Numerical Analysis HW 5

Michael Vu

August 31, 2023

Problem 1. Using a divided difference program, I recreated the table 3.9 from our textbook with this code:

```
program hw5pr1
implicit none
integer :: n, i, k
double precision :: x(0:4), f(0:4), a(0:4)
double precision :: t(0:4, 0:4), top, bottom

x(0) = 1.0d0
x(1) = 1.3d0
x(2) = 1.6d0
x(3) = 1.9d0
x(4) = 2.2d0

f(0) = 0.7651977d0
f(1) = 0.6200860d0
f(2) = 0.4554022d0
f(3) = 0.2818186d0
f(4) = 0.1103623d0

do i = 0, 4
t(i, 0) = f(i)
end do

do k = 1, 4
do i = k, 4
```

```
top = t(i, k-1) - t(i-1, k-1)
bottom = x(i) - x(i-k)
t(i, k) = top / bottom
end do
end do

write(*,6) t(0,0)
write(*,6) t(1,0:1)
write(*,6) t(2,0:2)
write(*,6) t(3,0:3)
write(*,6) t(4,0:4)

6 format(7(1x,g13.6))

end program
```

The code and inputs from the textbook match those of table 3.9. Below is the output:

```
0.765198
0.620086     -0.483706
0.455402     -0.548946     -0.108734
0.281819     -0.578612     -0.494433E-01  0.658784E-01
0.110362     -0.571521      0.118183E-01  0.680685E-01  0.182510E-02
```

Problem 2. Next we were asked to solve problem number 3 from Section 3.2 using the above code and our knowledge of polynomial interpolation. Below is the sample of code used for Part A of the problem (parts B-C use the same code but with different x_i and $f(x_i)$ values for $i = 0, 1, 2, 3, 4$):

```
program hw5pr2a
implicit none
integer :: n, i, k
double precision :: x(0:4), f(0:4), a(0:4)
double precision :: t(0:4, 0:4), top, bottom

n = 3
```

```
x(0) = 8.6d0
x(1) = 8.3d0
x(2) = 8.1d0
x(3) = 8.7d0
x(4) = 8.4d0

f(0) = 18.50515d0
f(1) = 17.56492d0
f(2) = 16.94410d0
f(3) = 18.82091d0

do i = 0, n
t(i, 0) = f(i)
end do

do k = 1, n
do i = k, n
top = t(i, k-1) - t(i-1, k-1)
bottom = x(i) - x(i-k)
t(i, k) = top / bottom
end do
end do

do i = 0, n
a(i) = t(i, i)
end do

f(4) = f(0)+(x(4)-x(0))*a(1)
print*, 'degree 1 f(8.4) =', f(4)

f(4) = f(0)+(x(4)-x(0))*a(1)+(x(4)-x(0))*(x(4)-x(1))*a(2)
print*, 'degree 2 f(8.4) =', f(4)

f(4) = f(0)+(x(4)-x(0))*a(1)+(x(4)-x(0))*(x(4)-x(1))*a(2)
       +(x(4)-x(0))*(x(4)-x(1))*(x(4)-x(2))*a(3)
print*, 'degree 3 f(8.4) =', f(4)

end program
```

Here are outputs from part A (they match the book):

```
degree 1 f(8.4) =   17.878330000000002
degree 2 f(8.4) =   17.877130000000001
degree 3 f(8.4) =   17.877142500000001
```

Here are outputs from part B (they match the book):

```
degree 1 f(-1/3) =  0.21504166666666669
degree 2 f(-1/3) =  0.16988888888888892
degree 3 f(-1/3) =  0.17451851851851855
```

Here is the code and outputs from part C (degree 3 did not match):
Code:

```
program hw5pr2c
implicit none
integer :: n, i, k
double precision :: x(0:4), f(0:4), a(0:4)
double precision :: t(0:4, 0:4), top, bottom

n = 3

x(0) = 0.3d0
x(1) = 0.2d0
x(2) = 0.1d0
x(3) = 0.4d0
x(4) = 0.25d0

f(0) = 0.00660095d0
f(1) = -0.28398668d0
f(2) = 0.62049958d0
f(3) = 0.24842440d0

do i = 0, n
t(i, 0) = f(i)
end do

do k = 1, n
```

```
do i = k, n
top = t(i, k-1) - t(i-1, k-1)
bottom = x(i) - x(i-k)
t(i, k) = top / bottom
end do
end do

do i = 0, n
a(i) = t(i, i)
end do

f(4) = f(0)+(x(4)-x(0))*a(1)
print*, 'degree 1 f(0.25) =', f(4)

f(4) = f(0)+(x(4)-x(0))*a(1)+(x(4)-x(0))*(x(4)-x(1))*a(2)
print*, 'degree 2 f(0.25) =', f(4)

f(4) = f(0)+(x(4)-x(0))*a(1)+(x(4)-x(0))*(x(4)-x(1))*a(2)
       +(x(4)-x(0))*(x(4)-x(1))*(x(4)-x(2))*a(3)
print*, 'degree 3 f(0.25) =', f(4)

end program
```

Outputs (I verified over and over and could not reproduce the book result for degree 3 = -0.13277477):

```
degree 1 f(0.25) = -0.13869286500000000
degree 2 f(0.25) = -0.13259734249999999
degree 3 f(0.25) = -0.21033722187499998
```

Here are outputs from part D (they match the book):

```
degree 1 f(0.9) =  0.44086279499999997
degree 2 f(0.9) =  0.43841351999999989
degree 3 f(0.9) =  0.44198500249999984
```

Problem 3. Next I used an interpolation form with trig functions and data set, both given by Dr. Glunt. Below are the codes for main and additional subroutines:

Main:

```
implicit none
double precision, allocatable :: c(:,:),b(:),a(:),xdata(:),ydata(:)
double precision :: p, x
integer :: n, i, j, nplot

open(unit=11,file='interpdata',status='old')

read(11,*) n
print*, 'n = ', n

allocate(c(0:n,0:n),b(0:n),a(0:n),xdata(0:n),ydata(0:n))

do i = 0,n
read(11,*) xdata(i), ydata(i)
end do

close(11)

print*, 'Forming matrix'

do i = 0,n
do j = 0,n
c(i,j) = cos(j*xdata(i))
end do
end do

b = ydata

print*, 'Matrix '
print*
do i = 0,n
write(*,6) (c(i,j),j=0,n)
end do

6 format(7(1x,g13.6))

print*
print*, 'right sides of equations'
```

```
do i = 0,n
write(*,*) b(i)
end do

print*, 'Calling solver'

call solver(n+1,c,b,a)
print*, 'coefficients in interp polynomial'
print*, 'smallest indexed to biggest'

do i = 0,n
write(*,*) a(i)
end do

print*, 'Sanity check printing x, p(x), and p(x)-ydata '
do i = 0,n
x = xdata(i)
print*, x,p(n,x,a),p(n,x,a)-ydata(i)
end do

nplot = 100
call makeplotfiles(n,xdata,ydata,nplot,a)

print*, 'Freeing memory'
deallocate(c,b,xdata,ydata)
stop
end
```

Subroutine 1:

```
double precision function p(n,x,a)
implicit none
integer :: n, i
double precision :: x, a(0:n)

p = a(0)
do i = 1, n
```

```
p = p + a(i) * cos(dble(i)*x)
end do

return
end
```

Subroutine 2:

```
subroutine makeplotfiles(n,xdata,ydata,nplot,a)
implicit none
integer :: n, nplot, i
double precision :: xdata(0:n), ydata(0:n), a(0:n), p
double precision :: dx, big, small
double precision :: x, y

open(unit=11,file='plota',status='replace')

do i = 0,n
write(11,*) xdata(i), ydata(i)
end do

close(11)

big = xdata(0)
small  = xdata(0)

do i = 1,n
if (xdata(i) > big) big = xdata(i)
if (xdata(i) < small) small = xdata(i)
end do

open(unit=11,file='plotb',status='replace')
dx = (big-small) / dble(nplot)

do i = 0,nplot
x = small + dble(i)*dx
y = p(n,x,a)
write(11,*) x,y
```

```
end do

close(11)
return
end
```

Subroutine 3:

```
subroutine solver(neq, a, b, x)
implicit none

integer :: i, j, k, bigindex, neq
double precision :: a(neq,neq), b(neq), x(neq), temp(neq)
double precision :: multiplier, bigvalue, s

do k = 1, neq-1
bigindex = k
bigvalue = abs(a(k,k))
do i = k, neq
if (abs(a(i,k)) > bigvalue) then
bigvalue = abs(a(i,k))
bigindex = i
end if
end do

if (bigindex > k) then
temp(k:neq) = a(k,k:neq)
a(k,k:neq) = a(bigindex, k:neq)
a(bigindex, k:neq) = temp(k:neq)

s = b(k)
b(k) = b(bigindex)
b(bigindex) = s
end if

do i = k+1, neq
multiplier = a(i,k)/a(k,k)
a(i,k:neq) = a(i,k:neq) - multiplier * a(k,k:neq)
```

```
b(i) = b(i) - multiplier * b(k)
end do
end do

x(neq) = b(neq)/a(neq,neq)

do i = neq-1,1,-1
s = 0.0d0
do j = i+1, neq
s = s+ a(i,j)*x(j)
end do

x(i) = (b(i)-s) / a(i,i)
end do

return
end
```

The data from my "interpdata" file is:

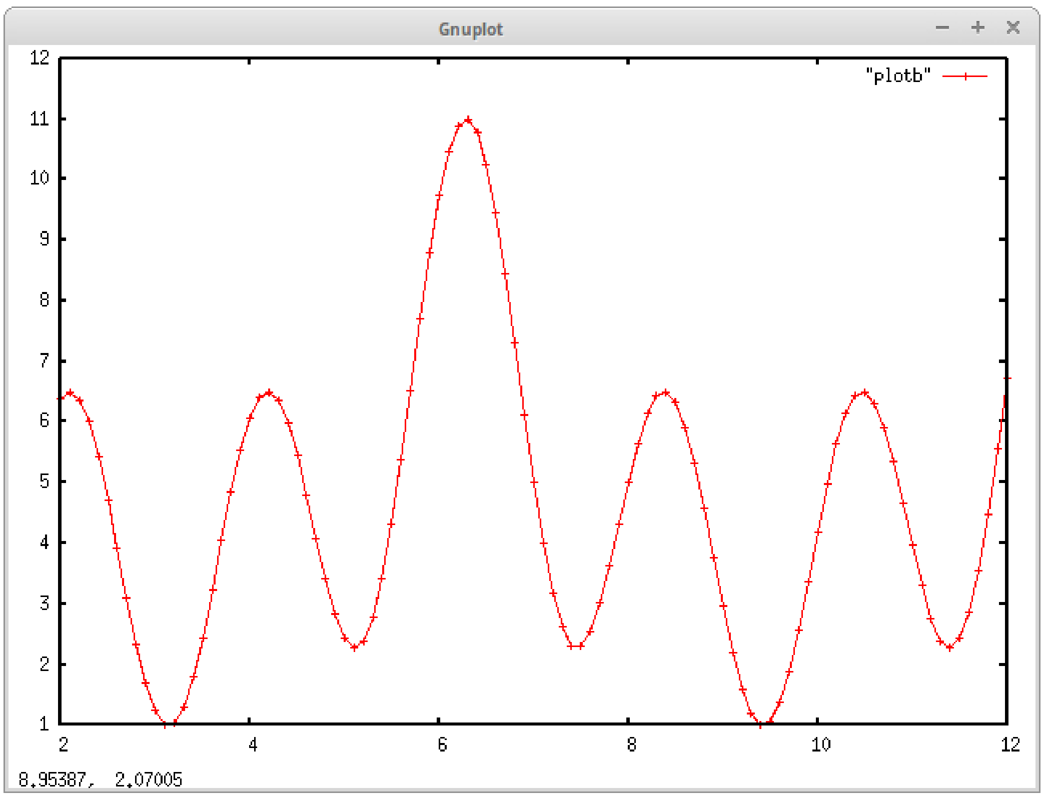
```
4

2 6.3946

3 1.2468

9 2.9616

10 4.1927

12 6.7280
```

This is the resulting output:

```
n = 4
Forming matrix
Matrix
```

```
1.00000  -0.416147  -0.653644   0.960170  -0.145500
1.00000  -0.989992   0.960170  -0.911130   0.843854
1.00000  -0.911130   0.660317  -0.292139  -0.127964
1.00000  -0.839072   0.408082   0.154251  -0.666938
1.00000   0.843854   0.424179  -0.127964  -0.640144

right sides of equations
6.3945999999999996
1.2467999999999999
2.9615999999999998
4.1927000000000003
6.7279999999999998

Calling solver

coefficients in interp polynomial
smallest indexed to biggest
4.9993935427193890
2.0004866623997248
1.0031680500366087
3.0032005765463210
1.1471371726266277E-003

Sanity check printing x, p(x), and p(x)-ydata
2.0000000000000000   6.3945999999999996   0.0000000000000000
3.0000000000000000   1.2468000000000004   4.4408920985006262E-016
9.0000000000000000   2.9616000000000002   4.4408920985006262E-016
10.000000000000000   4.1927000000000003   0.0000000000000000
12.000000000000000   6.7279999999999998   0.0000000000000000

Freeing memory
```

Using gnuplot, entered the syntax: plot “plotb” w lp. This is the resulting graph:


Gnuplot
"plotb"
12
11
10
9
8
7
6
5
4
3
2
1
2
4
6
8
10
12
8.95387, 2.07005