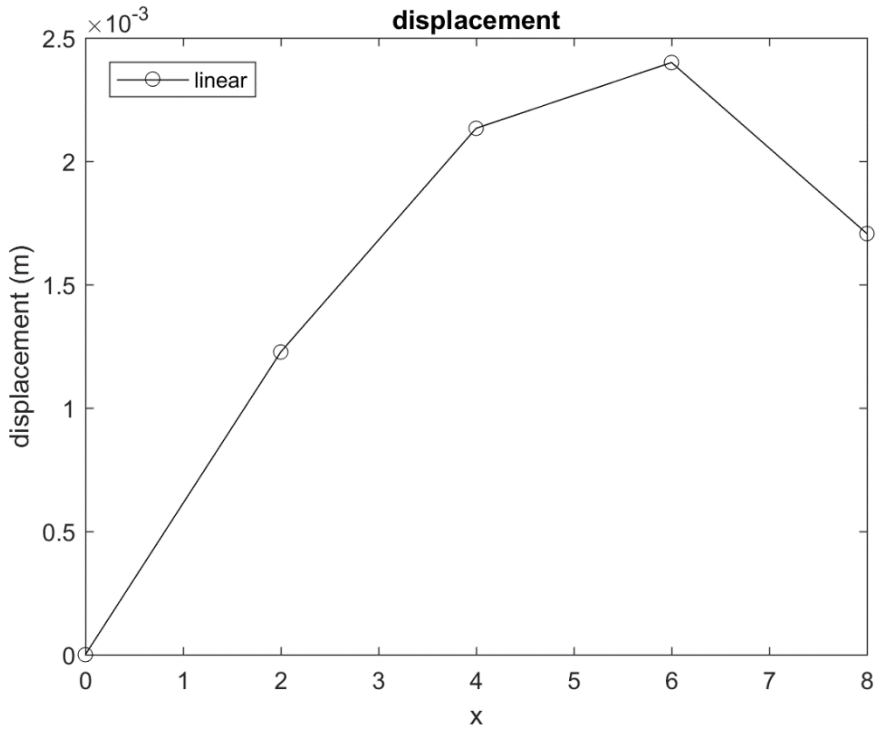
```
% reactions
F=stiffness*displacements;
reactions=F(prescribedDof)-force(prescribedDof);
disp('Reactions')
fprintf('node     reactions\n')
for jj=1:size(prescribedDof,1)
    fprintf('%2.0f:       %10.4e\n', prescribedDof(jj), reactions(jj))
end
disp('Element Forces')
fprintf('element   node_I   node_J      force I       force J\n')
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  k(e)=E(e)*A(e)/L(e) ;
  elementForce=k(e)*[1 -1;-1 1]*displacements(elementDof);
  fprintf('%5d    %5d    %5d     %10.4e
%10.4e\n',e,elementDof(1),elementDof(2),elementForce(1),elementForce(2))
end
```

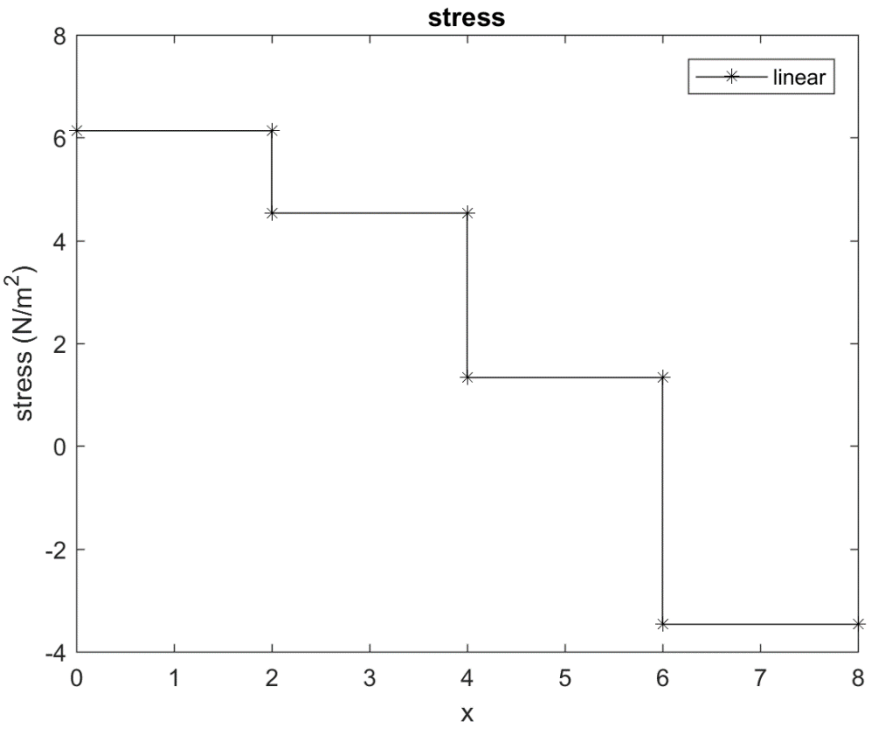
2. (20 %) Consider a bar under compression illustrated below



The material parameters considered are $E = 10^4\ {}^{\mathrm{N}}/_{\mathrm{m}^2}$, $A = 5.0\ \mathrm{m}^2$, $c = 2.0\ {}^{\mathrm{N}}/_{\mathrm{m}^2}$ and $l = 4\ \mathrm{m}$. Model the problem with <u>4 uniformly spacing linear elements</u>. Report nodal displacements and reactions. Plot the distribution of displacements and stresses along the bar. Below are sample outputs and MATLAB plots.

```
Displacements
node      displacements
 1:         0.0000e+00
 2:         1.2267e-03
 3:         2.1333e-03
 4:         2.4000e-03
 5:         1.7067e-03
Reactions
node      reactions
 1:         -3.2000e+01
```



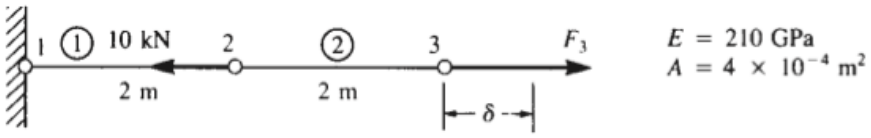



```
% Isoparametric Formulation Implementation
% clear memory
clear all
%%
% E: modulus of elasticity
% A: area of cross section
% L: length of bar
E=1e4; A=5; L=[2 2 2 2];

% numberElements: number of elements
numberElements=4;
% numberNodes: number of nodes
numberNodes=5;
```

```
% generation of coordinates and connectivities
elementNodes=[1 2;2 3;3 4;4 5];
nodeCoordinates=[0 2 4 6 8];
% for structure:
   % displacements: displacement vector
   % force : force vector
   % stiffness: stiffness matrix
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);
force(5)=-32;
%%
% computation of the system stiffness matrix and force vector
syms x
b=2*x;
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  detJacobian=L(e)/2;
  invJacobian=1/detJacobian;
  ngp = 2;
  [w,xi]=gauss1d(ngp);
  for ip=1:ngp;
      [shape, nB]=shapeFunc(2,xi(ip));
      N=vpa(shape);
      x_n=N*nodeCoordinates(elementDof)';
      B=vpa(nB)*invJacobian;
      stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+ B'*B*w(ip)*detJacobian*E*A;
      force(elementDof)=force(elementDof)+...
          subs(b,x_n)*N'*detJacobian*w(ip);
  end
end

% boundary conditions and solution
% prescribed dofs
prescribedDof=[1];
% solution
GDof=numberNodes;
displacements=solution(GDof,prescribedDof,stiffness,force);
% output displacements/reactions
outputDisplacementsReactions(displacements,stiffness, ...
    numberNodes,prescribedDof,force)
%% stress
plot_stress_x=zeros(2*numberElements,1);
plot_stress=zeros(2*numberElements,1);
for e=1:numberElements;
    % elementDof: element degrees of freedom (Dof)
    elementDof=elementNodes(e,:) ;
    detJacobian=L(e)/2;
    invJacobian=1/detJacobian;
    [~, nB]=shapeFunc(2);
    B=vpa(nB)*invJacobian;
    plot_stress(2*e-1: 2*e)=vpa(subs(B,[-1;1])*E*displacements(elementDof));
    plot_stress_x(2*e-1: 2*e)=nodeCoordinates(elementDof);
end
%%
figure(1)
plot(nodeCoordinates,displacements,'ko-')
title('displacement');
xlabel('x');ylabel('displacement (m)');
legend('linear','location','northwest')
figure(2)
plot(plot_stress_x,plot_stress,'k*-')
```

```
title('stress');
xlabel('x');ylabel('stress (N/m^2)');
legend('linear')


%.....................................................................
function [shape, B]=shapeFunc(n,xi)
syms x;
eN=n-1;
coor=-1:2/eN:1;
shape=sym(ones(1,n));
for i=1:n
    for j=1:n
        if (i~=j)
            shape(i)=shape(i)*(x-coor(j))/(coor(i)-coor(j));
        end
    end
end
B=diff(shape,x,1);
if nargin==2
    shape=subs(shape,x,xi);
    B=subs(B,x,xi);
end
end
```

3. (30%) Consider a bar shown below with two linear element discretization. F_3 is the force large enough to induce the prescribed displacement. Prompt the user to enter the maximum allowable member force and report the maximum allowable prescribed displacement.



Below are two sample runs:

Run 1

```
Enter the maximum allowable member force (unit: kN):500
The maximum allowable prescribed displacement is 2.3571e+01
(unit: mm)
```

Run 2

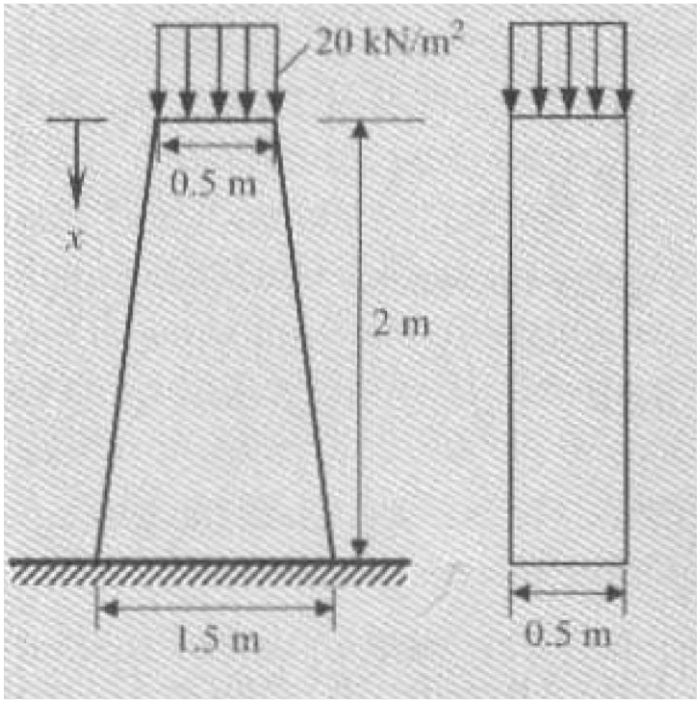
```
Enter the maximum allowable member force (unit: kN):700
The maximum allowable prescribed displacement is 3.3095e+01
(unit: mm)
```

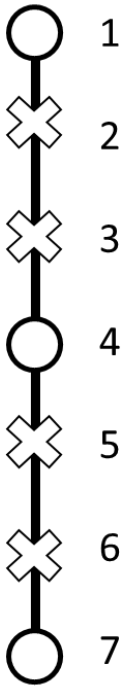
```
% 1D Simple Bar
% clear memory
clear all
```

```
%%
% E: modulus of elasticity
% A: area of cross section
% L: length of bar
E=210; A=400; L=[2000 2000];

% numberElements: number of elements
numberElements=2;
% numberNodes: number of nodes
numberNodes=3;
% generation of coordinates and connectivities
elementNodes=[1 2;2 3];
nodeCoordinates=[0 2 4];
% for structure:
   % displacements: displacement vector
   % force : force vector
   % stiffness: stiffness matrix
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);
force(2)=-10;
force(3)=input('Enter the maximum allowable member force (unit: kN):');
%%
% computation of the system stiffness matrix
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  k(e)=E*A/L(e) ;
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+k(e)*[1 -1;-1 1];
end
% boundary conditions and solution
% prescribed dofs
prescribedDof=[1];
% solution
GDof=numberNodes;
displacements=solution(GDof,prescribedDof,stiffness,force);
% output displacements/reactions
fprintf('The maximum allowable prescribed displacement is %.4e
(unit:mm)\n',displacements(3))
```

4. **(30%)** Consider the concrete pier problem shown below. The load $20\ \frac{\text{kN}}{\text{m}^2}$ represent the weight of bridge and a distribution of the traffic on the bridge. The concrete weighs $24\ \frac{\text{kN}}{\text{m}^3}$ and its modulus is $E = 2 \times 10^7\ \frac{\text{kN}}{\text{m}^2}$. Solve for the axial displacement and stress using 2 uniformly spacing cubic elements with a linearly varying area. Use three Gauss points for the problem. Below are the cubic shape functions in the parametric space $\xi \in [-1 \quad +1]$.

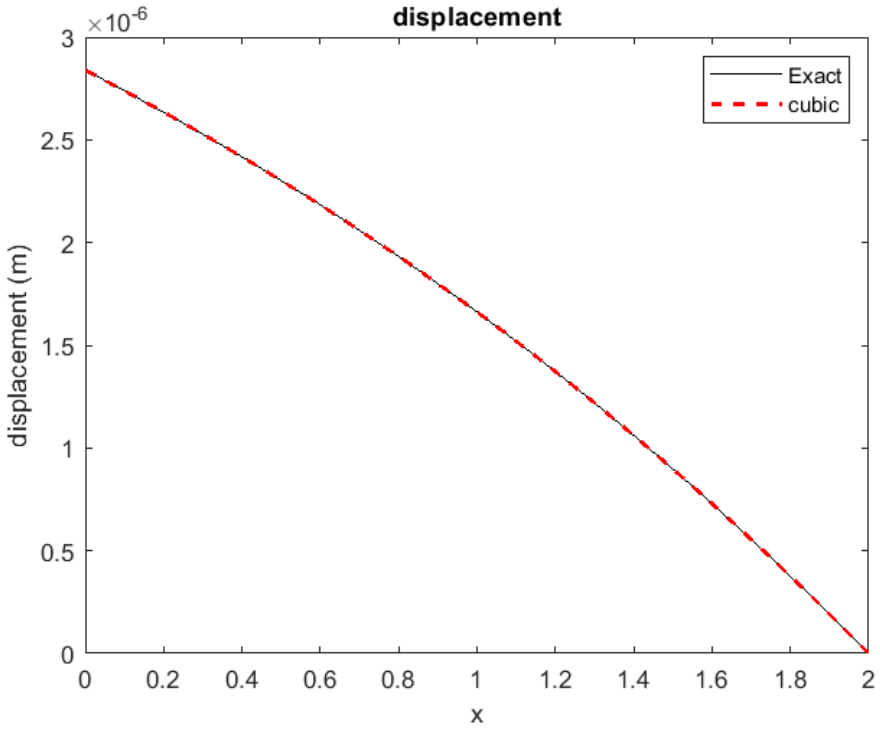
$$\mathbf{N}^{\text{L4}} = \left[-\frac{9}{16}\left(\xi+\frac{1}{3}\right)\left(\xi-\frac{1}{3}\right)(\xi-1) \quad \frac{27}{16}(\xi+1)\left(\xi-\frac{1}{3}\right)(\xi-1) \right.$$
$$\left. -\frac{27}{16}(\xi+1)\left(\xi+\frac{1}{3}\right)(\xi-1) \quad \frac{9}{16}(\xi+1)\left(\xi+\frac{1}{3}\right)\left(\xi-\frac{1}{3}\right)\right]$$

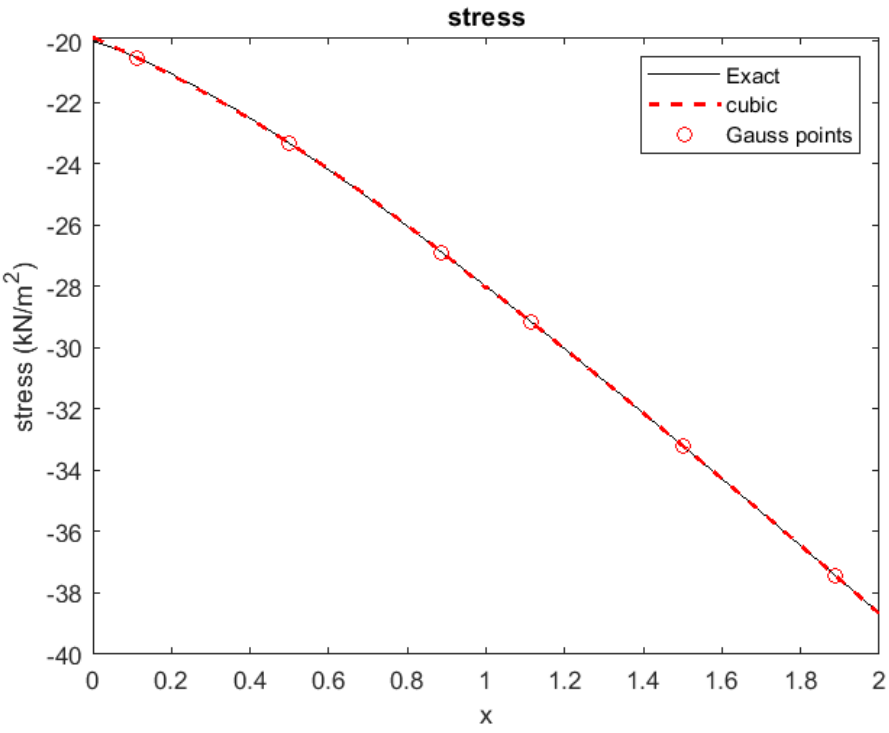




Below are sample outputs and MATLAB plots.

```
Displacements
node      displacements
 1:         2.8394e-06
 2:         2.4910e-06
 3:         2.1017e-06
 4:         1.6622e-06
 5:         1.1672e-06
 6:         6.1377e-07
 7:         0.0000e+00
Reactions
node      reactions
 7:         -2.9000e+01
```






```
% Isoparametric Formulation Implementation
% clear memory
clear all
%%
% E: modulus of elasticity
% A: area of cross section
% L: length of bar
syms x
E=2e7; A=0.25*(1+x); L=[1 1];

% numberElements: number of elements
```

```
numberElements=2;
% numberNodes: number of nodes
numberNodes=7;
% generation of coordinates and connectivities
elementNodes=[1 2 3 4;4 5 6 7];
nodeCoordinates=0:2/6:2;
% for structure:
   % displacements: displacement vector
   % force : force vector
   % stiffness: stiffness matrix
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);
force(1)=20*0.25;
%%
% computation of the system stiffness matrix and force vector
b=24*A;
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  detJacobian=L(e)/2;
  invJacobian=1/detJacobian;
  ngp = 3;
  [w,xi]=gauss1d(ngp);
  for ip=1:ngp;
      [shape, nB]=shapeFunc(4,xi(ip));
      N=vpa(shape);
      x_n=N*nodeCoordinates(elementDof)';
      B=nB*invJacobian;
      stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+ B'*B*w(ip)*detJacobian*E*subs(A,x_n);
      force(elementDof)=force(elementDof)+...
          subs(b,x_n)*N'*detJacobian*w(ip);
  end
end

% boundary conditions and solution
% prescribed dofs
prescribedDof=[7];
% solution
GDof=numberNodes;
displacements=solution(GDof,prescribedDof,stiffness,force);
% output displacements/reactions
outputDisplacementsReactions(displacements,stiffness, ...
    numberNodes,prescribedDof,force)
%% stress
x=linspace(-1,1,100);
plot_stress_x=zeros(1,100*numberElements);
plot_stress=zeros(1,100*numberElements);
for e=1:numberElements;
    % elementDof: element degrees of freedom (Dof)
    elementDof=elementNodes(e,:) ;
    detJacobian=L(e)/2;
    invJacobian=1/detJacobian;
    dx=(nodeCoordinates(elementDof(end))-nodeCoordinates(elementDof(1)))/100;
    [~, nB]=shapeFunc(4);
    B=nB*invJacobian;
    plot_stress(e+100*(e-1): e+100*e)=vpa(subs(B*E*displacements(elementDof),-
1:0.02:1));
    plot_stress_x(e+100*(e-1): e+100*e)=nodeCoordinates(elementDof(1)): dx:
nodeCoordinates(elementDof(end));
end
%% gauss
for e=1:numberElements;
```

```
    % elementDof: element degrees of freedom (Dof)
    elementDof=elementNodes(e,:) ;
    detJacobian=L(e)/2;
    invJacobian=1/detJacobian;
    ngp = 3;
    [w,xi]=gauss1d(ngp);

    [shape, nB]=shapeFunc(4);
    B=nB*invJacobian;
    plot_gauss(3*e-2: 3*e)=vpa(subs(B*E*displacements(elementDof),xi));
    plot_gauss_x(3*e-2:
3*e)=xi*detJacobian+0.5*(nodeCoordinates(elementDof(1))+nodeCoordinates(elementD
of(end)));
end
%% displacements
x=linspace(-1,1,100);
plot_u_x=zeros(1,100*numberElements);
plot_u=zeros(1,100*numberElements);
for e=1:numberElements;
    % elementDof: element degrees of freedom (Dof)
    elementDof=elementNodes(e,:) ;
    detJacobian=L(e)/2;
    invJacobian=1/detJacobian;
    dx=(nodeCoordinates(elementDof(end))-nodeCoordinates(elementDof(1)))/100;
    [shape, ~]=shapeFunc(4);
    N=vpa(shape);
    plot_u(e+100*(e-1): e+100*e)=subs(N*displacements(elementDof),-1:0.02:1);
    plot_u_x(e+100*(e-1): e+100*e)=nodeCoordinates(elementDof(1)): dx:
nodeCoordinates(elementDof(end));
end
%% exact
syms x u(x);
ode=diff(2e7*0.25*(1+x)*diff(u,1),1)+24*0.25*(1+x)==0;
cond1=u(2)==0;
du=diff(u,x);
cond2=E*du(0)==-20;
exact_disp=dsolve(ode,[cond1, cond2]);
ext_stress=E*diff(exact_disp);
%% plot
figure(1)
ext_disp_plot=ezplot(exact_disp,[0 2]);hold on
set(ext_disp_plot,'Color','k')
plot(plot_u_x,plot_u,'r--','linewidth',2);
title('displacement');
xlabel('x');ylabel('displacement (m)');
legend('Exact','cubic')

figure(2)
ext_stress_plot=ezplot(ext_stress,[0 2]);hold on
set(ext_stress_plot,'Color','k')
plot(plot_stress_x,plot_stress,'r--','linewidth',2);hold on
plot(plot_gauss_x,plot_gauss,'ro');
title('stress');
xlabel('x');ylabel('stress (kN/m^2)');
axis([0 2 -40 -20]);
legend('Exact','cubic','Gauss points')


%.....................................................................
function [shape, B]=shapeFunc(n,xi)
syms x;
eN=n-1;
coor=-1:2/eN:1;
```

```
shape=sym(ones(1,n));
for i=1:n
    for j=1:n
        if (i~=j)
            shape(i)=shape(i)*(x-coor(j))/(coor(i)-coor(j));
        end
    end
end
B=diff(shape,x,1);
if nargin==2
    shape=subs(shape,x,xi);
    B=subs(B,x,xi);
end
end
```