LAB 8

Spring 2011, BESE- 15 A&B

Interpolation

Objective

The aim of this introductory lab is to introduce you to the basic functions in the Matlab and Numerical Methods with Matlab toolbox. By the end of today's lab, you should be able to understand the Interpolation.

Submission Requirements

You are expected to complete the assigned tasks within the lab session and show them to the lab engineer/instructor. Some of these tasks are for practice purposes only while others (marked as '*Exercise*' or '*Question*') have to be answered in the form of a lab report that you need to prepare. Following guidelines will be helpful to you in carrying out the tasks and preparing the lab report.

Guidelines

- In the exercises, you have to put the output in your Lab report. You may add screen print to the report by using the 'Print Screen' command on your keyboard to get a snapshot of the displayed output. This point will become clear to you once you actually carry out the assigned tasks**.**

- Name your reports using the following convention:

  Lab#_Rank_YourFullName

  - '#' replaces the lab number
  - '*Rank*' replaces Maj/Capt/TC/NC/PC
  - '*YourFullName*' replaces your complete name.
- You need to submit the report even if you have demonstrated the exercises to the lab engineer/instructor or shown them the lab report during the lab session.

Newton Polynomials

Newton polynomial is another approach to construct polynomials that have the recursive pattern. To find the coefficients a_k for all the polynomials $P_1(x)$, ... P_N (x) that approximate a given: function f(x). For the polynomial $P_1(x)$ the coefficient a_0 and a_1 have a familiar meaning.

$$P_1(x_1) = f(x_1)$$

$$a_0 = f(x_1) = P_1(x_1)$$

$$a_1 = (f(x_2) - f(x_1))/(x_2-x_1)$$

$X = [x_1\ \ x_2\ \ x_3\ \ x_4\ \ x_5]$

$Y = [f(x_1)\ f(x_2)\ \ f(x_3)\ \ f(x_4)\ \ f(x_5)]$

$$D=\begin{array}{lllll}
a_0=f(x_1) & & & & \\
f(x_2) & a_1 = \dfrac{f(x_2)-f(x_1)}{x_2-x_1} & & & \\
f(x_3) & \dfrac{f(x_3)-f(x_2)}{x_3-x_2} & a_2 = ? & & \\
f(x_4) & \dfrac{f(x_4)-f(x_3)}{x_4-x_3} & & a_3 = ? & \\
f(x_5) & \dfrac{f(x_5)-f(x_4)}{x_5-x_4} & & & a_4 = ?
\end{array}$$

$P_1(x) = a_0+a_1(x-x_1)$

$P_2(x) = a_0+a_1(x-x_1)+a_2(x-x_1)(x-x_2)$

$P_3(x) = a_0+a_1(x-x_1)+a_2(x-x_1)(x-x_2)+a_3(x-x_1)(x-x_2)(x-x_3)$

$P_4(x) = a_0+a_1(x-x_1)+a_2(x-x_1)(x-x_2)+a_3(x-x_1)(x-x_2)(x-x_3)+a_4(x-x_1)(x-x_2)(x-x_3)(x-x_4)$

Example of Newton interpolating polynomial

$f(x)= e^{-x}$ having five interpolating points $x_0=0.0$, $x_1=1.0$, $x_2=2.0$, $x_3=3.0$ and $x_4=4.0$

In order to construct the Newton polynomial in Matlab, first is to construct the divided difference table. This can be done by storing the values in the row of a 5x5 matrix D.

Create matrix D with all zeros elements.
The first column of D i.e. D(:,1) stores the function values at the interpolating points.
The second column of D i.e. D(:,2) stores the first divided differences.

The third column of D i.e. D(:,3) stores the second divided differences.
The fourth column of D i.e. D(:,4) stores the third divided differences.
The fifth column of D i.e. D(:,5) stores the fourth divided differences.

Create a 5x5 matrix D initially with all zeros

```
>>D = zeros(5,5)
```

Setup the vector X with the x-coordinates of the interpolating values

```
>>X=[0  1  2  3  4]
```

Now compute the function f(x) = e^{-x} at the values in X

```
>>Y = exp(-X)
```

Now start computing the divide difference column by column for the matrix D. The first column is just the value of the function at the interpolating points.

```
>> D( :,1) = Y
```

For the second column of D. Strating in second row of D and working down to fifth row.

```
>>D(2,2) = (D(2,1)-D(1,1))/(X(2)-X(1))
>>D(3,2) = (D(3,1)-D(2,1))/(X(3)-X(2))
>>D(4,2) = (D(4,1)-D(3,1))/(X(4)-X(3))
>>D(5,2) = (D(5,1)-D(4,1))/(X(5)-X(4))
```

For the third column of D. Strating in third row of D and working down to fifth row.

```
>>D(3,3) = (D(3,2)-D(2,2))/(X(3)-X(1))
>>D(4,3) = (D(4,2)-D(3,2))/(X(4)-X(2))
>>D(5,3) = (D(5,2)-D(4,2))/(X(5)-X(3))
```

For the fourth and fifth columns of D. Starting in fourth and fifth row of D respectively. Compute these by yourself.

The final value of matrix D should be:

```
D =
        1.0000       0.0000       0.0000       0.0000       0.0000
        0.3679      -0.6321       0.0000       0.0000       0.0000
        0.1353      -0.2325       0.1998       0.0000       0.0000
        0.0498      -0.0855       0.0735      -0.0421       0.0000
        0.0183      -0.0315       0.0270      -0.0155       0.0067
```

To construct the Newton Polynomial of degrees 1 through 4 recursively as follows:

```
>>P1 = [0   D(1,1)]  +D(2,2)*poly(X(1))
```

```
P1 = -0.6321   1.0000


>>P2 = [0   P1]  +D(3,3)*poly(X(1:2))
P2 = 0.1998    -0.8319          1.0000


>>P3 = [0   P2]  +D(4,4)*poly(X(1:3))
P3 = -0.0421   0.3261           -0.9161         1.0000


>>P4 = [0   P3]  +D(5,5)*poly(X(1:4))
P4 = ?
```

Exercise 1

Calculate the third degree Newton and Lagrange polynomial for f(x)=cos(x). X=[0.0 0.4 0.8 1.2] and explain the output of both polynomials.

Implementation of Newton Polynomials: Sudo code as follows

```
procedure (C: Newton coefficients, D: Div Diff table) newpoly(X: given points, Y: fun vals)
begin
n=lenght of given points
D is the matrix of all zeros
Store value of Y in D's first column
for each j:2 to n do
    for each k=j to n do
        D(k,j)=(D(k,j-1)-D(k-1,j-1)/(X(k)-X(k-j+1))
    end
end

C=D(n,n);
for each k=(n-1):-1:1
    C=conv(C,poly(X(k)));
    m=lenght of C
```

```
        C(m)=C(m)+D(k,k);
end
```

Now Call the above written function as

```
>>[C D] = newpoly(X,Y)
```

Now compare your polynomial P4 to the polynomial C returned.

```
>>P4-C
```

If you do not get 0 then you made an mistake!

Now Create an m-file to plot the data points, four polynomials and the function exp(-x) on the same graph in various colors. Put the command below into an m-file called plotnewton.m.

```
XPTS = -1:0.1:6
EXP = exp(-XPTS);
Y1 = polyval(P1, XPTS);
Y2 = polyval(P2, XPTS);
Y3 = polyval(P3, XPTS);
Y4 = polyval(P4, XPTS);
clf
axis([-1  6  -4  4]);
hold on
plot(X,Y,’b*’)
plot(XPTS, EXP, ‘r-‘)
plot(XPTS, Y1, ‘g-‘)
plot(XPTS, Y2, ‘b-‘)
plot(XPTS, Y3, ‘y-‘)
plot(XPTS, Y4, ‘m-‘)
title(‘Your Name  Your Class ---- Newton Polynomials’)
```