The fourth schoolwork of Computational Physics

万炫均 物理 1701 U201710170

Description of this chapter:

For this chapter, we try various computing methods to find the solutions of a series of linear equations. We usually use Matrices to represent the equations and transform them to get the solutions. Both transformative and iterative methods are used.

- **Description of the problem**

Homework

Use *Lagrange* and *Newton* interpolation method to rebuild the function

$$f(x)=\frac{1}{1+x^2}, x\in[-5,5]$$

based on points (N=15):

$$x_i=-5+\frac{10}{N}i,\quad i=0,1,\cdots N$$

show the results in the figure with error bar.

- **Formula to use**

  Here we will use four methods in total:

  Lagrange Interpolation

  Newton Interpolation

So the *Newton* interpolation function is

$$N_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1})$$

here $$a_n = f[x_0, \cdots, x_n]$$

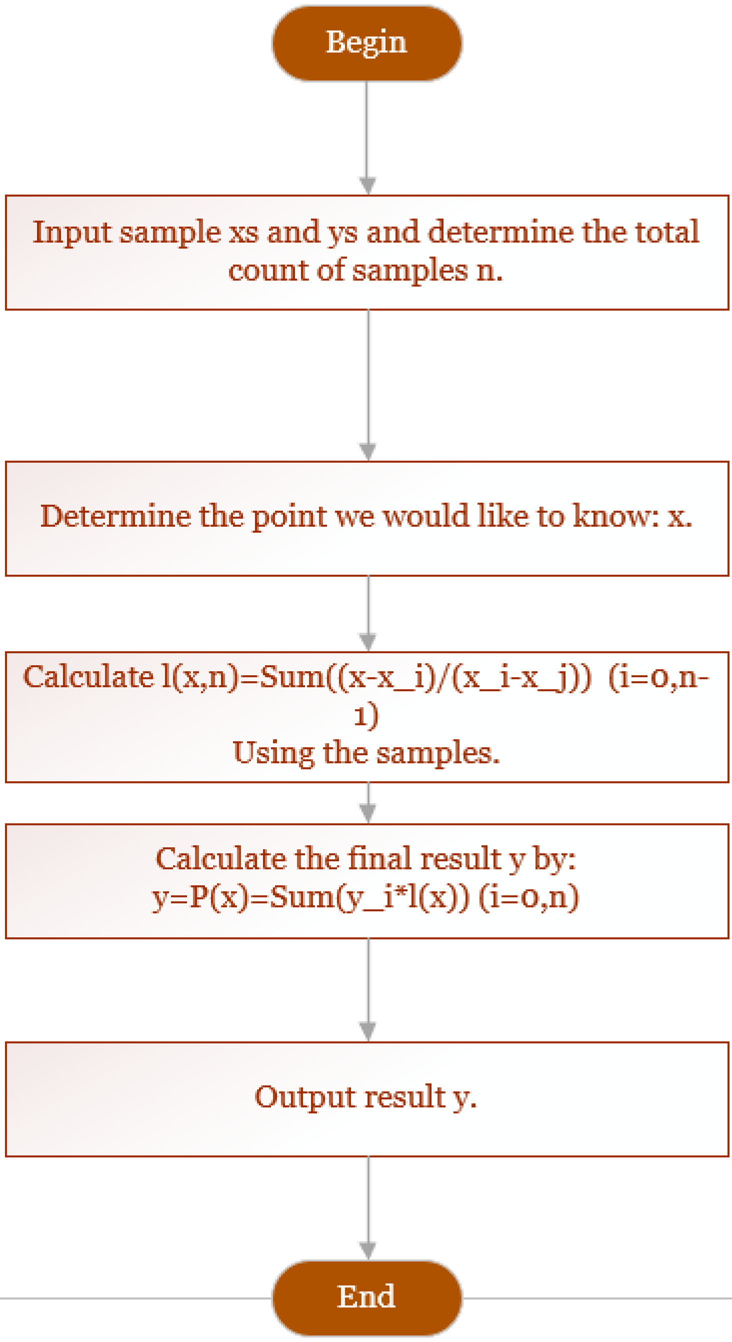
Limitations of *Lagrange* interpolation

$$\boxed{P(x) = \sum_{i=0}^{n} l_i(x) y_i}$$

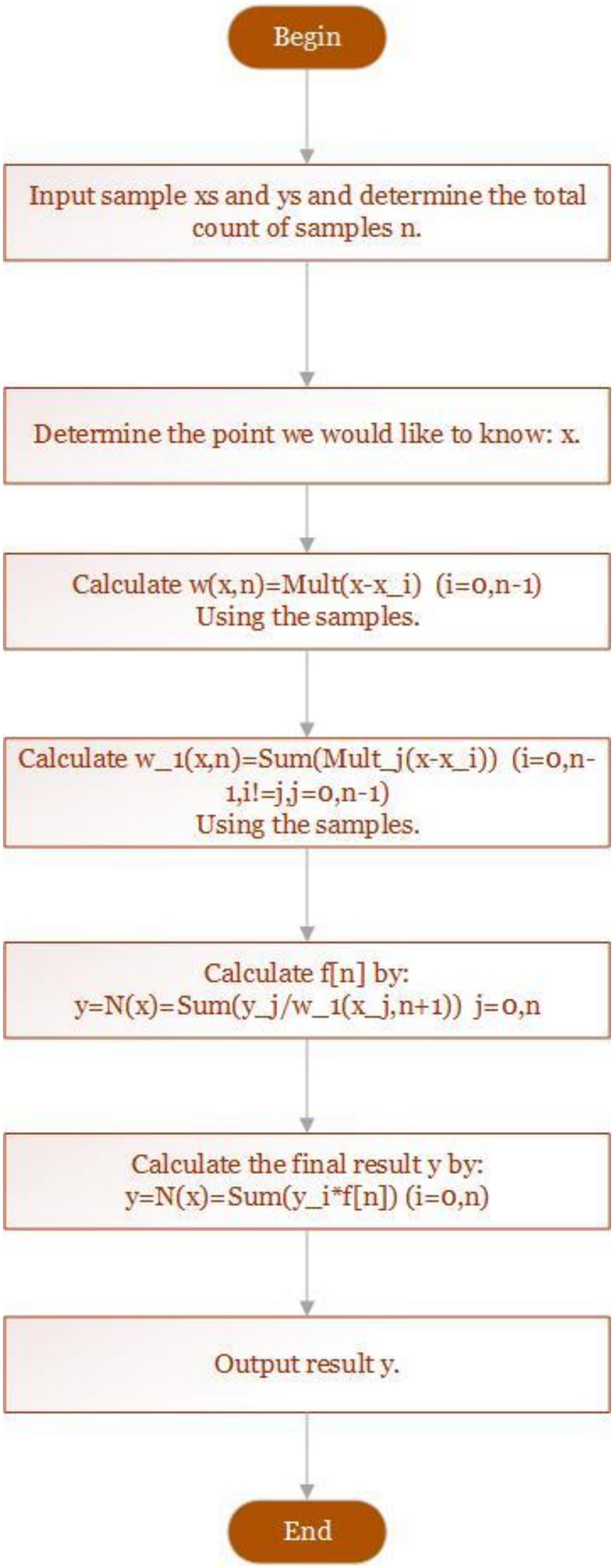
$$\boxed{l_i(x) = \frac{(x - x_0)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}}$$

Not so extensible!

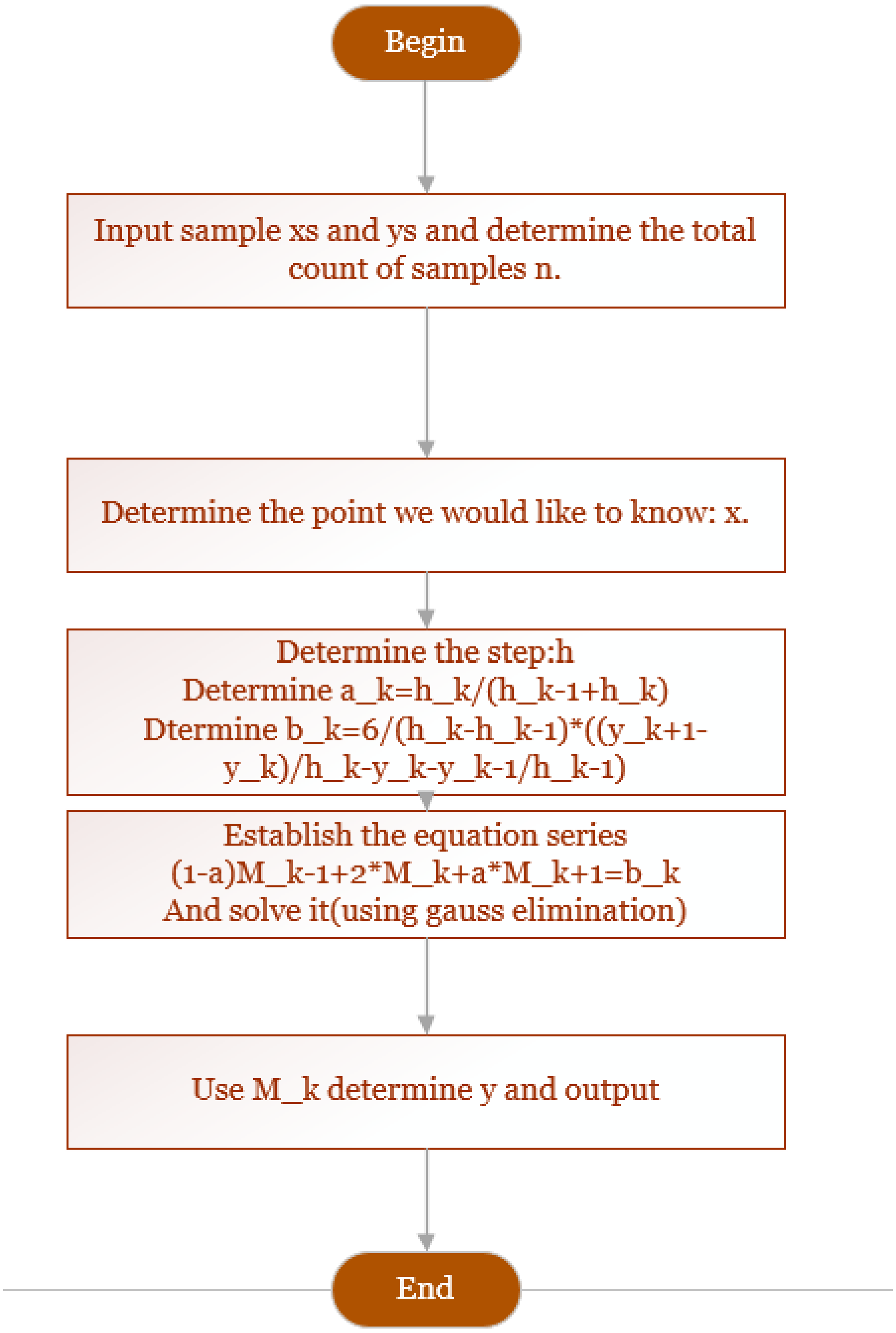
- **Flow chart**
  1. **Lagrange Interpolation Flowchart**



2. Newton Interpolation Flowchart


Begin
Input sample xs and ys and determine the total count of samples n.
Determine the point we would like to know: x.
Calculate w(x,n)=Mult(x-x_i) (i=0,n-1)
Using the samples.
Calculate w_1(x,n)=Sum(Mult_j(x-x_i)) (i=0,n-1,i!=j,j=0,n-1)
Using the samples.
Calculate f[n] by:
y=N(x)=Sum(y_j/w_1(x_j,n+1)) j=0,n
Calculate the final result y by:
y=N(x)=Sum(y_i*f[n]) (i=0,n)
Output result y.
End

3. Cubic Spline Interpolation Flowchart



- **Source Code**

```
    program Interpolation
        integer :: operation
        operation = 0

        do while(.true.)
```

```fortran
        print *,"****************************"
        print *,"Enter the operation you would like to choose to
interpolate the function:"
        print *,"The function is : f(x)=1/(1+x**2)"
        print *,"The sampling function is :x(i)=-5+10/N*i"
        print *,"1.Lagrange Interpolation"
        print *,"2.Newton Interpolation"
        print *,"3.Exit the programm"
        read *,operation
        !read the operator from the keyboard
        select case(operation)
              case (1)
                  call Lagrange()
              case (2)
                  call Newton()
              case (3)
                  exit
              case default
                  cycle
         end select
    end do

end program

!--------------------------------------------------Lagrange method and its
functions-----------------------------------------
subroutine Lagrange()
    real*8 :: x,P,f

    !Print texts
    print *,"Through sampling, the function is interpolated by Lagrange
Interpolation."
    print *,"The function is shape downbelow with the interval of 0.1 in
range [-5,5]"

    !Open files
    open(file="lagrange_y_real.txt",unit=10)
    open(file="lagrange_y.txt",unit=11)
    open(file="lagrange_x.txt",unit=12)

    !Core calculation
    do x=-5,5,0.1
        print *,"-----"
        print "(a,es10.3)","X value:",x
```

```
        print "(a,es10.3)","Predicted value:",P(x,15)
        print "(a,es10.3)","Function value:",f(x)
        !Write data into files
        write(10,"(es10.3)")f(x)
        write(11,"(es10.3)")P(x,15)
        write(12,"(es10.3)")x

    end do

    close(10)
    close(11)
    close(12)
end subroutine

function l(x,i,N)
    real*8 :: l,x,x_sample
    integer :: i,N
    l=1
    do j=0,N
        if (j==i) then
            cycle
        else
            l=l*(x-x_sample(j,N))/(x_sample(i,N)-x_sample(j,N))
        end if
    end do
end function

function P(x,N)
    real*8 :: P,x,x_sample,l,f
    integer :: N
    P=0
    do i=0,N
        P=P+l(x,i,N)*f(x_sample(i,N))
    end do
end function

!------------------------------------------------------------Newton
method and its functions--------------------------------
subroutine Newton()
    real*8 :: x,f,Newt

    !Print texts
    print *,"Through sampling, the function is interpolated by Newton
Interpolation."
```

```fortran
    print *,"The function is shape downbelow with the interval of 0.1 in
range [-5,5]"

    !Open files
    open(file="newton_y_real.txt",unit=10)
    open(file="newton_y.txt",unit=11)
    open(file="newton_x.txt",unit=12)

    !Core calculation
    do x=-5,5,0.1
        print *,"-----"
        print "(a,es10.3)","X value:",x
        print "(a,es10.3)","Predicted value:",Newt(x,15)
        print "(a,es10.3)","Function value:",f(x)
        !Write data into files
        write(10,"(es10.3)")f(x)
        write(11,"(es10.3)")Newt(x,15)
        write(12,"(es10.3)")x
    end do

    close(10)
    close(11)
    close(12)

end subroutine


function Newt(x,N)
    real*8::F_DevDivN,w,x,Newt
    integer :: N
    Newt=0
    do i=0,N
        Newt=Newt+F_DevDivN(i,N)*w(i-1,x,N)
    end do

end function

function F_DevDivN(nn,N)
    real*8 :: F_DevDivN,x_sample,f,w_1
    integer :: nn,N

    F_DevDivN=0
    do i=0,nn
        F_DevDivN=F_DevDivN+f(x_sample(i,N))/w_1(i,x_sample(i,N),N)
```

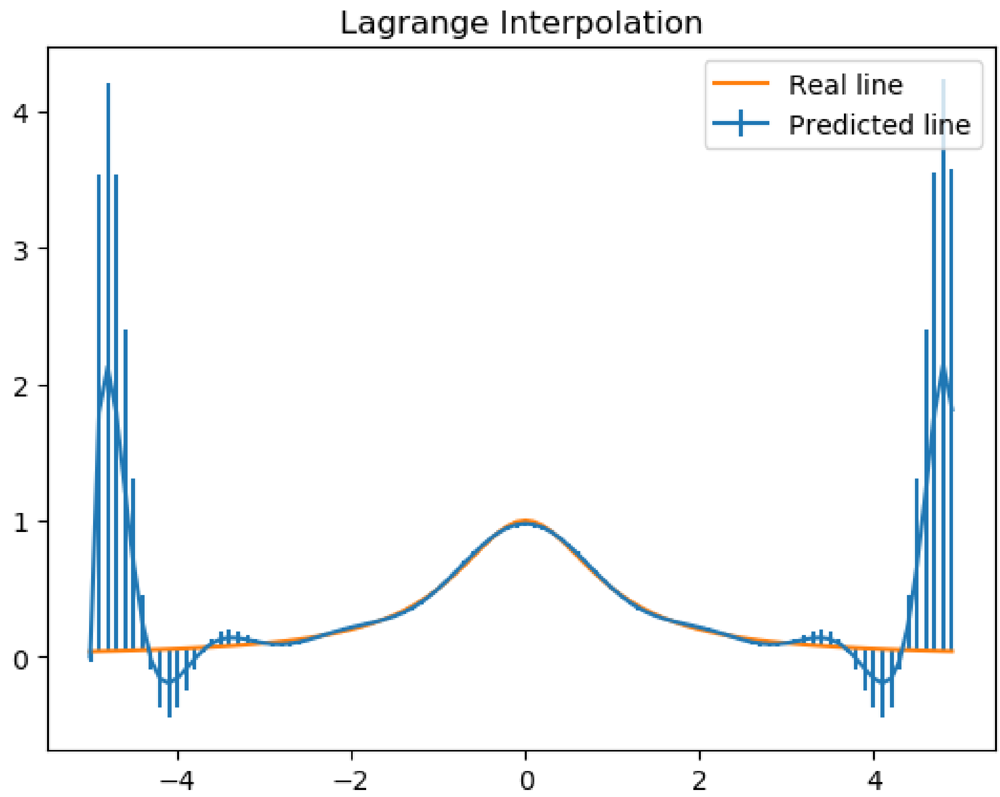
```
    end do

end function

function w(nn,x,N)
    real*8 :: w,x,x_sample
    integer :: N,nn
    w=1
    do i=0,nn
        w=w*(x-x_sample(i,N))
    end do
end function

function w_1(nn,x,N)
    real*8 :: w_1,x,k,x_sample
    integer :: N,nn
    w_1=0
    do i=0,nn
        k=1
        do j=0,nn
            if (.not.i==j) then
                k=k*(x-x_sample(j,N))
            end if
        end do
        w_1=w_1+k
    end do

end function

!------------------------------------------------------------Cubic
Spline and its functions-------------------------------
subroutine Spline(x,N)
    !Input
    real*8,intent(in) :: x
    integer,intent(in) :: N
    !Output
    real*8 :: Newt
    !Used functions
    real*8 :: x_sample,y,y_1
    !Local vars
    real*8 ::
x_samples(16),y_samples(16),h(15),a(15),b(15),c(15,1),D(15,15)

    do i=0,14
```

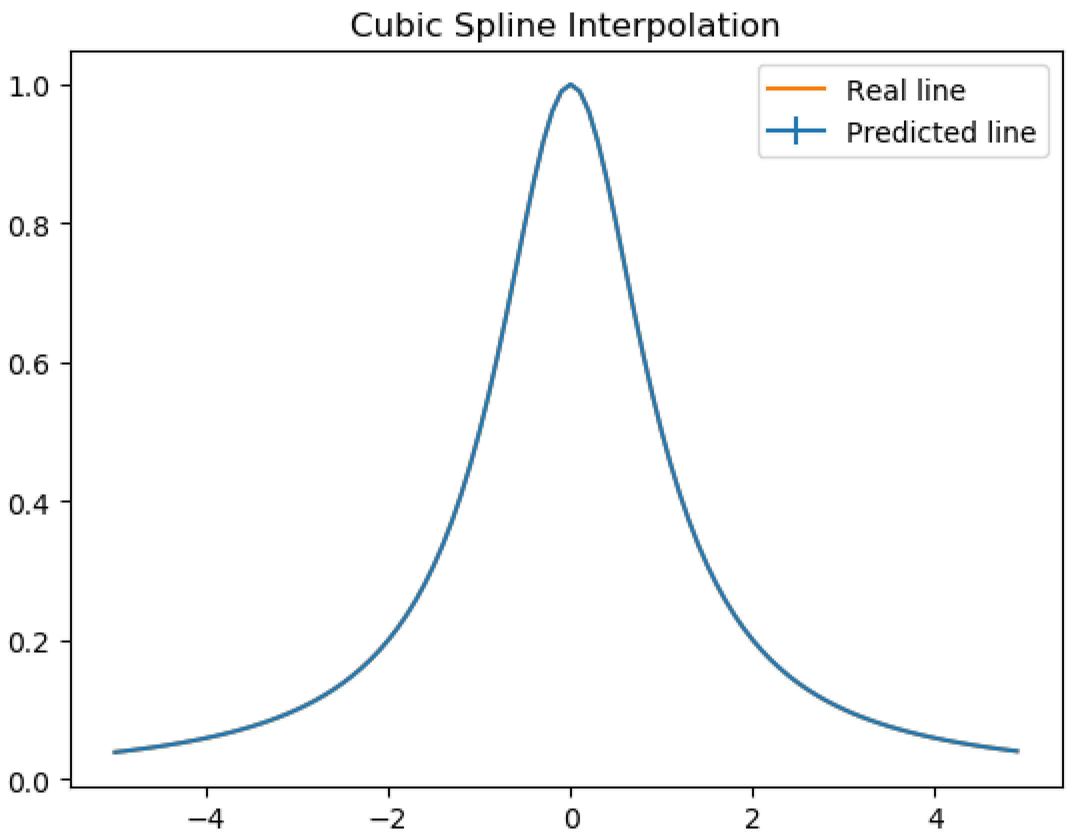
```
        h(i)=x_samples(i+1)-x_samples(i)
    end do

    do i=0,14
        a(i)=h(i)/(h(i)+h(i+1))
    end do
    do i=0,14
        if (i==0)then
            b(i)=3*y_1(-5)
        else if(i==14)then
            b(i)=3*y_1(5)
        else
            b(i)=3*(1-a(i))*((y_samples(i)-y_samples(i-1))/h(i-1))+
    (a(i)*(y_samples(i+1)-y_samples(i))/h(i))
        end if

    end do
    !setup D matrix
    do i=0,14
        D(i,i)=2
        if(i>0) then
            D(i,i-1)=1-a(i)
        end if
        if(i<14) then
            D(i,i+1)=a(i)
        end if
    end do

    c = reshape( b, (/ 15, 1 /) )
    call GaussElimination(D,b)



end subroutine
subroutine GaussElimination(A, b)
    real*8 :: A(15,15),b(15,1)
    real*8 :: factor,A_temp(15,15)

    !Creating copies of parameters in case of reference affecting
    A_temp = A

    do i=1,15
        !Cast the diag elements to unit 1
        factor = A_temp(i,i)
        do j=1,15
```

```
            A_temp(i,j) = A_temp(i,j)/factor
        end do
        b(i,1) = b(i,1)/factor

        !Eliminate bottom triangle
        do j = i+1,15
            factor = A_temp(j,i)
            do k = i,15
                A_temp(j,k) = A_temp(j,k) - factor*A_temp(i,k)
            end do
            b(j,1) = b(j,1) - factor*b(i,1)
        end do

    end do

    !Eliminate upper triangle
    do i=1,15
        do j=i+1,15
            factor = A_temp(16-j,16-i)
            do k = 16-i,15
                A_temp(16-j,k) = A_temp(16-j,k) - factor*A_temp(16-i,k)
            end do
            b(16-j,1) = b(16-j,1) - factor*b(16-i,1)
        end do
     end do
end subroutine
!-------------------------------------------------------------------------
-Basic Definition----------------------------
!Function definition
function f(x)
     real*8 :: f
     real*8 :: x
     f=1/(1+x**2)
end function
!Sampling definition
function x_sample(i,N)
     real*8 :: x_sample
     integer :: i,N
     if (i>N) then
          i=N
     else if (i<0) then
          i=0
     end if
     x_sample=-5+dble(10)/N*i
```

```
end function
```

- **Example and Result**





- **Demo**
  Check the folder ”Interpolation” in the directory and follow the instruction to set up the matrices and vectors