**Problem:**

In this project we are given the task of constructing a fifth degree interpolating polynomial through the points for and

where each xi is equidistant on the interval [0, π]. We will attempt a solution by using Newton’s Divided Differences to construct the polynomial in question.

Recall that the kth divided difference is given by

Using this formula, we will calculate all the divided differences corresponding to the points provided and store the results in a matrix whose main diagonal will represent the coefficients ai of the interpolating polynomial . We will then construct the polynomial using the coefficients and points provided, and then proceed to plot it against f(x) = sin(x), and determine the maximum error on the interval.

**Source Code:**

% File Name: main\_hw2.m

% Assignment: Project 2

% Student: Joseph Free

% Course: MATH3261

%

% Purpose: This script is where all the work for the project is carried

% out. Here we will construct a fifth degree interpolating polynomial P(x)

% for f(x\_i) = sin(x\_i) over the interval [0, pi], plot sin(x) vs. P(x),

% and find the error of P(x).

%

syms p(x)

x\_5 = [0:pi/5:pi];

f = sin(x\_5);

p(x) = poly\_hw2( interp\_hw2(x\_5, f), x\_5, 5 );

pString = char(p);

% Display p(x) symbolically.

fprintf('Degree five interpolating polynomial:\n');

fprintf('Factored:\n%s', pString);

fprintf('\nExpanded:\n%s\n', char(expand(p)));

% Plot sin(x), p(x) over 250 points in [0,pi]

x\_250 = 0:pi/250:pi;

f=sin(x\_250);

figure

h = plot( x\_250, f, 'b\*', x\_250, p(x\_250), 'r' );

h(2).LineWidth = 2;

axis( [-.01, 3.1\*pi/3, -.01, 1.5] );

title('Fig. 1: Graph of f(x) = sin(x) and p(x)');

legend('f(x) = sin(x)', 'p(x)');

err = double( max( abs( f - p(x\_250) ) ) );

fprintf('Maximum error on [0,pi]: %f\n', err);

**Source Code: (cont.)**

% File Name: poly\_hw2.m

% Assignment: Project 2

% Student: Joseph Free

% Course: MATH3261

%

% Purpose: This function takes an nxn upper triangular matrix F which

% contains the divided differences corresponding to a data set. From

% F, a symbolic representation of the interpolating polynomial is

% constructed and returned.

%

% Required input:

%

% F -- An nxn upper triangular matrix containing

% the divided differences of a data set.

% x\_n -- The x-coordinates from the data set in question.

% N -- Number of digits to round right of the decimal.

function P = poly\_hw2( F, x\_n, N )

syms x;

P = round( F(1,1), N );

for( i = 2:length(F) )

P = P + round(F(i,i), N)\*prod( ( x-round(x\_n(1:i-1), N) ) );

end

%Convert coefficients to decimal.

P = vpa(P);

end

**Source Code: (cont.)**

% File Name: interp\_hw2.m

% Assignment: Project 2

% Student: Joseph Free

% Course: MATH3261

%

% Purpose: This function calculates the coefficients of an n

% degree interpolating polynomial and stores them as the diagonal of

% an n by n matrix, which also happens to contain all ith order divided

% differences for i = 0, 1,..., n.

%

% Required input:

%

% x\_n -- Array of x-coordinates.

% f\_n -- Array of observed values f(x\_n).

%

function F = interp\_hw2( x\_n, f\_n)

% Initialize variables n and F.

n = numel(x\_n);

F = zeros( n );

% Replace first row of F with the observed values f\_n.

F(1,:) = f\_n;

% Calculate divided differences. By this end of this process

% F will be an nxn upper triangular matrix.

for (i = 2:n)

for (j = i:n)

F(i,j) = (F(i-1,j) - F(i-1,j-1))/(x\_n(j) - x\_n(j-i+1));

end

end

end

**Output / Solution:**

*Interp\_hw2:*

x\_5 =

0 0.6283 1.2566 1.8850 2.5133 3.1416

>> f = sin(x\_5)

f =

0 0.5878 0.9511 0.9511 0.5878 0.0000

>> interp\_hw2(x\_5, f)

ans =

0 0.5878 0.9511 0.9511 0.5878 0.0000

0 0.9355 0.5782 0.0000 -0.5782 -0.9355

0 0 -0.2844 -0.4601 -0.4601 -0.2844

0 0 0 -0.0932 -0.0000 0.0932

0 0 0 0 0.0371 0.0371

0 0 0 0 0 0.0000

*poly\_hw2:* (Using upper triangular matrix above, x\_5, and N = 5)

>> poly\_hw2(interp\_hw2(x\_5, f), x\_5, 5)

ans =

0.93549\*x - 0.28435\*x\*(x - 0.62832) - 0.09323\*x\*(x - 0.62832)\*(x - 1.25664) + 0.0371\*x\*(x - 1.88496)\*(x - 0.62832)\*(x - 1.25664)

**Output / Solution: (cont.)**

*Main\_hw2:*

>> main\_hw2

Degree five interpolating polynomial:

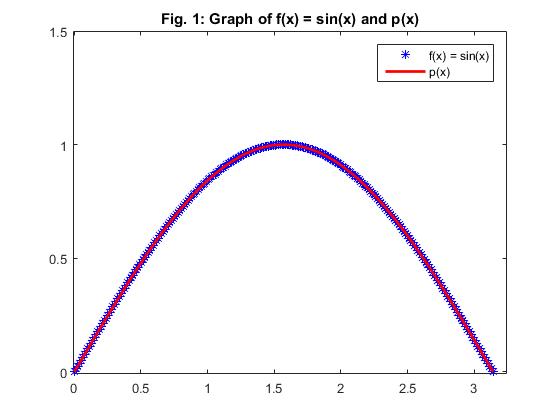
Factored:

0.93549\*x - 0.28435\*x\*(x - 0.62832) - 0.09323\*x\*(x - 0.62832)\*(x - 1.25664) + 0.0371\*x\*(x - 1.88496)\*(x - 0.62832)\*(x - 1.25664)

Expanded:

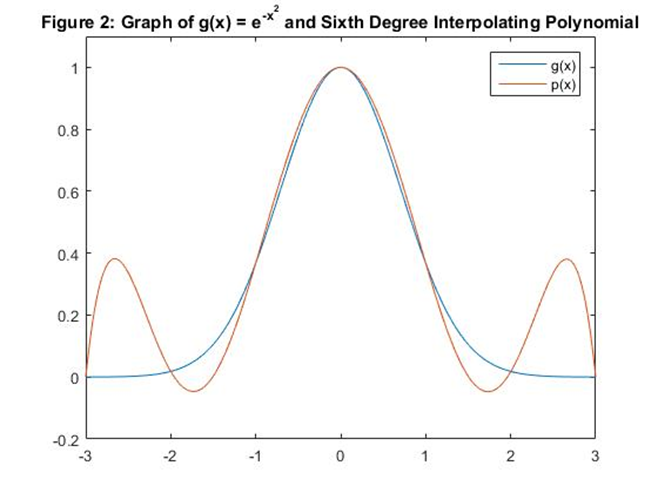
0.9853246253931896832\*x + 0.05249699654144\*x^2 - 0.233094032\*x^3 + 0.0371\*x^4

Maximum error on [0,pi]: 0.001313



**Figure 1:** Graph of f(x) = sin(x) and the fifth degree interpolating polynomial passing through the points for on the interval [0, π].

**Observations:**

 There are a couple of points that need to be made regarding the interpolating polynomial yielded in the computations above. The first and foremost is that it is not, in fact, a fifth degree polynomial but a fourth degree. This isn’t due to any inherent error in the source code or the logic involved. As it turns out the coefficient of the x5 term is so small as to render the whole term nearly non-existent, a problem that is only exasperated when rounding is involved (as in the solution provided). If we so wished, we could adjust the parameter N so as to have a fifth degree term, but I will not endeavor to do so.

Secondly, if we examine figure 1 we see that p(x) and f(x) seem to be almost exactly identical – a result that is rightfully suspect. But if we consider that maximum error ( calculated above, we see that it isn’t an altogether unreasonable proposition. Thus, together with the graph and the error we can conclude that p(x) is a remarkably good approximation of f(x) = sin(x) on the interval [0,π].

Outside the interval it is interesting to note how quickly p(x) begins to deviate from f(x). We summarize the behavior in the following table using common angles:

|  |  |  |  |
| --- | --- | --- | --- |
| x | f(x) = sin(x) | p(x) | Error |
|  | -0.5000 | -0.4650 | 0.035 |
|  | -0.7071 | -0.6141 | 0.093 |
|  | -0.8660 | -0.6615 | 0.2045 |
|  | -1 | -0.2881 | 0.7119 |

As a final talking point, it should be noted that the source code provided is sufficiently capable of constructing interpolating polynomials for any arbitrary data set with minimal modifications. For instance, consider the points for . By supplying the appropriate arguments to interp\_hw2 and poly\_hw2, we get the following results:

>> expand( poly\_hw2(interp\_hw2(x,f), x, 5) )

ans =

- 0.01275\*x^6 + 0.00001\*x^5 + 0.19263\*x^4 - 0.00011\*x^3 - 0.81207\*x^2 + 0.00002\*x + 0.99999

See figure 2 for the graph of this polynomial against