M439 Laboratory on Newton Polynomials in MATLAB

1. Download the following two files to your N:\m439 folder: **newpoly.m** **PlotNewton**

Example Problem for Constructing Newton Polynomial: Exercise 8, page 229
Approximating: $f(x) = e^{-x}$
Five Interpolating points: $x_0 = 0.0$, $x_1 = 1.0$, $x_2 = 2.0$, $x_3 = 3.0$, $x_4 = 4.0$.

2. In order to construct the Newton polynomial in MATLAB, we would want to first construct the divided difference table. We can do this by storing the values in the rows of a 5 x 5 matrix D.

The **first column** of D, referenced in MATLAB as **D(: ,1)**, will store the function values at the interpolating points.
The **second** column of D -- D(:, 2) -- will store the **first** divided differences.
The **third** column of D -- D(:, 3) -- will store the **second** divided differences.
The **fourth** column of D -- D(:, 4) -- will store the **third** divided differences.
The **fifth** column of D -- D(:, 5) -- will store the **fourth** divided difference.

3. Create a 5x5 matrix D initially with all zeros:
» D = zeros(5,5)

4. Set up the vector X with the x-coordinates of the interpolating values:
» X = [0 1 2 3 4]
Then compute the function $f(x) = e^{-x}$ at the values in X:
» Y = exp(-X)

5. Now start computing the divided differences column by column for the matrix D
The first column is just the values of the function at the interpolating points, stored in Y:

» D(:,1) = Y

6. We next work on the **second column of D** -- starting in second row (D(2,2)) 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))
etc.

7. The **third column of D** -- starting with D(3,3):
» D(3,3) = (D(3,2)-D(2,2))/(X(3)-X(1))
etc.
Enter the remaining commands to compute entries the rest of column 3, as well as in the last two columns (column 4 and 5).

Your final values for the matrix D should be:

D =

1.0000	0	0	0	0
0.3679	-0.6321	0	0	0
0.1353	-0.2325	0.1998	0	0
0.0498	-0.0855	0.0735	-0.0421	0

0.0183   -0.0315   0.0270   -0.0155   0.0067

8. We can now construct the Newton Polynomials of degrees 1 through 4 recursively as follows: (Note the reason to adjoin the 0 at the beginning of the previous polynomial is to make sure the polynomials have the same number of terms (i.e. vectors representing the polynomials have the same number of terms for adding). Recall that the poly command creates a polynomial with the specified root(s) in its argument.

» 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 =

0.0067   -0.0820   0.3993   -0.9560   1.0000

---

9. The MATLAB program newpoly.m implements a general procedure for computing Newton polynomials of arbitrary degree. It efficiently computes the coefficients of the polynomial by exploiting nested multiplication.

Execute the program with the following command:

» [C D1] = newpoly(X,Y)

and compare your polynomial P4 to the polynomial C returned.

» P4-C

If you don't get 0 (up to computer round off error) you made an mistake!

10. Evaluate the 1st, 2nd, 3rd, and 4th degree Newton polynomials that you calculated at the points 0.5, 1.5 and compare with the actual values (using **exp (-.5)** , **polyval(P1,.5**) etc.).

11. An m-file to plot the data points, four polynomials and the function exp(-x) on the same graph in various colors:
The commands below are in the m-file called **PlotNewton.m**. Execute the m-file with the command **PlotNewton**.

```
XPTS = linspace(-4,10);
EXP = exp(-XPTS);
Y1=polyval(P1,XPTS);
Y2=polyval(P2,XPTS);
Y3=polyval(P3,XPTS);
Y4=polyval(P4,XPTS);
clf
axis([-4 10 -8 8]);
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('Newton Polynomials for exp(-x)')
legend('Points','exp(-x)','Degree 1','Degree 2','Degree 3', 'Degree 4');
```

Print your graph -- on the back summarize what you observe about the polynomial approximations to the function (2-3 sentences).