9. Linear Algebra Computation
Basic Linear Algebra Subprograms (BLAS)

- Routines that provide standard, low-level, building blocks for performing basic vector and matrix operations.

- Originally developed in 1979.

- **Level-1** BLAS perform scalar, vector and vector-vector operations
- **Level-2** BLAS perform matrix-vector operations
- **Level-3** BLAS perform matrix-matrix operations

- Machine-specific optimized BLAS libraries are available for a variety of computer architectures. (provided by the computer or software vendor, e.g., Intel® Math Kernel Library)

- Because the BLAS are efficient, portable, and widely available, they are commonly used in the development of high quality linear algebra software, e.g., LAPACK (Linear Algebra PACKage).

- For source codes and documents see: [www.netlib.org/blas](http://www.netlib.org/blas)
Example: matrix-matrix multiplication (Level-3 BLAS)  

\[ C \leftarrow \alpha \times \text{op}(A) \times \text{op}(B) + \beta \times C \]

**SUBROUTINE SGEMM**(TRANSA, TRANSB, M, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC)

**S**: single precision  
**GE**: general matrix  
**MM**: matrix-matrix

**TRANSA**: the form of \( \text{op}(A) \) to be used

- 'N' or 'n', \( \text{op}(A) = A \)
- 'T' or 't', \( \text{op}(A) = A^T \)
- 'C' or 'c', \( \text{op}(A) = A^T \) (transpose, when \( A \) is real)  
  \( A^H \) (conjugate transpose, when \( A \) is complex in CGEMM)

**M**: the number of rows of \( \text{op}(A) \) and \( C \)  
**N**: the number of columns of \( \text{op}(B) \) and \( C \)  
**K**: the number of columns of \( \text{op}(A) \) and the number of rows of \( \text{op}(B) \)  
**ALPHA, BETA**: the scalar coefficients \( \alpha \) and \( \beta \)  
**LDA, LDB, LDC**: the first dimension of \( A, B \) and \( C \) as declared in the calling program  
**A**: real array of dimension \((LDA, ka)\), where \( ka \) is \( K \) when \( \text{TRANSA} = 'N' \), and is \( M \) otherwise  
**B**: real array of dimension \((LDB, kb)\), where \( kb \) is \( N \) when \( \text{TRANSB} = 'N' \), and is \( K \) otherwise  
**C**: real array of dimension \((LDC, N)\)
• To use BLAS and LAPACK:

  1. Download the zipped file `blas_lapack_lib.zip` containing the pre-built libraries.

  2. Unzip and put the four files `libblas.dll`, `libblas.lib`, `liblapack.dll` and `liblapack.lib` in the `GNU_emacs_Fortran` folder

• To link the BLAS library in compiling:

  ```
  > gfortran -L. -lblas my_program.f90
  ```
Example: using BLAS \texttt{sgemm} to re-do matrix multiplication exercise

\begin{verbatim}
$ gfortran -L. -lblas example_blas.f90$
\end{verbatim}

```fortran
program example_blas
  implicit none
  integer, parameter :: m=3, k=2, n=5
  real :: a(m,k) = reshape((/1.3,3.6,3.05,2.5,-2.0,-0.03/),(/m,k/))
  real :: b(k,n) = reshape((/2.0,12.4,-0.2,2.7,3.4,-7.1,38.9,1.2,23.9,2.4/),(/k,n/))
  real :: c(m,n) = 0.0, c_blas(m,n) = 0.0, alpha, beta
  integer :: i, j, ii, lda, ldb, ldc
  do i = 1, m
    do j = 1, n
      do ii = 1, k
        c(i,j) = c(i,j) + a(i,ii)*b(ii,j)
      end do
    end do
  end do
  write(*,*) 'Hand crafted:', (c(i,i), i=1,min(m,n))
  alpha = 1.0
  beta = 0.0
  lda = m
  ldb = k
  ldc = m
  call sgemm('N','N',m,n,k,alpha,a,lda,b,ldb,beta,c_blas,ldc)
  write(*,*) 'BLAS sgemm: ', (c_blas(i,i), i=1,min(m,n))
  write(*,*) 'Max error: ', maxval(abs(c-c_blas))
end program
```

\[ a = \begin{bmatrix} 1.3 & 2.5 \\ 3.6 & -2.0 \\ 3.05 & -0.03 \end{bmatrix} \]

\[ b = \begin{bmatrix} 2.0 & -0.2 & 3.4 & 38.9 & 23.9 \\ 12.4 & 2.7 & -7.1 & 1.2 & 2.4 \end{bmatrix} \]
Example: A test driver for using BLAS \texttt{sgemm}; compare with intrinsic function \texttt{matmul}

```
gfortran -L. -lblas t_blas.f90
```

```fortran
program t_blas
implicit none
integer,parameter :: m=2000, k=1500, n=1000
integer :: lda=m, ldb=k, ldc=m
real :: random, t1, t2, t3, alpha, beta
real :: a(m,k), b(k,n), c(m,n), c_intr(m,n)
integer :: i, j, iseed

!-- Assign random values to a, b and c
iseed = 65432
do j = 1, k
   do i = 1, m
      a(i,j) = random(iseed)
   end do
end do
do j = 1, n
   do i = 1, k
      b(i,j) = random(iseed)
   end do
end do
do j = 1, n
   do i = 1, m
      c(i,j) = random(iseed)
   end do
end do
alpha = random(iseed)
beta = random(iseed)
```
Compute $\alpha a + \beta c$ using intrinsic function matmul and BLAS sgemm

```fortran
    call cpu_time(t1)
    !
c_intr = alpha*matmul(a,b) + beta*c
    !
call cpu_time(t2)
    !
call sgemm('N','N',m,n,k,alpha,a,lda,b,ldb,beta,c,ldc)
    !
call cpu_time(t3)
    !
    !-- Find the maximum error of abs(c-c_intr) using intrinsic function maxval
write(*,*) 'Max error: ', maxval(abs(c-c_intr))
write(*,*) 'Elapsed time of matmul: ', t2-t1
write(*,*) 'Elapsed time of sgemm: ', t3-t2
end program
```

```
> gfortran -L. -lblas t_blas.f90 random.f90
> a.exe
> Max error: 1.23977661E-05
> Elapsed time of matmul: 3.5724230
> Elapsed time of sgemm: 25.474962
```
• The pre-built BLAS has *not* been optimized for GNU Fortran.

• But, using the optimized BALS of [Intel math kernel library](https://software.intel.com/en-us/intel-mkl) for [Intel Fortran](https://software.intel.com/en-us/intel-fortran-compilers), the performance of BLAS becomes much better.

Using **GNU gfortran** and pre-built BLAS:

- Max error: 1.23977661E-05
- Elapsed time of `matmul`: 2.6832180
- Elapsed time of `sgemm`: 23.712152

Using **Intel Fortran** and BLAS in Intel math kernel library:

- Max error: 8.8214874E-06
- Elapsed time of `matmul`: 1.007847
- Elapsed time of `sgemm`: 0.3469471
Linear Algebra PACKage (LAPACK)

- Solve systems of linear equations, linear least-squares problems, eigenvalue problems, and singular value problems.

- LAPACK can also handle many associated computations, such as matrix factorizations or estimating condition numbers.

- Written in Fortran and call level-3 BLAS as much as possible for the computation.

- Originally written in FORTRAN 77, but moved to Fortran 90 in version 3.2 (2008); The current version is 3.6.1 (06/2016).

- LAPACK can be seen as the successor to the linear equations and linear least-squares routines of LINPACK and the eigenvalue routines of EISPACK.

LINPACK is still being used as the benchmark for measuring the performance of top 500 supercomputers. (www.top500.org/project/linpack/)
The original goal of the LAPACK project was to make the widely used EISPACK and LINPACK libraries run efficiently on shared-memory vector and parallel processors.

For source codes and documents see:  www.netlib.org/lapack

LAPACK contains three levels of routines

**driver routines**: for solving standard types of problems, typically calls a sequence of computational routines.

**computational routines**: to perform a distinct computational task.

**auxiliary routines**: to perform a certain subtask or common low-level computation.

Both real and complex matrices are provided for in LAPACK.

Dense and band matrices are provided for in LAPACK, but not general sparse matrices.
**Example:** solve a linear system with a general non-symmetric matrix

\[ [A] \{x\} = \{B\} \]

```plaintext
subroutine SGESV(N,NRHS,A,LDA,IPIV,B,LDB,INFO)
```

**S:** single precision  
**GE:** general non-symmetric matrix  
**SV:** solve linear system

---

**[in] N**  
The number of linear equations, i.e., the order of the matrix \( A \).

**[in] NRHS**  
The number of right hand sides, i.e., the number of columns of the matrix \( B \).

**[in,out] A(LDA,N)**  
On entry, the \( N \times N \) coefficient matrix \( A \).  
On exit, the factors \( L \) and \( U \) from the factorization \( A = P*L*U \).

**[in] LDA**  
The leading dimension of the array \( A \).

**[out] IPIV(N)**  
The pivot indices that define the permutation matrix \( P \).

**[in,out] B(LDB,NRHS)**  
On entry, the right-hand-side matrix \( B \).  
On exit, if \( INFO = 0 \), the \( N \times NRHS \) solution matrix \( X \).

**[in] LDB**  
The leading dimension of the array \( B \).

**[out] INFO**  
\( INFO = 0 \): successful exit.  
\( INFO = -i \): unsuccessful exit, the i-th argument had an illegal value.  
\( INFO = i \): unsuccessful exit, \( U(i,i) \) is exactly zero.
To use BLAS and LAPACK:

1. Download the zipped file `blas_lapack_lib.zip` containing the pre-built libraries.

2. Unzip and put the four files `libblas.dll`, `libblas.lib`, `liblapack.dll` and `liblapack.lib` in the `GNU_emacs_Fortran` folder.

To link the LAPACK library in compiling:

```bash
> gfortran -L. -llapack my_program.f90
```
**Example:** using LBLAS `sgesv` to solve a system of linear equations

\[
\begin{align*}
    x_1 + x_2 + x_3 &= 1. & x_1 &= 1. \\
    2x_1 + x_2 + x_3 &= 2. & x_2 &= 3. \\
    x_1 + 3x_2 + 2x_3 &= 4. & x_3 &= -3.
\end{align*}
\]

\[
\begin{align*}
    x_1 + x_2 + x_3 &= 0. & x_1 &= -1. \\
    2x_1 + x_2 + x_3 &= -1. & x_2 &= 0. \\
    x_1 + 3x_2 + 2x_3 &= 1. & x_3 &= 1.
\end{align*}
\]

> `gfortran -L. -llapack example_lapack.f90`
**Exercise:** Test driver for the subroutine of solution of system of linear equations

- A genetic method to test a solver for a linear system \([A]x\{x\} = \{B\}\) is to invent a coefficient matrix \([A]\) and an “known” unknown vector \(\{x\}\).

  The right-hand side of the equation \(\{B\}\) can then be calculated by multiplying the two matrices \(\{B\} = [A]x\{x\}\).

  The vector \(\{x\}\), which has been generated, is the known exact solution of the system of equations, and is denoted by \(\{x\_exact\}\).

- Given \([A]\) and \(\{B\}\), the subroutine of solution of system of linear equations (such as SGESV in LAPACK) is then called to solve the system of equations, and the result is denoted as \(\{x\_solve\}\).

- Since we know the exact solution \(\{x\_exact\}\), the numerical error when solving the equations, which is defined as the maximum of the absolute value of \((x\_exact_j - x\_solve_j)\), can be computed.

  If the system of equation has been solved correctly, the maximum error should be within the a limit determined by the precision of the computer.
Procedure:

- Generate matrix \([A]\)
- Generate vector \(\{x_{exact}\}\)
- Compute vector \(\{B\} = [A] \times \{x_{exact}\}\)
- Solve \([A] \times \{x\} = \{B\}\) for \(\{x\}\)
- Find the maximum of \(\|\{x\} - \{x_{exact}\}\|\)
Write a test driver for SGESV in LAPACK. The program should contain the following components:

1. Input the dimension of the linear system, \texttt{ndim}.

2. Generate the elements of the matrix \([A]\) and the vector \{\texttt{x\_exact}\} using the random number generation function/subroutine. The values of the elements of \([A]\) and \{\texttt{x}\} will be between 1 and -1.

3. Call the matrix-vector multiplication routine \texttt{SGEMV} in \texttt{BLAS} to compute
   \(\{B\} = [A]\times\{\texttt{x\_exact}\}\). The values of the elements of \([B]\) will also have the magnitude of order one.

4. Call the subroutine \texttt{SGESV} in \texttt{LAPACK} to solve the system of equations: \([A]\times\{\texttt{x\_solve}\} = \{B\}\) for the solution \{\texttt{x\_solve}\}.

5. Find the maximum error of the solved unknowns \{\texttt{x\_solve}\} given the known solution \{\texttt{x\_exact}\}, and output the result.

6. You need to link both libraries of Blas and Lapack in compiling the program:
   
   > gfortran -L. -lblas -llapack my_program.f90