TIAN Xiaojie (田晓杰)'s TA report for assignment06
SID: 12132572
Github: https://github.com/sweet234/ESE5023_Assignments_12132572
Responsible TA: HUANG Hao
Grade: 40

1. Matrix multiplication

Good. (15/15)

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`.

```
Program Main

Implicit none

integer                    :: u, i, j
real(4), dimension(5,3) :: M
real(4), dimension(3,5) :: N
real(4), dimension(5,5) :: MN

u = 50

open(unit=u, file='M.dat', status='old')
read(u,*) ((M(i,j),j=1,3),i=1,5)
close(u)

print *, 'M matrix'

do i = 1,5
    write(*,*) M(i,:)
enddo

open(unit=u, file='N.dat', status='old')
read(u,*) ((N(i,j),j=1,5),i=1,3)
close(u)
```

```
print *, 'M matrix'

do i = 1,5
    write(*,*) M(i,:)
enddo

open(unit=u, file='N.dat', status='old')
read(u,*) ((N(i,j),j=1,5),i=1,3)
close(u)

print *, 'N matrix'

do i = 1,3
    write(*,*) N(i,:)
enddo

call Matrix_multip(M,N,MN)

write(*,*) 'shape MN:', shape(MN)

open(unit=u, file='MN.dat', status='replace')
do i = 1,5
  write(u,'(f9.2)') MN(i,:)
```

Have you found your MN.dat is shown in one column? To get 5*5 matrix, please use write(u, ‘(5f9.2)’) MN(i,:)

```
do i = 1,3
    write(*,*) N(i,:)
enddo

call Matrix_multip(M,N,MN)

write(*,*) 'shape MN:', shape(MN)

open(unit=u, file='MN.dat', status='replace')
do i = 1,5
  write(u,'(f9.2)') MN(i,:)
enddo

close(u)

End Program Main
```

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

```
subroutine Matrix_multip(a,b,c)

implicit none

real(4), dimension(5,3), intent(in)  :: a
real(4), dimension(3,5), intent(in)  :: b
real(4), dimension(5,5), intent(out) :: c

c = matmul(a, b)

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`.

```
M matrix
  19.4799995       15.7900000       19.2800007
  19.2800007       12.9200001       15.8599997
  15.8599997       11.2900000       14.0400000
  11.9300003       18.6000004       18.2299995
  19.2800007       12.9200001       15.8599997
N matrix
  7.71999979       4.11000013       1.44000006       4.80000019       5.55000019
  5.55000019       4.80000019       4.03999996      0.589999974       8.57999992
 0.589999974       8.57999992       2.25999999       7.71999979       4.11000013
shape MN:              5            5
```

2. Calculate the Solar Elevation Angle

The solar elevation angle (SEA) is the angle between the imaginary horizontal plane on which you are standing and the sun in the sky. SEA is very important in deciding the inclination of solar panels, in both photovoltaics (PV) and thermal. The value of the SEA depends on the location on the Earth and the local date and time.

Please read this Solar Elevation Angle – Calculating Altitude of Sun and links therein for how to calculate SEA.

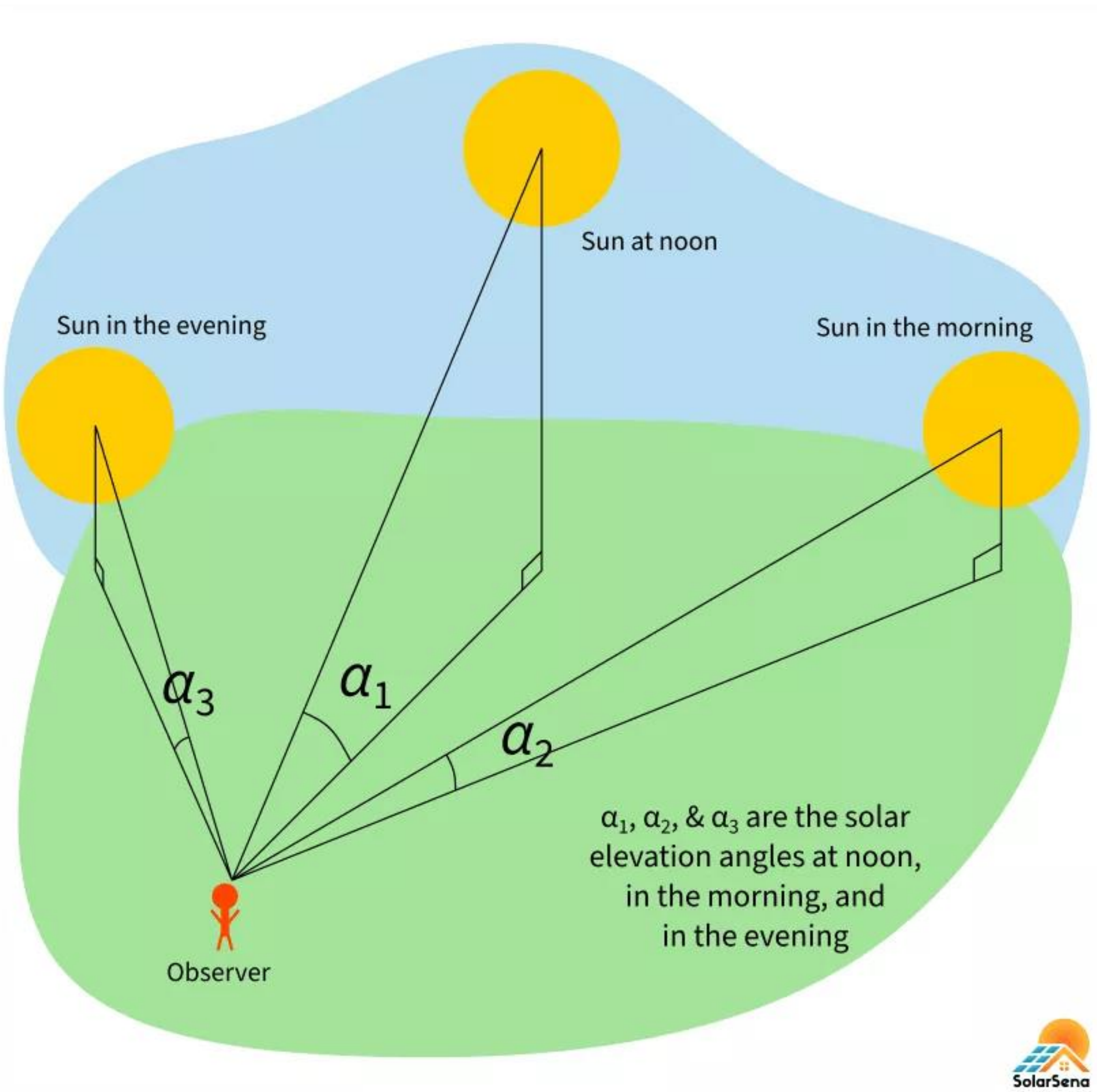

Figure source

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

[**Hint:** using the "Better formula" from Solar Declination Angle & How to Calculate it]

I suggest you to use asind and sin, instead of using toarc.

```
MODULE Declination_angle_module
    CONTAINS

    subroutine Declination_angle(days,angle)
        implicit none
        integer, intent(in)  :: days
        real(8), intent(out)    :: angle
        real(8)                 :: pi,toarc,delta

        pi = 3.1415926
        toarc = pi / 180.0

        delta = asin(sin(-23.44*toarc)*cos(toarc*(360.0/365.24*(days+10)+(360.0/pi)*0.0167*sin(3$

        angle = delta

    end subroutine Declination_angle

end MODULE Declination_angle_module
```

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.

[**Hint:** using the formulas from Solar Hour Angle & How to Calculate it]

```
MODULE Solar_hour_angle_module
    CONTAINS

    subroutine Solar_hour_angle(days,LST,lon,TZ,h)

    implicit none

    real(8), intent(in) :: lon, TZ, LST
    integer, intent(in) :: days
    real(8), intent(out) :: h
    real(8) :: pi,gamma, EoT, Offset, LST_corrected

    pi = 3.1415926

    gamma = 2*pi/365*(days-1+(LST-12)/24)

    EoT = 229.18*(0.000075+0.001868*cos(gamma)-0.032077*sin(gamma)-0.014615*cos(2*gamma)-0.04084$

    Offset = EoT + 4*(lon - 15*TZ)

    LST_corrected = LST + Offset/60

    h = 15*(LST_corrected - 12)

    end subroutine Solar_hour_angle
```

```
end MODULE Solar_hour_angle_module
```

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.

```fortran
Program Solar_elevation_angle

    USE Declination_angle_module
    USE Solar_hour_angle_module

    implicit none

    real(8), parameter :: pi = 3.1415926536
    real(8) :: lat, lon, TZ, LST, h, angle, SEA, toarc
    integer :: days,year,month,day
    integer,dimension(12) :: month_array,month_leap_array

    month_array=(/31,28,31,30,31,30,31,31,30,31,30,31/)
    month_leap_array=(/31,29,31,30,31,30,31,31,30,31,30,31/)

    toarc = pi / 180

    lat = 22.542883
    lon = 114.062996
    TZ = 8.0
    LST = 10.53333
    year = 2021
    month = 12
    day = 31
```

```fortran
    if ((mod(year,400)==0) .or. (mod(year,4)==0 .and. mod(year,100)/=0)) then
        days = sum(month_leap_array(:month-1)) + day
    else
        days = sum(month_array(:month-1)) + day
    end if

    call Declination_angle(days, angle)

    call Solar_hour_angle(days,LST, lon, TZ, h)

    SEA = asin(sin(lat*toarc)*sin(angle*toarc)+cos(lat*toarc)*cos(angle*toarc)*cos(h*toarc))*1/t$
    write(*,'(a6,f7.3)') 'SEA = ', SEA

End Program Solar_elevation_angle
```

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 `libsea.a`. Print the SEA for Shenzhen (`22.542883N, 114.062996E`) at `10:32` (Beijing time; UTC+8) on `2021-12-31`.

```
[ese-tianxj@login02 ~]$ nano Declination_angle.f90
[ese-tianxj@login02 ~]$ nano Solar_hour_angle.f90
[ese-tianxj@login02 ~]$ nano Solar_elevation_angle.f90
[ese-tianxj@login02 ~]$ gfortran -c Declination_angle.f90
[ese-tianxj@login02 ~]$ gfortran -c Solar_hour_angle.f90
[ese-tianxj@login02 ~]$ gfortran Solar_elevation_angle.f90 Declination_angle.O Solar_hour_angle.o
 -o Solar_elevation_angle.f90
gfortran: error: Declination_angle.O: No such file or directory
[ese-tianxj@login02 ~]$ gfortran -c Solar_elevation_angle.f90
[ese-tianxj@login02 ~]$ gfortran Solar_elevation_angle.f90 Declination_angle.O Solar_hour_angle.o
 -o Solar_elevation_angle.x
gfortran: error: Declination_angle.O: No such file or directory
[ese-tianxj@login02 ~]$ gfortran Solar_elevation_angle.f90 Declination_angle.o Solar_hour_angle.o
 -o Solar_elevation_angle.x
[ese-tianxj@login02 ~]$ ar rcvf libsea.a Declination_angle.o Solar_hour_angle.o
a - Declination_angle.o
a - Solar_hour_angle.o
[ese-tianxj@login02 ~]$ gfortran Solar_elevation_angle.f90 -o Solar_elevation_angle.x -L. -lsea
[ese-tianxj@login02 ~]$ ./Solar_elevation_angle.x
SEA =   36.544
```

Good, you have known how to create library and compile by using library