CEE 618 Scientific Parallel Computing (Lecture 3)

Linear Algebra Basics using LAPACK

Albert S. Kim

Department of Civil and Environmental Engineering
University of Hawai'i at Manoa
2540 Dole Street, Holmes 383, Honolulu, Hawaii 96822

Table of Contents

Outline

Convection-Diffusion-Reaction Equation

- General form

$$\frac{\partial C}{\partial t} = \nabla \cdot (D \nabla C) - \nabla \cdot (\boldsymbol{v} C) - kC \quad (1)$$

Convection-Diffusion-Reaction Equation

- General form

$$\frac{\partial C}{\partial t} = \nabla \cdot (D \nabla C) - \nabla \cdot (\boldsymbol{v} C) - kC \tag{1}$$

- In a steady state without convection and reaction

$$0 = \nabla \cdot (D \nabla C) \tag{2}$$

Convection-Diffusion-Reaction Equation

- General form

$$\frac{\partial C}{\partial t} = \nabla \cdot (D \nabla C) - \nabla \cdot (\boldsymbol{v} C) - kC \quad (1)$$

- In a steady state without convection and reaction

$$0 = \nabla \cdot (D \nabla C) \quad (2)$$

- In 2D with a **constant** diffusion coefficient

$$0 = \frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} \quad (3)$$

Laplacian eq.
$\nabla^2 C = 0$

Convection-Diffusion-Reaction Equation

- General form

$$\frac{\partial C}{\partial t} = \nabla \cdot (D \nabla C) - \nabla \cdot (\boldsymbol{v} C) - kC \tag{1}$$

- In a steady state without convection and reaction

$$0 = \nabla \cdot (D \nabla C) \tag{2}$$

- In 2D with a **constant** diffusion coefficient

$$0 = \frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} \tag{3}$$

- Mathematically identical to heat diffusion ($C \to T$)

$$0 = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \tag{4}$$

- Examples? Let's watach some videos in http://albertsk.org/videos/physical/

Example problem

Solve the following equation using the method of separation of variables:

$$\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} = 0 \tag{5}$$

- Boundary conditions $(0 < x, y < L)$
  1. $C\left(x = 0, y\right) = 0$
  2. $C\left(x, y = 0\right) = 0$
  3. $C\left(x, y = L\right) = 0$
  4. $C\left(x = L, y\right) = 10 \sin\left(\frac{\pi y}{L}\right)$

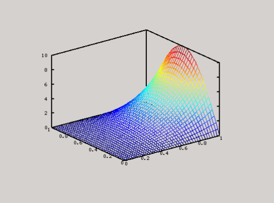


Figure: How does your solution look like?

Solution

1. By the method of the separation of variables

$$C(x, y) = X(x)\, Y(y) \tag{6}$$

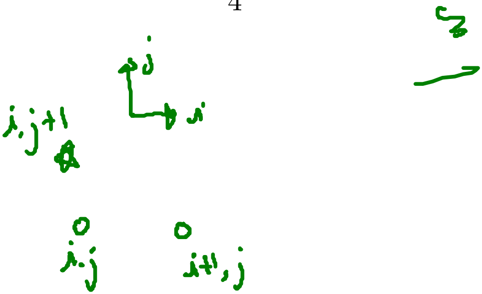
$$C = \frac{10}{\sinh \pi} \sinh \frac{\pi x}{L} \sin \frac{\pi y}{L} \tag{7}$$

Prove.

Solution

2. By MS Excel: *I am nothing but an average of my neighbors.*

$$C_{ij} = \frac{C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1}}{4} \tag{8}$$



Outline

Example

Tony is two years odlder than Sam and the sum of their current ages is twenty. How old are Tony and Sam? Use a two by two matrix to solve this problem.

Cramer's rule

$$T - S = 2 \quad (9)$$

$$T + S = 20 \quad (10)$$

$2T + 0\cdot S = 22$

$T = 11, \; S = 9$

$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} T \\ S \end{bmatrix} = \begin{bmatrix} 2 \\ 20 \end{bmatrix}$

$\text{Det} = \begin{vmatrix} 1 & -1 \\ 1 & 1 \end{vmatrix} = 1 - (-1) = 2$

$T = \dfrac{\begin{vmatrix} 2 & -1 \\ 20 & 1 \end{vmatrix}}{\text{Det}} = \dfrac{2-(-20)}{2} = \dfrac{22}{2} = 11$

$S = \dfrac{\begin{vmatrix} 1 & 2 \\ 1 & 20 \end{vmatrix}}{\text{Det}} = \dfrac{20-2}{2} = 9$

A Linear System

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{11}$$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}_{n\times n}, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n\times 1} \text{ and } \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}_{n\times 1}$$

where $\mathbf{A}$ is a $n \times n$ **square matrix**, and $\mathbf{b}$ is a $n \times 1$ **column vector**, of which all elements are known.

Then, how can we calculate $\mathbf{x}$?

LU decomposition

The square matrix $\mathbf{A}$ can be decomposed into

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \tag{12}$$

where $\mathbf{L}$ and $\mathbf{U}$ are **lower** and **upper** triangular matrixes, respectively, and calculated as

$$L = \begin{pmatrix} \alpha_{11} & 0 & \cdots & 0 \\ \alpha_{21} & \alpha_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{pmatrix}, U = \begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1n} \\ 0 & \beta_{22} & \cdots & \beta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{nn} \end{pmatrix}$$

Then,

$$\mathbf{A} \cdot \mathbf{x} = (\mathbf{L} \cdot \mathbf{U}) \cdot \mathbf{x} = \mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b} \tag{13}$$

Let's set $\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$, then

$$\mathbf{L} \mathbf{y} = \mathbf{b} \tag{14}$$

Forward substitution with known $\mathbf{L}$ and $\mathbf{b}$ to solve for $\mathbf{y}$

$\alpha_{11} y_1 + 0 \cdots = b_1$

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$$

$$\begin{pmatrix} \alpha_{11} & 0 & \cdots & 0 \\ \alpha_{21} & \alpha_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Then,

$$y_1 =$$

Forward substitution with known $\mathbf{L}$ and $\mathbf{b}$ to solve for $\mathbf{y}$

$y = U \cdot x$

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$$

$$\begin{pmatrix} \alpha_{11} & 0 & \cdots & 0 \\ \alpha_{21} & \alpha_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{pmatrix} \cdot \begin{pmatrix} \mathrm{y}_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \mathrm{b}_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Then,

$$y_1 = \frac{b_1}{\alpha_{11}}, \quad y_2 = \frac{b_2 - \alpha_{11} y_1}{\alpha_{22}}, \quad \dots \tag{15}$$

Using back substitution,

$$y_i = \frac{1}{\alpha_{ii}} \left[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j \right] \tag{16}$$

where $i = 2, 3, ..., n$.

Backward substitution with $\mathbf{U}$ and $\mathbf{y}$ to solve for $\mathbf{x}$

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y} \tag{17}$$

$$\begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1n} \\ 0 & \ddots & \cdots & \vdots \\ \vdots & \vdots & \beta_{n-1,n-1} & \beta_{n-1,n} \\ 0 & \cdots & 0 & \beta_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \\ \mathrm{x_n} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{n-1} \\ \mathrm{y_n} \end{pmatrix}$$

Then,

$$x_n =$$

Backward substitution with $\mathbf{U}$ and $\mathbf{y}$ to solve for $\mathbf{x}$

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y} \tag{17}$$

$$\begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1n} \\ 0 & \ddots & \cdots & \vdots \\ \vdots & \vdots & \beta_{n-1,n-1} & \beta_{n-1,n} \\ 0 & \cdots & 0 & \beta_{\mathrm{nn}} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \\ \mathrm{x_n} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{n-1} \\ \mathrm{y_n} \end{pmatrix}$$

Then,

$$x_n = \frac{y_n}{\beta_{nn}}, \quad x_{n-1} = \frac{y_{n-1} - \beta_{n-1,n} x_n}{\beta_{n-1,n-1}}, \quad \ldots \tag{18}$$

Using back substitution,

$$x_i = \frac{1}{\beta_{ii}} \left[y_i - \sum_{j=i+1}^{n} \beta_{ij} x_j \right] \tag{19}$$

where $i = n-1, n-2, ..., 1$.

Combined matrix of α's and β's with less memory

Using $\alpha_{ii} = 1$ where $i = 1, 2, ..., n$

$$\mathbf{L} \oplus \mathbf{U} \to \mathbf{C} = \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \cdots & \beta_{1,n-1} & \beta_{1n} \\ \alpha_{21} & \beta_{22} & \beta_{23} & \cdots & \beta_{2,n-1} & \beta_{2n} \\ \alpha_{31} & \alpha_{32} & \beta_{33} & \cdots & \beta_{3,n-1} & \beta_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{n-1,1} & \alpha_{n-1,2} & \alpha_{n-1,3} & \cdots & \beta_{n-1,n-1} & \beta_{n-1,n} \\ \alpha_{n1} & \alpha_{n2} & \alpha_{n3} & \cdots & \alpha_{n,n-1} & \beta_{nn} \end{pmatrix}$$

Example:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{20}$$

$$A = \begin{pmatrix} 1 & 3 & 1 \\ 1 & 1 & 2 \\ 2 & 3 & 4 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} =? \tag{21}$$

$$C = \begin{pmatrix} 2.000000 & 3.000000 & 4.000000 \\ 0.500000 & 1.500000 & -1.000000 \\ 0.500000 & -0.333333 & -0.333333 \end{pmatrix}, x = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} \tag{22}$$

However, C does not directrly represent L and U of matrix A because pivoting exchanges row index during the LU decomposition.

Makefile

1. Use files in "/opt/cee618s13/class03/" to solve this problem using **ludcmp** and **lubksb** subroutines from NRF77[1]
2. Use LAPACK routines[2] of **DGETRF** and **DGETRS**.
3. Check how to link LAPACK in Makefile.

[1] Section 2.3 of "Numerical Recipes in FORTRAN 77", available at http://www.nrbook.com/a/bookfpdf.php

[2] LAPACK user's guide at http://www.netlib.org/lapack/lug/index.html

Using subroutines in NRF: ludcmp & lubksb

```
program LU
implicit none
integer :: i,j, indx(3)
real    :: a(3,3)=(/1.,1.,2.,3.,1.,3.,1.,2.,4./)
real    :: d,b(3)=(/1.,0.,0./)

open(11, file='lu.dat')
!  Display the given matrix, A and b
write(11,*)
do i=1,3
   write(11,"(4(2x,F12.6))") (a(i,j),j=1,3), b(i)
enddo
!  Decomposition of the given matrix A
call ludcmp(a,3,3,indx,d)
!  Display the decomposed matrix, A and b
write(11,*)
do i=1,3
   write(11,"(4(2x,F12.6))") (a(i,j),j=1,3)
enddo
!  Solving for x with the decomposed matrix using backsubsitution
call lubksb(a,3,3,indx,b)
!  Display the decomposed matrix, A and the solution x
write(11,*)
do i=1,3
   write(11,"(4(2x,F12.6))") (a(i,j),j=1,3), b(i)
enddo
stop
end
```

unit

to write a blank line to '11'

$$A = \begin{pmatrix} 1 & 3 & 1 \\ 1 & 1 & 2 \\ 2 & 3 & 4 \end{pmatrix}$$

$a_{11}\ a_{12}\ a_{13}\ b_1$

$a_{11}\ a_{12}\ a_{13}$
$a_{21}\ a_{22}\ a_{23}$
$a_{31}\ a_{32}\ a_{33}$

$\Rightarrow a_{11}\ a_{21}\ a_{31}\ a_{12}\ a_{22}$

./codes/LU/LU3.f90

Results using ludcmp & lubksb

$$\begin{pmatrix} 1 & 3 & 1 \\ 1 & 1 & 2 \\ 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{23}$$

```
   1.000000     3.000000     1.000000     1.000000
   1.000000     1.000000     2.000000     0.000000
   2.000000     3.000000     4.000000     0.000000

   2.000000     3.000000     4.000000
   0.500000     1.500000    -1.000000
   0.500000    -0.333333    -0.333333

   2.000000     3.000000     4.000000     2.000000
   0.500000     1.500000    -1.000000     0.000000
   0.500000    -0.333333    -0.333333    -1.000000
```

./codes/LU/lu.dat

```
      SUBROUTINE ludcmp(a,n,np,indx,d)
      INTEGER n,np,indx(n),NMAX
      REAL d,a(np,np),TINY
      PARAMETER (NMAX=500,TINY=1.0e-20)
      INTEGER i,imax,j,k
      REAL aamax,dum,sum,vv(NMAX)
      d=1.
      do 12 i=1,n
        aamax=0.
        do 11 j=1,n
          if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
11      continue
        if (aamax.eq.0.) pause 'singular matrix in ludcmp'
        vv(i)=1./aamax
12    continue
      do 19 j=1,n
        do 14 i=1,j-1
          sum=a(i,j)
          do 13 k=1,i-1
            sum=sum-a(i,k)*a(k,j)
13        continue
          a(i,j)=sum
14      continue
```

```
      SUBROUTINE lubksb(a,n,np,indx,b)
      INTEGER n,np,indx(n)
      REAL a(np,np),b(n)
      INTEGER i,ii,j,ll
      REAL sum
      ii=0
      do 12 i=1,n
        ll=indx(i)
        sum=b(ll)
        b(ll)=b(i)
        if (ii.ne.0)then
          do 11 j=ii,i-1
            sum=sum-a(i,j)*b(j)
11        continue
        else if (sum.ne.0.) then
          ii=i
        endif
        b(i)=sum
12    continue
      do 14 i=n,1,-1
        sum=b(i)
        do 13 j=i+1,n
          sum=sum-a(i,j)*b(j)
```

Using subroutines in LAPACK: dgetrf & dgetrs

```
program LUlapack
implicit none
integer              :: i,j, ipiv(3), info
double precision     :: a(3,3)=(/1.,1.,2.,3.,1.,3.,1.,2.,4./)
double precision     :: b(3)=(/1.,0.,0./)

open(11,file='lulapack.dat')
!                 Display the given matrix, A and b
write(11,*)
do i=1,3
   write(11,"(4(2x,F12.6)) ") (a(i,j),j=1,3), b(i)
end do
!                 Decomposition of the given matrix A
call dgetrf(3,3,a,3,ipiv,info)
!                 Display the decomposed matrix, A and b
write(11,*)
do i=1,3
   write(11,"(4(2x,F12.6)) ") (a(i,j),j=1,3), b(i)
end do
!                 Solving for x with the decomposed matrix using backsubsitution
call dgetrs('N',3,1,a,3,ipiv,b,3,info)
!                 Display the decomposed matrix, A and the solution x
write(11,*)
do i=1,3
   write(11,"(4(2x,F12.6)) ") (a(i,j),j=1,3), b(i)
end do
stop
end
```

./codes/LUlapack/LU3dlapack.f90

Archives

- **Linear Equations** at `http://www.netlib.org/lapack/lug/node38.html`
- **Individual** at `http://www.netlib.org/lapack/individualroutines.html`
- **Single, REAL** at `http://www.netlib.org/lapack/single/`
- **Double, REAL** at `http://www.netlib.org/lapack/double/`
- **dgetrf** at `http://www.netlib.org/lapack/double/dgetrf.f`
- **dgetrs** at `http://www.netlib.org/lapack/double/dgetrs.f`
- **dgetri** at `http://www.netlib.org/lapack/double/dgetri.f`

Specifically

- call dgetrf (3 , 3 , a , 3 , ipiv , info)

- call dgetrs(’N’ , 3 , 1 , a , 3 , ipiv , b , 3 , info)

Outline

Eigen Value & Eigen Vector

Example: Rotate to principal axes the quadratic surface

$$x^2 + 6xy - 2y^2 - 2yz + z^2 = 24 \tag{24}$$

In matrix form this equation is

$$\begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} 1 & 3 & 0 \\ 3 & -2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 24 \tag{25}$$

or

$$X^T M X = 24 \tag{26}$$

The characteristic equation of this matrix is

$$\begin{vmatrix} 1-\mu & 3 & 0 \\ 3 & -2-\mu & -1 \\ 0 & -1 & 1-\mu \end{vmatrix} = -\mu^3 + 13\mu - 12 = -(\mu - 1)(\mu + 4)(\mu - 3) = 0 \tag{27}$$

The characteristic values are $\underline{\mu = 1, \ -4, \ 3}$.

From

$$\begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} 1 & 3 & 0 \\ 3 & -2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 24 \tag{28}$$

relative to the principal axes (x', y', z'), the quadratic sufrace equation becomes

$$\begin{pmatrix} x' & y' & z' \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = 24 \tag{29}$$

or

$$1 \cdot {x'}^2 + (-4) \cdot {y'}^2 + 3 \cdot {z'}^2 = 24 \tag{30}$$

or

$${X'}^T M' X' = 24 \tag{31}$$

where

$$M' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 3 \end{pmatrix} \tag{32}$$

Eigen vectors are

$$\left(\frac{1}{\sqrt{10}}, \frac{0}{\sqrt{10}}, \frac{3}{\sqrt{10}}\right) \quad \text{for} \quad \mu = 1 \tag{33}$$

$$\left(\frac{-3}{\sqrt{35}}, \frac{5}{\sqrt{35}}, \frac{1}{\sqrt{35}}\right) \quad \text{for} \quad \mu = -4 \tag{34}$$

$$\left(\frac{-3}{\sqrt{14}}, \frac{-2}{\sqrt{14}}, \frac{1}{\sqrt{14}}\right) \quad \text{for} \quad \mu = 3 \tag{35}$$

$$\begin{pmatrix} \frac{1}{\sqrt{10}} & \frac{-3}{\sqrt{35}} & \frac{-3}{\sqrt{14}} \\ 0 & \frac{5}{\sqrt{35}} & \frac{-2}{\sqrt{14}} \\ \frac{3}{\sqrt{10}} & \frac{1}{\sqrt{35}} & \frac{1}{\sqrt{14}} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \tag{36}$$

or

$$C \cdot X = X'$$

$$X^T \cdot C^T = X'^T$$

In other words,

$$\begin{pmatrix} \frac{1}{\sqrt{10}} & 0 & \frac{3}{\sqrt{10}} \\ \frac{-3}{\sqrt{35}} & \frac{5}{\sqrt{35}} & \frac{1}{\sqrt{35}} \\ \frac{-3}{\sqrt{14}} & \frac{-2}{\sqrt{14}} & \frac{1}{\sqrt{14}} \end{pmatrix} \begin{pmatrix} 1 & 3 & 0 \\ 3 & -2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{10}} & \frac{-3}{\sqrt{35}} & \frac{-3}{\sqrt{14}} \\ 0 & \frac{5}{\sqrt{35}} & \frac{-2}{\sqrt{14}} \\ \frac{3}{\sqrt{10}} & \frac{1}{\sqrt{35}} & \frac{1}{\sqrt{14}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 3 \end{pmatrix} \tag{37}$$

- In the eigen vector matrix, the columns can be exchanged and **signs** can be reverted. It is a matter of using right-handed or left-haded coordinates.
- Using transformed coordinates makes the problem mathematically so convenient.
- In quantum mechanics, eigen values are energy and eigen vectors are quantum states.

```fortran
PROGRAM EIGENVV
 IMPLICIT NONE
INTEGER             :: I, INFO, J,  N,   LWORK
DOUBLE PRECISION :: DUMMY(1,1)
DOUBLE PRECISION, ALLOCATABLE, DIMENSION (:,:) :: A, B , VR
DOUBLE PRECISION, ALLOCATABLE, DIMENSION (:)   :: ALPHAR, ALPHAI, BETA, WORK
   open (11,file='mat.in',status='old')
   read (11,*) N
   LWORK = 8*N
   allocate(A(N,N),B(N,N),ALPHAR(N),ALPHAI(N),BETA(N),VR(N,N),WORK(LWORK))
   B = 0.0; do i = 1, N; B(i,i) = 1.0; end do
   READ (11,*) ((A(I,J),J=1,N),I=1,N)
   CALL DGGEV('N','V',N,A,N,B,N, ALPHAR,ALPHAI,BETA,DUMMY,1,VR,N,WORK,LWORK,INFO)
   write(*,*) 'Eigen values are (diagonal) :'
   write(*,"(3(2X,F12.8))") ((A(i,j),J=1,N),I=1,N)
   write(*,*)
   call eigvec_norm (N,VR)
   write(*,*) 'Eigen vectors are :'
   write(*,"(3(2X,F12.8))") ((VR(i,j),J=1,N),I=1,N)
   write(*,*)
   deallocate(A,B,ALPHAR,ALPHAI,BETA,VR,WORK)

contains

    subroutine eigvec_norm (N,VR)
    DOUBLE PRECISION :: VR(N,N)
    integer          :: N, i, j
    double precision :: norm
    do i = 1, 3
       norm = DOT_PRODUCT (VR(:,i)   , VR(:,i)   )
       VR(:,i) = VR(:,i) / sqrt(norm)
    enddo
    end subroutine eigvec_norm
end program
```

(:,:,:) → for 3D

./codes/eigen/eigvv.f90

Makefile

```
srcroot=eigvv
srcfile=$(srcroot).f90
exefile=$(srcroot).x

all:
	ifort $(srcfile) -o $(exefile) -llapack

run:
	./$(exefile)


edit:
	vim $(srcfile)

clean:
	rm -f *.x *.o
```

./codes/eigen/Makefile

Output

```
./eigvv.x
 Eigen values are (diagonal) :
   -4.00000000    -0.00000000     0.00000000
    0.00000000     3.00000000     0.00000000
    0.00000000     0.00000000     1.00000000

 Eigen vectors are :
   -0.50709255    -0.80178373     0.31622777
    0.84515425    -0.53452248    -0.00000000
    0.16903085     0.26726124     0.94868330
```

./codes/eigen/output.dat

Outline

sample0.pbs & sample1.pbs

```
#PBS -S /bin/bash
#PBS -V
uname -n
echo $PBS_O_JOBID
```

./codes/PBS/sample0.pbs

```
#!/bin/bash
#PBS -l walltime=12:00:00
#PBS -N MyJob
#PBS -V
uname -n
echo $PBS_O_JOBID
cd $PBS_O_WORKDIR
pwd
```

./codes/PBS/sample1.pbs

sample2.pbs

```
#!/bin/bash
#PBS -l host=fractal
#PBS -l walltime=12:00:00
#PBS -l select=1:mpiprocs=4:ncpus=4
#PBS -N Sample
#PBS -V
#PBS -j oe
cd $PBS_O_WORKDIR
### put your specific job here after 'time' command ###
time ls -laF
#####################################################
qstat -f $PBS_JOBID
```

./codes/PBS/sample2.pbs

Commands

1. $␣qsub␣<␣sample0.pbs
2. $␣qstat

- The first comman is to submit a job described in "sample0.pbs" to a queueing system, i.e. "torque".
- The second comman is to monitor a status of the job, of which job number was assigned automatically by the first command.
- Observe the directory since each command of "qsub" will generate two files with the job number.
- Look at contents of newly generated files.