

# ESE5023 Assignment06

12132191 陈鹏翰

## 1. Matrix multiplication

**1.1 [5 points]** Write a program `Main.f90` to read `fortran_demo1/M.dat` as the matrix `M`, and `fortran_demo1/N.dat` as the matrix `N`.

`Main.f90`

```
program MainRead

implicit none

integer                                :: u1, u2, mc, mr, nc, nr, i, j
real(8), dimension(:, :), allocatable :: M, N

u1=50
u2=51
mc=3
mr=4
nc=4
nr=3

open(unit=u1, file='M.dat', status='old')
open(unit=u2, file='N.dat', status='old')

allocate(M(mr, mc))
allocate(N(nr, nc))

do i=1, mr
    read(u1, *) M(i, :)
enddo

do i=1, nr
    read(u2, *) N(i, :)
enddo

do i=1, mr
    write(*, *) "Line ", i, ":", M(i, :)
enddo

do i=1, nr
    write(*, *) "Line ", i, ":", N(i, :)
enddo

deallocate(M)
deallocate(N)

End Program MainRead
```

output

```
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran Main.f90 -o all.x
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ ./all.x
Line   1 :   19.480000000000000    15.789999999999999    19.280000000000001
Line   2 :   19.280000000000001    12.920000000000000    15.859999999999999
Line   3 :   15.859999999999999    11.289999999999999    14.039999999999999
Line   4 :   11.930000000000000    18.600000000000001    18.230000000000000
Line   1 :   7.7199999999999998    4.110000000000003    1.439999999999999    4.799999999999998
Line   2 :   5.5499999999999998    4.799999999999998    4.040000000000000    0.589999999999997
Line   3 :   0.589999999999997    8.580000000000001    2.259999999999998    7.719999999999998
```

**1.2 [5 points]** Write a subroutine `Matrix_multip.f90` to do matrix multiplication.

`Matrix_multip.f90`

```
subroutine Matrix_multip(M,N,MN)

implicit none

real(8),dimension(4,3),intent(in)  :: M
real(8),dimension(3,4),intent(in)  :: N
real(8),dimension(4,4),intent(out) :: MN
integer                               :: i,j,k
real(8)                               :: t

do i=1,4
  do j=1,4
    t=0
    do k=1,3
      t=t+M(i,k)*N(k,j)
    enddo
    MN(i,j)=t
  enddo
enddo

end subroutine Matrix_multip
```

**1.3 [5 points]** Call the subroutine `Matrix_multip()` from `Main.f90` to compute  $M*N$ ; write the output to a new file `MN.dat`, values are in formats of `f9.2`.

`Main_.f90`

```
program MainRead

implicit none

integer                               :: u1, u2, mc, mr, nc, nr, i, j
real(8), dimension(:,,:),allocatable :: M, N
real(8), dimension(4,4)               :: MN

u1=50
u2=51
mc=3
mr=4
nc=4
nr=3

open(unit=u1,file='M.dat',status='old')
open(unit=u2,file='N.dat',status='old')

allocate(M(mr,mc))
```

```

allocate(N(nr,nc))

do i=1,mr
  read(u1,*) M(i,:)
enddo

do i=1,nr
  read(u2,*) N(i,:)
enddo

close(u1)
close(u2)

do i=1,mr
  write(*,*) "Line ",i,":",M(i,:)
enddo

do i=1,nr
  write(*,*) "Line ",i,":",N(i,:)
enddo

call Matrix_multip(M,N,MN)

do i=1,4
  write(*,*) "Line ",i,":",MN(i,:)
enddo

open(unit=u1,file='new1.dat',status='replace')
do i=1,4
  write(u1,'(f9.2)') MN(i,:)
enddo

close(u1)

deallocate(M)
deallocate(N)

End Program MainRead

```

## output

```

(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran Main_.f90 Matrix_multip.f90 -o a13.x
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ ./a13.x
Line      1 :  19.480000000000000      15.789999999999999      19.280000000000001
Line      2 :  19.280000000000001      12.920000000000000      15.859999999999999
Line      3 :  15.859999999999999      11.289999999999999      14.039999999999999
Line      4 :  11.930000000000000      18.600000000000001      18.230000000000000
Line      1 :  7.7199999999999998      4.1100000000000003      1.4399999999999999      4.7999999999999998
Line      2 :  5.5499999999999998      4.7999999999999998      4.0400000000000000      0.5899999999999997
Line      3 :  0.5899999999999997      8.5800000000000001      2.2599999999999998      7.7199999999999998
Line      1 :  249.395300000000002      321.277199999999999      135.415599999999998      251.661700000000000
Line      2 :  229.904999999999997      277.335600000000000      115.803600000000000      222.605999999999999
Line      3 :  193.382299999999999      239.839800000000000      100.180399999999999      191.177899999999999
Line      4 :  206.085299999999999      294.725699999999996      133.523000000000000      208.973600000000000
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ vi new1.dat

```

```
249.40
321.28
135.42
251.66
229.90
277.34
115.80
222.61
193.38
239.84
100.18
191.18
206.09
294.73
133.52
208.97
```

## 2. Calculate the Solar Elevation Angle

**2.1 [5 points]** Write a module `Declination_angle` that calculates the *declination angle* on a given date.

`Declination_angle.f90`

```
module Declination_angle

implicit none
!here I consider that there are 30 days in each month.

real, parameter      :: pi=3.1415926536

contains

  subroutine cal_angle(m,d,da)

    implicit none

    integer,intent(in)      :: m, d
    real(8),intent(out)     :: da
    integer                 :: doy

    doy=(m-1)*30+d
    da=asin(sin(-23.44/180*pi)*cos(((360/365.24)*
(doy+10)+360/pi*0.0167*sin(360/365.24*(doy-2)))/180*pi))
    da=da/pi*180

  end subroutine cal_angle
end module Declination_angle
```

`test1.f90_ (for example: Dec. 22 )`

```
program TestProgram

use Declination_angle

implicit none

real(8)           :: angle
integer           :: m, d
```

```

m=12
d=22

call cal_angle(m,d,angle)

write(*,*) angle

end program TestProgram

```

## output

```

(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran test1.f90 Declination_angle.f90 -o a21.x
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ ./a21.x
-23.371110349671525

```

**2.2 [10 points]** Write a module `Solar_hour_angle` that calculates the *solar hour angle* in a given location for a given date and time.

`Solar_hour_angle.f90`

```

module SolarAngleHour

implicit none

real, parameter      :: pi=3.1415926536

contains

subroutine cal_sla(lon,m,d,t,sah)

implicit none

integer,intent(in)      :: m, d
real(8),intent(in)      :: lon, t
real(8),intent(out)     :: sah
integer                 :: doy
real(8)                 :: offset, eot, gam
doy=(m-1)*30+d
gam=2*pi/365*(doy-1+(t-12)/24)
eot=229.18*
(0.000075+0.001868*cos(gam)-0.032077*sin(gam)-0.014615*cos(2*gam)-0.040849*sin(2
*gam))
offset=eot+MOD(lon,15.0)
sah=15*(t-12)+offset/60

end subroutine cal_sla

end module SolarAngleHour

```

`test2.f90`

Los Angeles is at longitude 118.24° West; Longitude = -118.24°. It falls in Pacific Standard Time (UTC-8); ΔTZ = -8 We want to find the solar hour angle at 3:30 PM on November 24th.

```

program Test2

use SolarAngleHour

```

```

implicit none

real(8)      :: t,lon,h
integer      :: m,d

t=15.5
lon=-118.24
m=11
d=24

call cal_sla(lon,m,d,t,h)

write(*,*) h

end program Test2

```

## output

```

(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran test2.f90 Solar_hour_angle.f90 -o a22.x
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ ./a22.x
52.513306131446733

```

**2.3 [5 points]** Write a main program ( `Solar_elevation_angle.f90` ) that uses module `Declination_angle` and `Solar_hour_angle` to calculate and print the SEA in a given location for a given date and time.

`Solar_elevation_angle.f90`

```

program SEA

use Declination_angle
use SolarAngleHour

implicit none

real, parameter      :: pii=3.1415926536
real(8)              :: lat,lon,t,sah,da
integer              :: m,d
real(8)              :: aes

lat=32.22
lon=1.0
t=10.0
m=3
d=3

call cal_angle(m,d,da)
call cal_sla(lon,m,d,t,sah)

aes=asin(sin(lat/180*pii)*sin(da/180*pii)+cos(lat/180*pii)*cos(da/180*pii)*cos(sah/180*pii))
aes=aes/pii*180.0

write(*,*) aes

end program SEA

```

## output

```
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran Solar_elevation_angle.f90 Declination_angle.f90 Solar_hour_angle.f90 -o a23.x
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ ./a23.x
41.045703954998608
```

**2.4 [5 points]** Create a library (`libsea.a`) that contains `Declination_angle.o` and `Solar_hour_angle.o`. Compile `Solar_elevation_angle.f90` using `libsolar.a`. Print the SEA for Shenzhen (22.542883N, 114.062996E) at 10:32 (Beijing time; UTC+8) on 2021-12-31.

`SEA_ShenZhen.f90`

```
! Shenzhen
program SEA

use Declination_angle
use SolarAngleHour

implicit none

real, parameter      :: pii=3.1415926536
real(8)               :: lat,lon,t,sah,da
integer               :: m,d
real(8)               :: aes

lat=22.542883
lon=114.062996
t=10.0+32/60
m=12
d=31

call cal_angle(m,d,da)
call cal_sla(lon,m,d,t,sah)

aes=asin(sin(lat/180*pii)*sin(da/180*pii)+cos(lat/180*pii)*cos(da/180*pii)*cos(sah/180*pii))
aes=aes/pii*180.0

write(*,*) aes

end program SEA
```

```
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran -c Declination_angle.f90
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran -c Solar_hour_angle.f90
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ ar rcvf libsea.a Declination_angle.o Solar_hour_angle.o
a - Declination_angle.o
a - Solar_hour_angle.o
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ gfortran SEA_ShenZhen.f90 -o a24.x -L. -lsea
(base) chen-p@dell-PowerEdge-T640:/portal1/dell/chen-p/Assignment06$ ./a24.x
35.790305803209272
```