NUMERICAL PRACTICE -1: NUMERICAL INTERPOLATION

Bipin Kumar Singh

November 12, 2017

1 THEORY:INTERPOLATION

To determine intermediate value between precise data points.The most common method used for this purpose is polynomial interpolation.

$$f(x) = a0 + a1 * x + a2 * x^2 + - - - - - - - + an * x^n$$

For $n + 1$ data points, there is one and only one polynomial of order n that passes through all the points. For example, there is only one straight line (that is, a first-order polynomial) that connects two points. Similarly, only one parabola connects a set of three points. Polynomial interpolation consists of determining the unique nth-order polynomial that fits $n + 1$ data points. This polynomial then provides a formula to compute intermediate values.

1.1 NEWTON'S DIVIDING DIFFERENCE INTERPOLATION METHOD

There are a variety of alternative forms for expressing an interpolating polynomial. Newton?s divided-difference interpolating polynomial is among the most popular and useful forms.

The generalized equation to fit an nth-order polynomial to n + 1 data points.The nth-order polynomial as per Newton's method is:

$$fn(x) = b0+b1*(x-x0)+---------+bn*(x-x0)*(x-x1)---------(x-xn-1)$$

Data points can be used to evaluate the coefficients b 0 , b 1 , —- , b n . For an nth-order polynomial, n + 1 data points are required: $[x0, f(x0)], [x1, f(x1)], ---, [xn, f(xn)]$. We use these data points and the following equations to evaluate the coefficients:
$b0 = f(x0)$
$b1 = f[x1, x0]$
$b2 = f[x2, x1, x0]$
-
-
-
$bn = f[xn, xn - 1, ..., x1, x0]$

where the bracketed function evaluations are finite divided differences. For example, the first finite divided difference is represented generally as

$$f[xi, xj] = f(xi) - f(xj)/(xi - xj)$$

Similarly, the nth finite divided difference is

$$f[xn, xn{-}1, ---, x1, x0] = f[xn, x{-}1, ---, x1]{-}f[xn{-}1, xn{-}2, --, x0]/(xn{-}x0)$$

These coefficients can be evaluated and put in the equation to yield the interpolation polynomial.

1.2 LAGRANGE INTERPOLATION METHOD

The Lagrange interpolating polynomial is simply a reformulation of the Newton polynomial that avoids the computation of divided differences. It can be represented concisely as

$$f(x) = \sum_{i=o}^{n} Li(x) * f(xi)$$

where

$$Li(x) = \prod_{j=o,i}^{b} f(i)$$

For example, the linear version (n = 1) is

$$f1(x) = (x - x1) * f(x0)/(x0 - x1) + (x - x0) * f(x1)/(x1 - x0)$$

the second-order version is

$$f2(x) = (x{-}x1){*}(x{-}x2){*}f(x0)/(x0{-}x1)(x0{-}x2){+}(x{-}x0){*}(x{-}x2){*}f(x1)/(x1{-}x0)(x1{-}x2)$$

$$+(x - x0) * (x - x1) * f(x2)/(x2 - x0)(x2 - x1)$$

2 ALGORITHM:

2.1 NEWTON'S DIVIDED DIFFERENCE

3 PROGRAM CODE:

3.1 LAGRANGE INTERPOLATION METHOD

Main Program:
FUNCTION: $f(x) = 1/(1 + 25x^2)$

```
program Mainfunction1 !program start
use Newton
use lagrange  !to call the module in the main program
implicit none!to force the programmer to define all the variables

real(8),dimension(0:4)::X4,Y14 !(X4:Datapoints for 4 order,Y14: Function va
```

```
real(8),dimension(0:9)::X9,Y19,Lagint9,Lagint4,XX,Lagrangey1,error14,error1
,Newint4,error14n,Newint9,error19n !(X9:Datapoints for 9 points,Y19: Functi
integer::i,j,N,k
character(len=30)::myfilename
real(8)::A,B,H
real(8), parameter::pi=3.14

B=1 !Max range
A=-1!min range

X4(:)=(/-1.0,-0.5,0.0,0.5,1.0/)
!datapoints for 4th order
X9(:)=(/-1.0,-0.8,-0.6,-0.4,-0.2,0.0,0.2,0.4,0.6,0.8/)
!datapoints for 9th order
Y14(:)=1/(1+(25*(X4(:)**2)))
!values defined at data points
Y19(:)=1/(1+(25*(X9(:)**2)))
!values defined at data points
N=9
!number of points at which the interpolation needs to be done


XX(:)=(/-0.93,-0.77,-0.64,-0.35,-0.23,0.0,0.13,0.26,0.53,0.76/)

do i=0,N
Lagint4(i)=lgrnge(X4,Y14,4,xx(i))
end do
do i=0,N
Newint4(i)=order4(X4,Y14,4,xx(i))
end do
do j=0,N
Lagint9(i)=lgrnge(X9,Y19,9,xx(i))
end do
do j=0,N
Newint9(i)=order4(X9,Y19,9,xx(i))
end do

Lagrangey1=1/(1+(25*(xx(:)**2)))

myfilename='function1.dat'
open(99,file=myfilename)
write(99,*)'for function f(x)=1/(1+25x^2), X VALUES'
write(99,*)xx
write(99,*)'for function f(x)=1/(1+25x^2), Y VALUES'
write(99,*)lagrangey1
write(99,*)'for LAGRANGE'
write(99,*)'4th order'
write(99,*)Lagint4
write(99,*)'9th order'
write(99,*)Lagint9
```

```
write(99,*)'for␣NEWTON'
write(99,*)'for␣function␣f(x)=1/(1+25x^2)'
write(99,*)'4th␣order'
write(99,*)Newint4
write(99,*)'9th␣order'
write(99,*)Newint9
close(99)


end Program Mainfunction1
```

FUNCTION: $F(X) = cos(x(:) * pi)$ //

```
program MainNI !program start
use Newton
use lagrange  !to call the module in the main program
implicit none!to force the programmer to define all the variables

real(8),dimension(0:4)::X4,Y24 !(X4:Datapoints for 4 order,Y14: Function va
real(8),dimension(0:9)::X9,Lagint9,Lagint4,xx,Lagrangey2,error24,&
error29,Y29,Newint4,error14n,error24n,Newint9,error19n,error29n !(X9:Datapo
integer::i,j,N,k
character(len=30)::myfilename
real(8)::A,B,H
real(8), parameter::pi=3.14

B=1 !Max range
A=-1!min range

xx(:)=(/-0.93,-0.77,-0.64,-0.35,-0.23,0.0,0.13,0.26,0.53,0.76/)


X4(:)=(/-1.0,-0.5,0.0,0.5,1.0/)
!datapoints for 4th order
X9(:)=(/-1.0,-0.8,-0.6,-0.4,-0.2,0.0,0.2,0.4,0.6,0.8/)
!datapoints for 9th order
Y24(:)=cos(X4(:)*pi)
!values defined at data points
Y29(:)=cos(X9(:)*pi)
!values defined at data points
N=9
!number of points at which the interpolation needs to be done

xx(:)=(/-0.93,-0.77,-0.64,-0.35,-0.23,0.0,0.13,0.26,0.53,0.76/)
Lagrangey2(:)=cos(xx(:)*pi)

do i=0,N
Lagint4(i)=lgrnge(X4,Y24,4,xx(i))
end do

do i=0,N
```

```
Newint4(i)=order4(X4,Y24,4,xx(i))
end do

do j=0,N
Lagint9(i)=lgrnge(X9,Y29,9,xx(i))
end do
do j=0,N
Newint9(i)=order4(X9,Y29,9,xx(i))
end do

myfilename='function2.dat'
open(99,file=myfilename)

write(99,*)'for␣LAGRANGE'
write(99,*)'for␣function␣f(x)=cos(x*pi),␣X␣VALUES'
write(99,*)xx
write(99,*)'for␣function␣f(x)=cos(x*pi),␣Y␣VALUES'
write(99,*)lagrangey2
write(99,*)'4th␣order'
write(99,*)Lagint4
write(99,*)'9th␣order'
write(99,*)Lagint9
write(99,*)'for␣NEWTON'
write(99,*)'for␣function␣f(x)=cos(x*pi)'
write(99,*)'4th␣order'
write(99,*)Newint4
write(99,*)'9th␣order'
write(99,*)Newint9
close(99)


end Program MainNI
```

MODULE LAGRANGE

```
module lagrange
implicit none

contains

function lgrnge(X,Y,N,XX)
real(8),dimension(0:N)::X,Y,Multiply !X: datapoints at which Interpolation
integer::i,j,N !i,j are do loop integers and N is the order of the input.
real(8)::lgrnge,total,XX !interpolated value of the xx input to the functio

total=0

do i=0,N
Multiply(i)=Y(i)

do j=0,N
```

```
if (j==i) cycle
Multiply(i)=Multiply(i)*(XX-X(j))/(X(i)-X(j))
end do
total=total+Multiply(i)
end do
lgrnge=total
end function lgrnge

end module lagrange
```

MODULE NEWTON

```
module Newton
implicit none

contains
!N=number of data point=+1 the order of the polynomial used for interpolati


function order4(X,Y,N,XX)
real(8),allocatable:: totala(:),roots(:)
real(8),dimension(0:N)::X,Y
real(8),dimension(0:N+1,0:N)::R
real(8)::total,order4,XX
real(8)::multiplier
integer::N,i,j,k,l,m,o,p

allocate(totala(N+1))

allocate(roots(N+1))


do i=2,N+1
do k=0,N
R(0,k)=X(k)
end do
do l=0,N
R(1,l)=Y(l)
end do

do j=0,N
if (j>=N+2-i) then
R(i,j)=0
else
R(i,j)=(R(i-1,j+1)-R(i-1,j))/(R(1,i-3+j+1)-R(1,j))
end if
end do
end do

do m=1,N+1
roots(m)=R(1,m)
```

```
end do


do o=2,N+1
multiplier=roots(o)
totala(1)=roots(1)
do p=2,o+1
totala(o)=multiplier*(XX-X((p-1)))
end do
end do
order4=sum(totala)

end function

end module Newton
```

4 RESULT:

For f(x)=1/(1+25x2)

S.No.	XX	YY	NEWINT4	NEWINT9	LAG
1	-0.9300000072	0.0442037788	-2.6470587841676605	4.42E-02	3.846153846
2	-0.7699999809	0.0632011406	-0.87531827994936018	6.32E-02	-0.3703931
3	-0.6399999857	0.0889679752	-0.72238725107607848	8.90E-02	-0.1956625
4	-0.349999994	0.2461538525	-0.38123341473881700	0.2461538525	0.5257999
5	-0.2300000042	0.430570497	-0.24006631790563979	0.430570497	0.7830152
6	0	1	0.9946949602	3.0503978779840846E-002	1.0000000
7	0.1299999952	0.702987713	0.7101839655	0.702987713	0.9286625
8	0.2599999905	0.371747229	0.4050835668	0.371747229	0.7260138
9	0.5299999714	0.1246494353	8.13E-02	0.1246494353	6.015859444
10	0.7599999905	0.0647668409	-0.587082452	6.48E-02	-0.3643310

For f(x)=cos(pi*x)

S.No.	XX	YY	NEWINT4	NEWINT9	LAGINT4	LAGINT9
1	-0.9300000072	-0.9755925897	-9.76E-01	-9.76E-01	-0.987357316	-0.9755926111
2	-0.7699999809	-0.7492994708	-1.45E+00	-7.49E-01	0.1260187557	-0.7492995243
3	-0.6399999857	-0.4248567412	-0.6604939021	-4.25E-01	0.2975304491	-0.424856802
4	-0.349999994	0.4544871185	0.4969066814	0.4544870858	0.8357205005	0.4544870858
5	-0.2300000042	0.750353256	0.7778651441	0.7503532401	0.990419558	0.7503532401
6	0	1	0.9946949602	1	0.9694960212	1
7	0.1299999952	0.9178368394	0.9250330865	0.917836834	0.7344018264	0.917836834
8	0.2599999905	0.6848489276	0.7181852455	0.6848489077	0.3484828712	0.6848489077
9	0.5299999714	-0.0932678302	-1.37E-01	-9.33E-02	-0.7472598944	-9.33E-02
10	0.7599999905	-0.7281394858	-1.3799888334	-7.28E-01	-1.6527018635	-0.7281395405