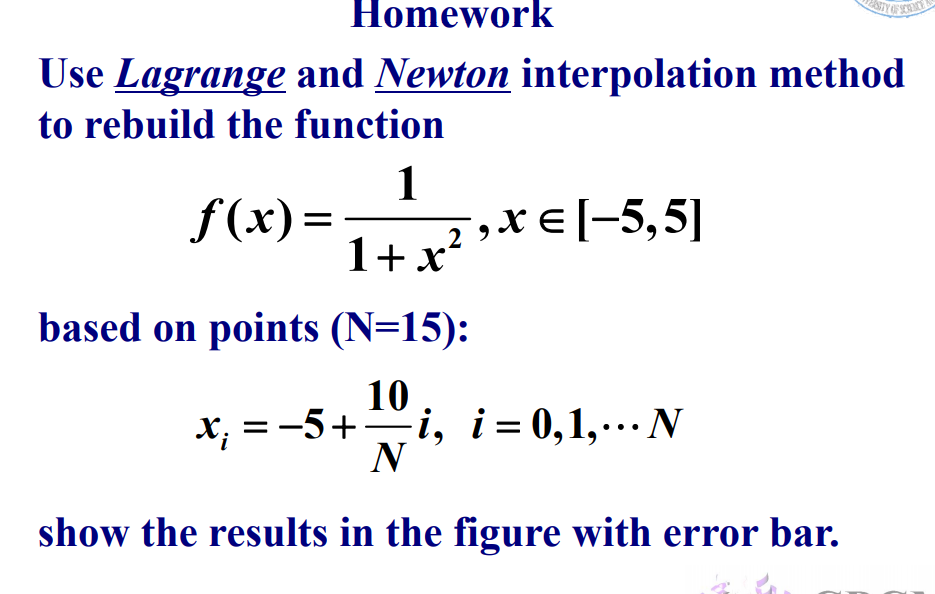
# The fourth schoolwork of Computational Physics

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**

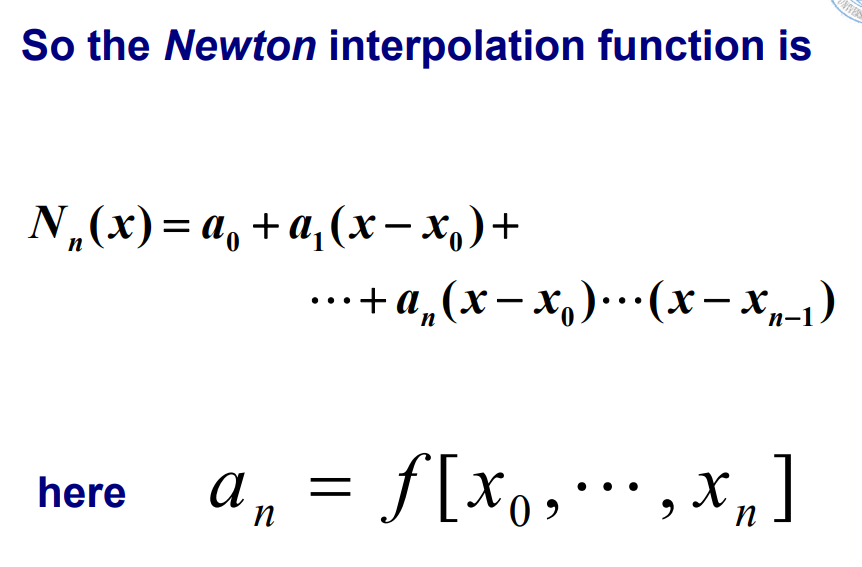


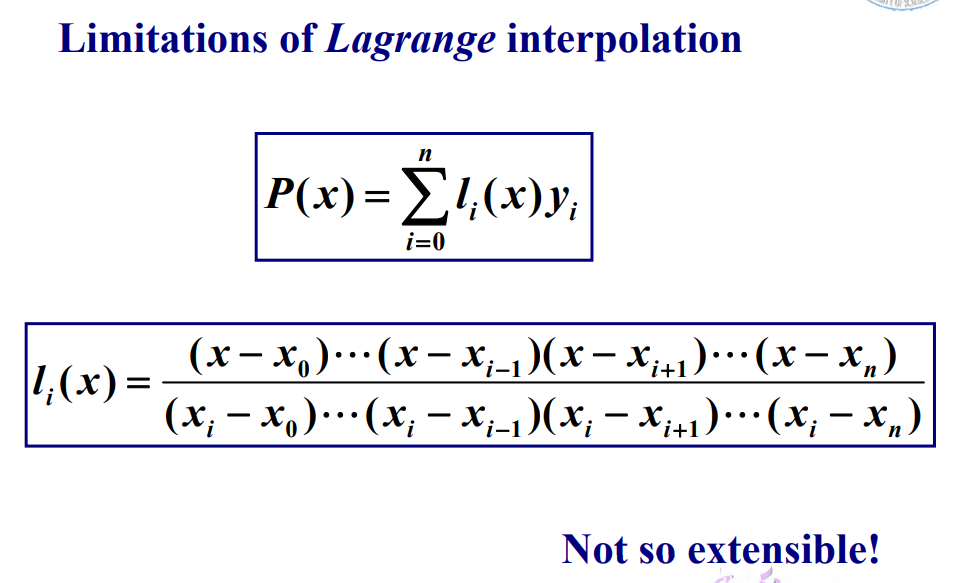
* **Formula to use**

Here we will use four methods in total:

Lagrange Interpolation

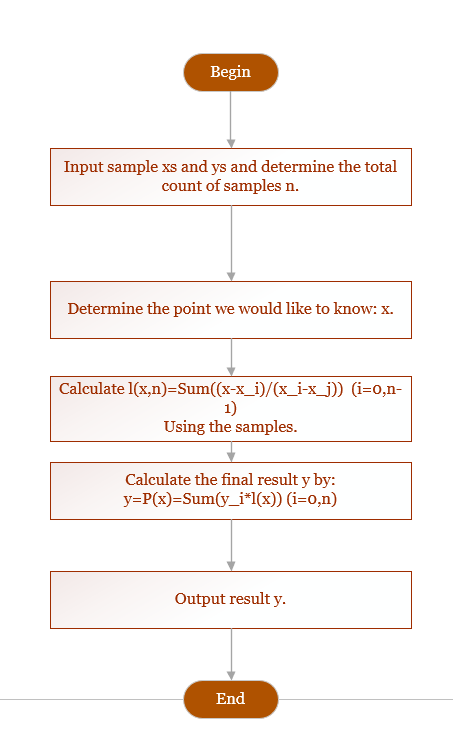
Newton Interpolation



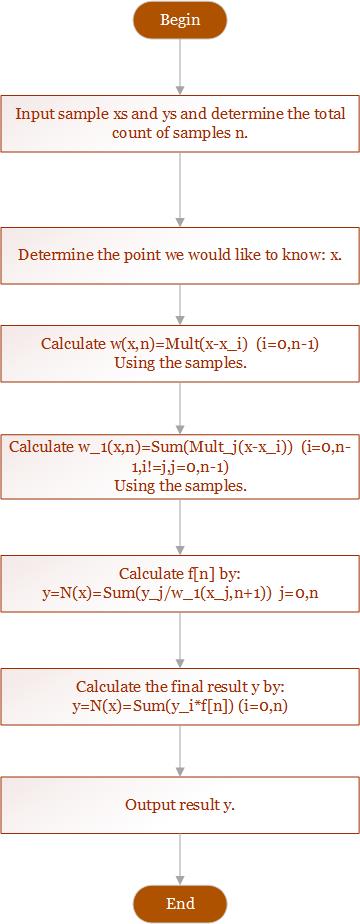


* **Flow chart**

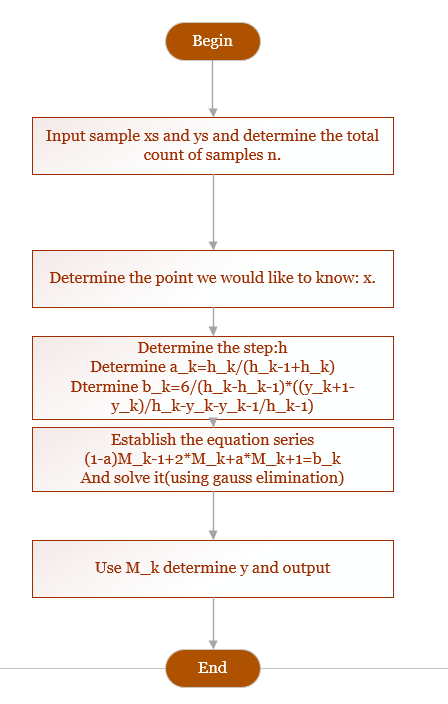
1. **Lagrange Interpolation Flowchart**



1. **Newton Interpolation Flowchart**

****

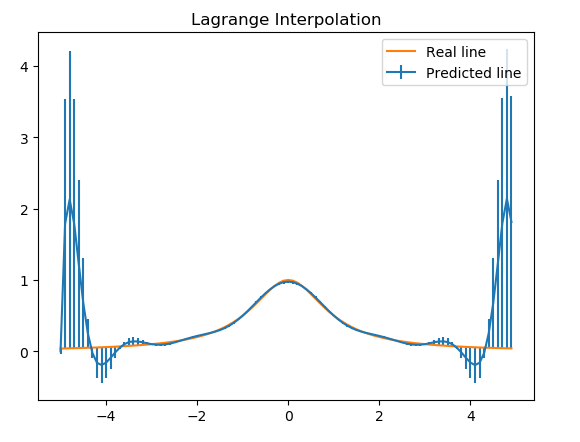
1. **Cubic Spline Interpolation Flowchart**

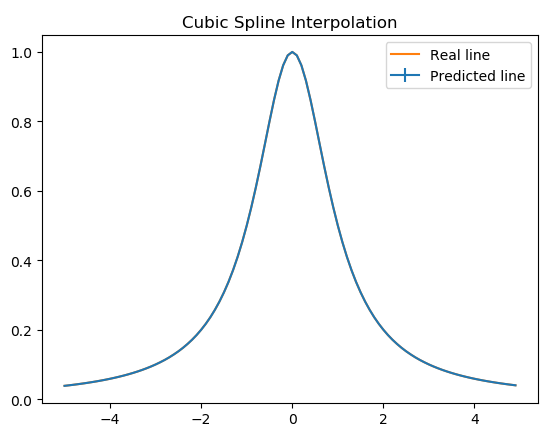


* **Source Code**
* 窗体顶端
* 窗体底端
* 窗体顶端
* 窗体底端

|  |  |
| --- | --- |
|  | program Interpolation |
|  | integer :: operation |
|  | operation = 0 |
|  |  |
|  | do while(.true.) |
|  | 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." |
|  | 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