CS 409 Introduction to Scientific Computing

PROJ 2

Interpolating Polynomial Analysis and Matlab Code

Student Name/Surname: Aleyna Kütük

ID: 23991

Date: 18/08/2019

A. Interpolating Polynomials and Discussion on the Solutions :

In this part, the program will be analysed with different polynomials and solutions are going to be displayed detailly. At the below, there is discussion and analysis part on the interpolating polynomials.

1)

At the below, the function is iterated to be analysed by using Newton's divided difference table and obtained a polynomial which has degree 14. The degree of this polynomial is determined according to the absolute error criteria and the program is stopped by checking the testing points.

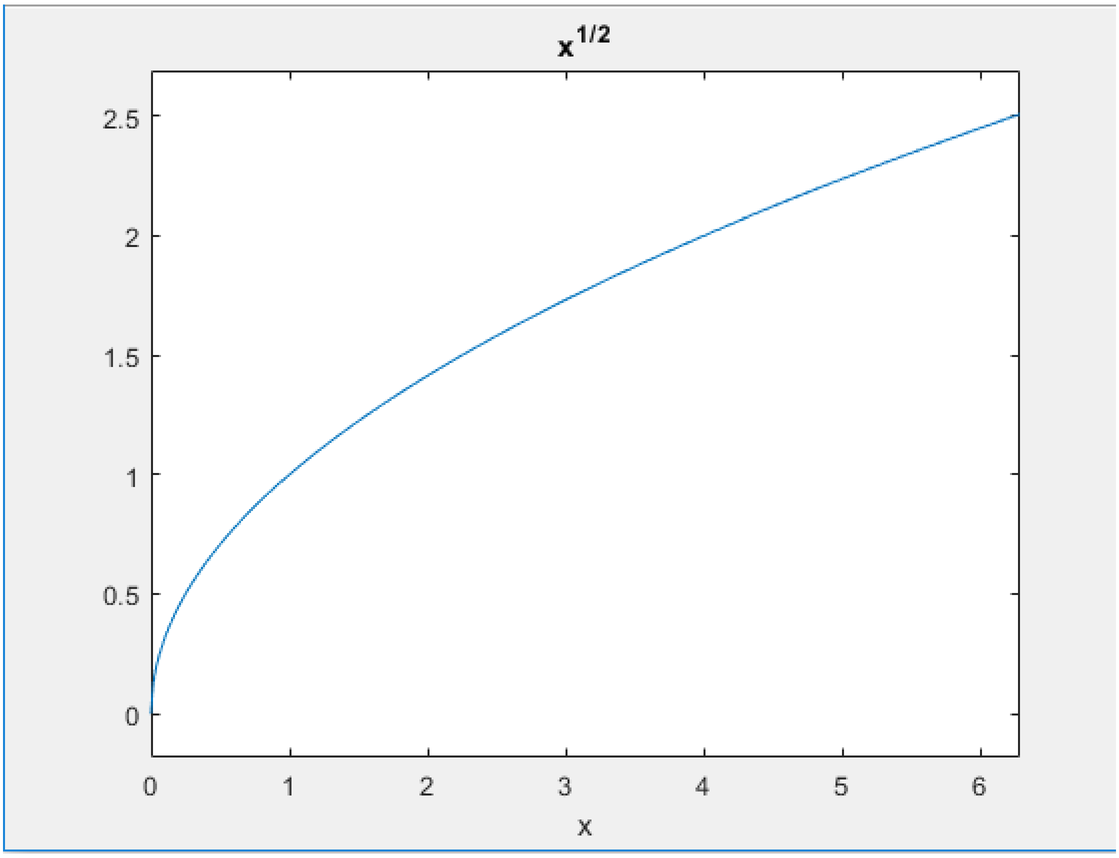
```
>> Interpolation
Enter the function in terms of x:sqrt(x)
What is the lower bound?4
What is the upper bound?20
What is the absolute error power(n=?) 1.0E-n?4

    a0 a1 a2 ... an

   1.4804e-10

   1.5036e-10

  -7.0414e-10

   2.9311e-09

  -1.2086e-08

   5.0306e-08

  -2.0768e-07

   7.6712e-07

  -4.3240e-07

   2.6780e-06
```

```
   Xi values:   [xlower,xupper]
 Columns 1 through 14

   4.0000   12.0000   20.0000    8.9810    4.9523    4.1811    4.0522         0    3.2633    3.7751    3.8624    3.8627    3.9217    3.9435

 Column 15

   3.9671

     F[Xi]
 Columns 1 through 14

   2.0000         0         0         0         0         0         0         0         0         0         0         0         0         0
   3.4641    0.1830         0         0         0         0         0         0         0         0         0         0         0         0
   4.4721    0.1260   -0.0036         0         0         0         0         0         0         0         0         0         0         0
   2.9968    0.1339   -0.0026    0.0002         0         0         0         0         0         0         0         0         0         0
   2.2254    0.1915   -0.0038    0.0002   -0.0000         0         0         0         0         0         0         0         0         0
   2.0448    0.2342   -0.0089    0.0003   -0.0000    0.0000         0         0         0         0         0         0         0         0
   2.0130    0.2464   -0.0136    0.0010   -0.0000    0.0000   -0.0000         0         0         0         0         0         0         0
        0    0.4968   -0.0599    0.0093   -0.0009    0.0000   -0.0000    0.0000         0         0         0         0         0         0
   1.8065    0.5536   -0.0720    0.0132   -0.0023    0.0002   -0.0000    0.0000   -0.0000         0         0         0         0         0
   1.9430    0.2667   -0.0760    0.0144   -0.0029    0.0005   -0.0001    0.0000   -0.0000    0.0000         0         0         0         0
   1.9653    0.2559   -0.0181    0.0150   -0.0031    0.0007   -0.0001    0.0000   -0.0000    0.0000   -0.0000         0         0         0
   1.9654    0.2544   -0.0167    0.0024   -0.0033    0.0007   -0.0002    0.0000   -0.0000    0.0000   -0.0000    0.0000         0         0
   1.9803    0.2534   -0.0163    0.0021   -0.0004    0.0007   -0.0002    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000         0
   1.9858    0.2521   -0.0162    0.0021   -0.0003    0.0001   -0.0002    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000
fx 1.9917    0.2514   -0.0160    0.0020   -0.0003    0.0001   -0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000
```

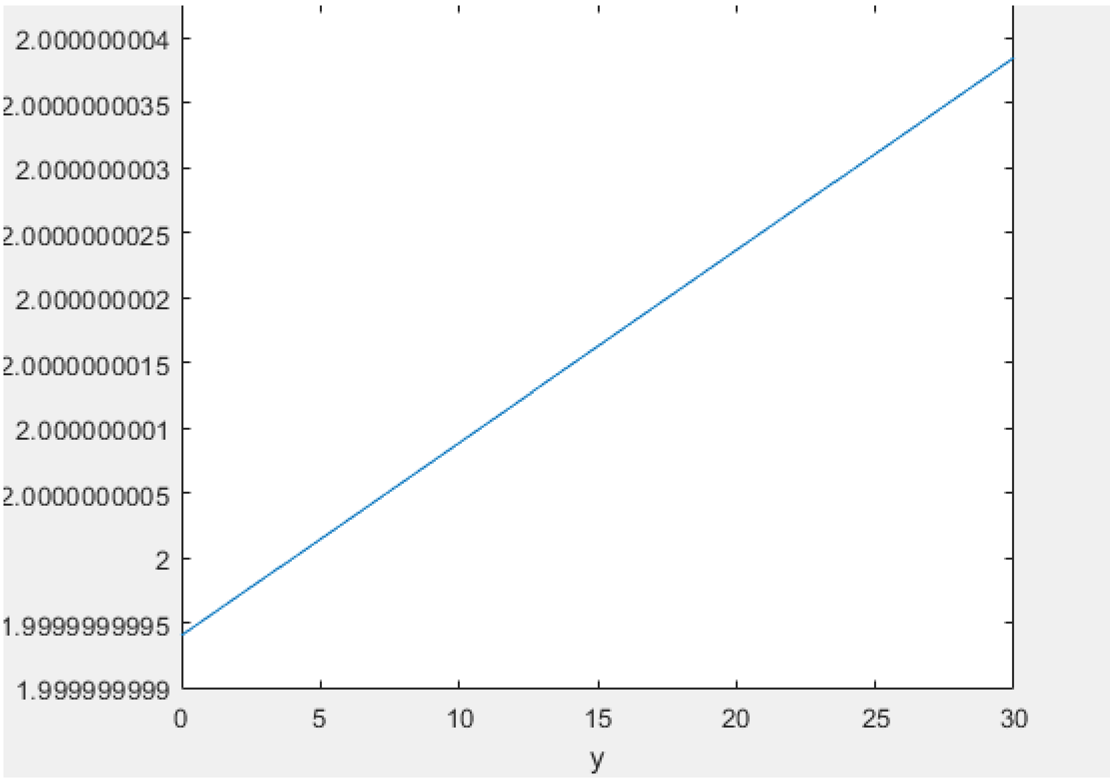
```
   1.9654    0.2544   -0.0167    0.0024   -0.0033    0.0007   -0.0002    0.0000   -0.0000    0.0000   -0.0000    0.0000         0         0
   1.9803    0.2534   -0.0163    0.0021   -0.0004    0.0007   -0.0002    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000         0
   1.9858    0.2521   -0.0162    0.0021   -0.0003    0.0001   -0.0002    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000
   1.9917    0.2514   -0.0160    0.0020   -0.0003    0.0001   -0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000

 Column 15

        0
        0
        0
        0
        0
        0
        0
        0
        0
        0
        0
        0
        0
        0
   0.0000

Final degree:
    14

fx >>
```

After some point, the divided difference table close enough to 0 and the absolute error criteria satisfied for the point: f(xi)= 1.9917 , xi =3,9668

And below, the function is plotted by using ezplot() func in Matlab so that we can see that how we are close enough with our polynomial which we obtained its coefficients with the divided difference table.



$x^{1/2}$
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
x



2.000000004
2.0000000035
2.000000003
2.0000000025
2.000000002
2.0000000015
2.000000001
2.0000000005
2
1.9999999995
1.999999999
0
5
10
15
20
25
30
y

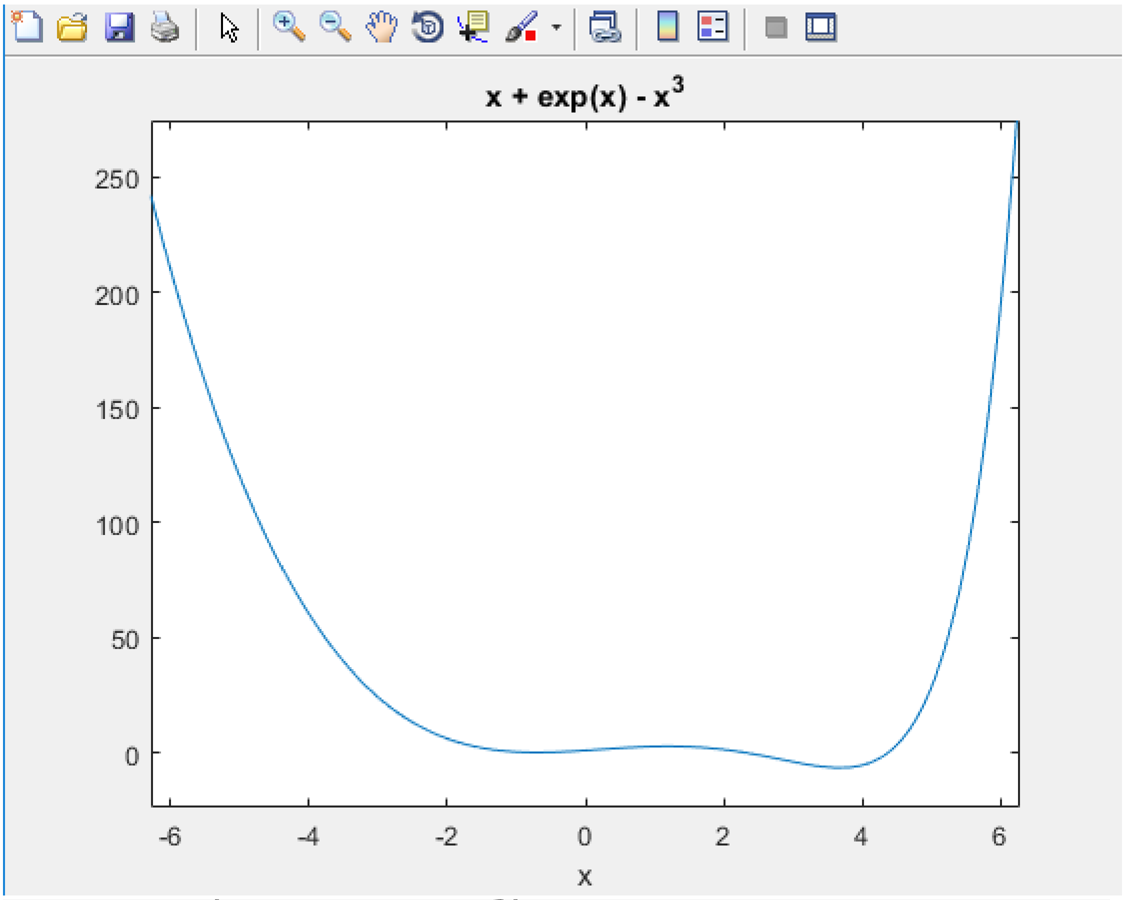
2)

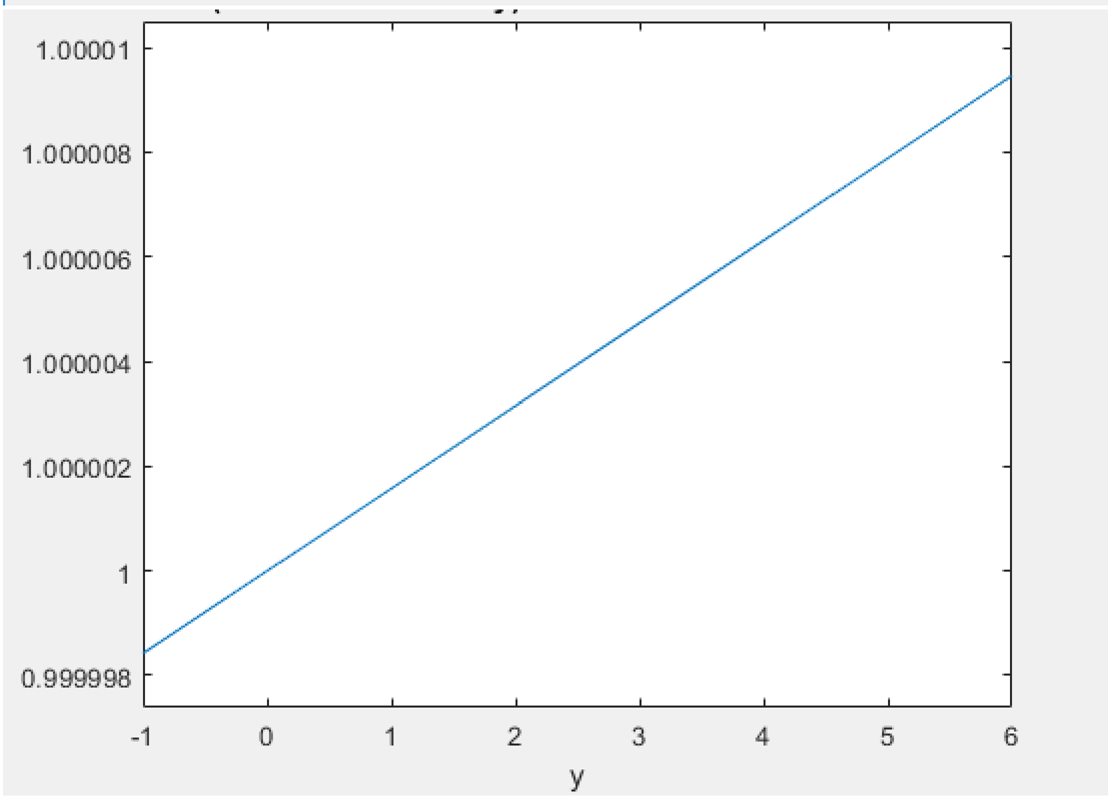
At the below, the function is analysed by using Newton's divided difference table again and obtained a polynomial which has degree 10.

The degree is determined according to the absolute error criteria and the program is stopped by checking the 1000 testing points. Each time I divided the entire range to 1000 testing points so that I can try for the absolute error criteria.

```
>> Interpolation
Enter the function in terms of x:exp(x)-(x*x*x)+x
What is the lower bound?0
What is the upper bound?5
What is the absolute error power(n=?) 1.0E-n?4

    a0 a1 a2 ... an

   1.5780e-06

   1.8773e-05

   2.0798e-04

    0.0021

    0.0203

    0.1763

    1.2984

    4.5038

    2.5039
```

```
    2.5039

   -0.7770

     1

  Xi values:   [xlower,xupper]
       0    2.5000    5.0000    4.9950    3.4250    0.9188    0.2327    0.0918    0.0917    0.0403    0.0275

    F[Xi]
    1.0000         0         0         0         0         0         0         0         0         0         0
   -0.9425   -0.7770         0         0         0         0         0         0         0         0         0
   28.4132   11.7423    2.5039         0         0         0         0         0         0         0         0
   28.0422   74.1174   25.0001    4.5038         0         0         0         0         0         0         0
   -6.0297   21.7019   33.2796    8.9509    1.2984         0         0         0         0         0         0
    2.6495   -3.4631    6.1737    6.6417    1.4604    0.1763         0         0         0         0         0
    1.4821    1.7014   -1.6178    1.6361    1.0500    0.1810    0.0203         0         0         0         0
    1.1871    2.0932   -0.4737   -0.3432    0.4037    0.1317    0.0205    0.0021         0         0         0
    1.1869    2.0708    0.1586   -0.7644    0.1264    0.0566    0.0153    0.0022    0.0002         0         0
    1.0813    2.0546    0.3150   -0.8131    0.0554    0.0210    0.0072    0.0016    0.0002    0.0000         0
    1.0554    2.0310    0.3679   -0.8225    0.0459    0.0106    0.0031    0.0008    0.0002    0.0000    0.0000

Final degree:
    10
```





And the function graph is provided at the above.

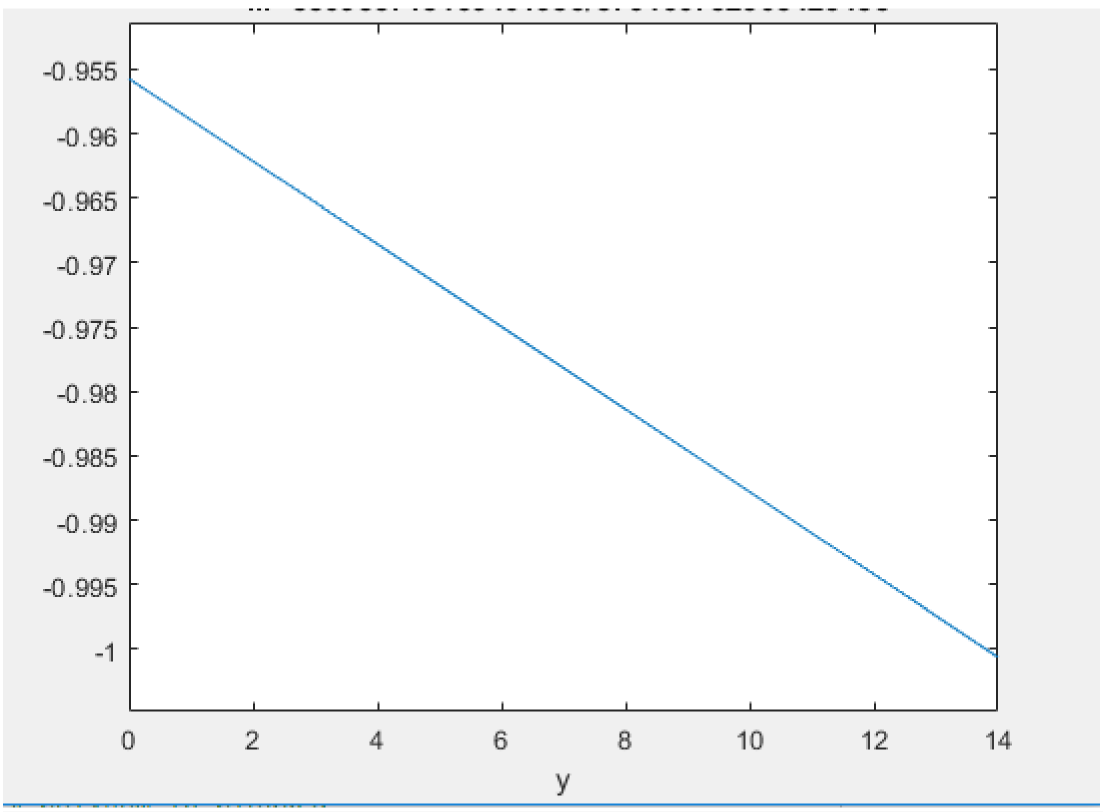
3)

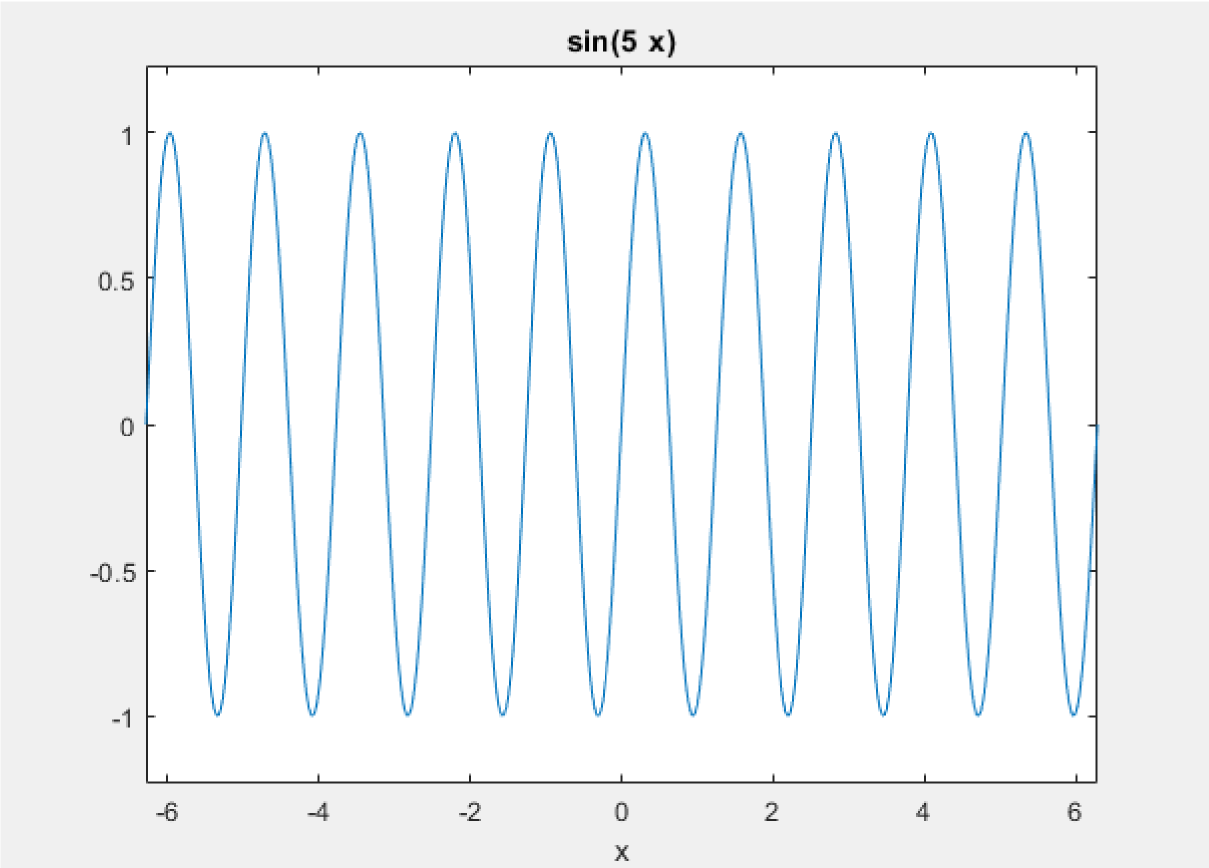
The degree of polynomial changes according to how fast we approached to the absolute error criteria for given function with the given interval. Each time we iterated over all testing points to find the point which makes the absolute error criteria satisfies. Then, I can take this point as a new interpolating point to be add to the divided difference table.

So, as it can be seen from the results below the sin function satisfied the error criteria at the 13th degree of the polynomial.

```
>> Interpolation
Enter the function in terms of x:sin(5*x)
What is the lower bound?1
What is the upper bound?13
What is the absolute error power(n=?) 1.0E-n?4

    a0 a1 a2 ... an

   -0.0032

    0.0082

    0.0136

   -0.0165

   -0.0340

    0.0032

    0.0302
```

```
  Xi values:   [xlower,xupper]
  1.0000    7.0000   13.0000    5.8168    2.0656    1.2272    1.0712         0    0.6296    0.7553    0.7556    0.8674    0.8983    0.9397

    F[Xi]
 -0.9589         0         0         0         0         0         0         0         0         0         0         0         0         0
 -0.4282    0.0885         0         0         0         0         0         0         0         0         0         0         0         0
  0.8268    0.2092    0.0101         0         0         0         0         0         0         0         0         0         0         0
 -0.7241    0.2159   -0.0057   -0.0033         0         0         0         0         0         0         0         0         0         0
 -0.7853    0.0163    0.0183   -0.0049   -0.0015         0         0         0         0         0         0         0         0         0
 -0.1467   -0.7617    0.1695   -0.0128    0.0014    0.0126         0         0         0         0         0         0         0         0
 -0.8000    4.1875   -4.9771    1.0845   -0.0920    0.0157    0.0438         0         0         0         0         0         0         0
       0   -0.7468    4.0208   -4.3561    0.9353   -0.0790    0.0135    0.0302         0         0         0         0         0         0
 -0.0066   -0.0104   -1.6678    9.5196   -9.6631    2.0432   -0.1716    0.0291    0.0032         0         0         0         0         0
 -0.5931   -4.6673   -6.1655   14.2391  -10.0013    0.2581    0.3527   -0.0428    0.0115   -0.0340         0         0         0         0
 -0.5941   -4.0237    5.1112   14.9251   -2.1734  -16.5971   12.8664   -2.4724    0.1984   -0.0299   -0.0165         0         0         0
 -0.9303   -3.0065    9.0764   16.6779    2.0207  -20.5794   11.0674    1.5014   -0.8029    0.0825   -0.0183    0.0136         0         0
 -0.9757   -1.4674   10.7815   11.9240  -17.6935  -21.9459    7.9049    9.6157   -6.9515    1.2501   -0.0965    0.0128    0.0082         0
 -0.9999   -0.5839   12.2089    7.7496  -22.6338  -15.9307    6.4010   11.4418   -6.3524   -0.5321    0.3654   -0.0383    0.0084   -0.0032

Final degree:
  13
```



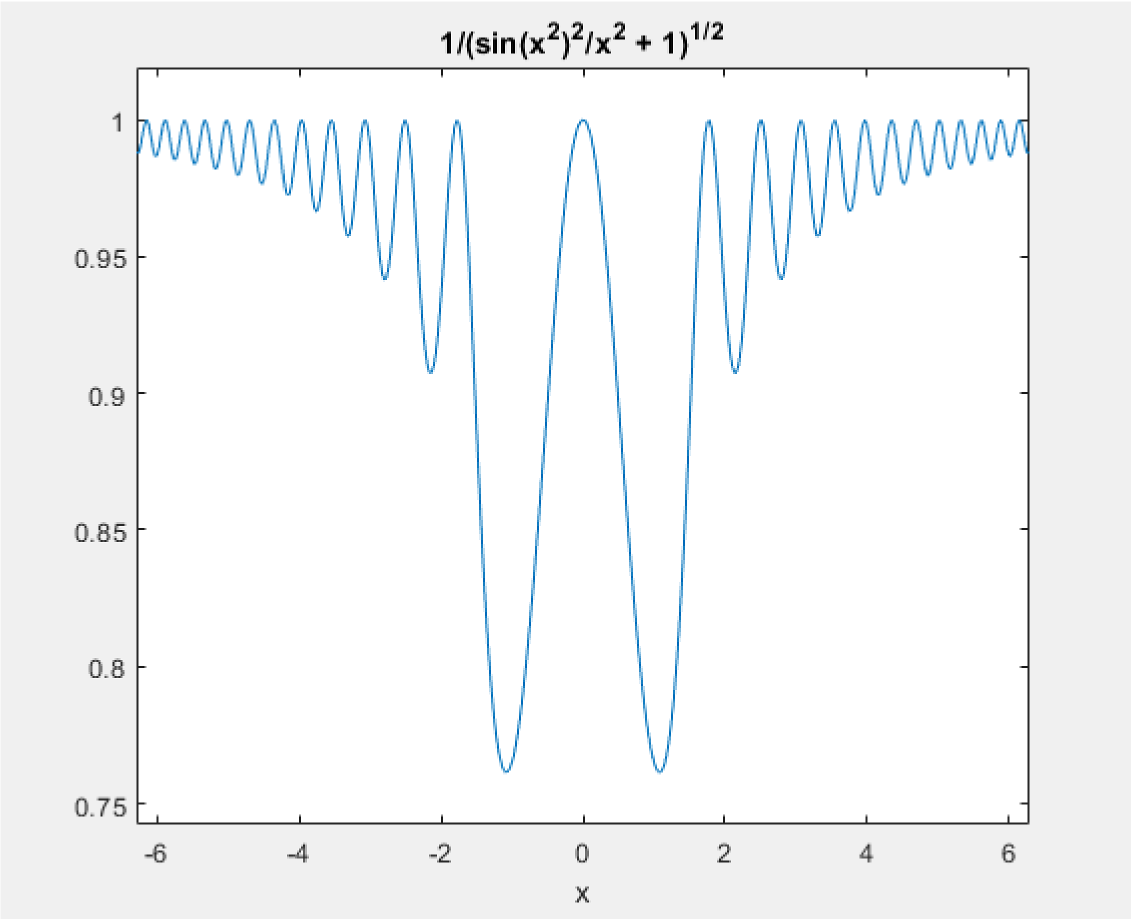


The function again provided for the purpose of visualization on how interpolating polynomial and function itself close to each other.

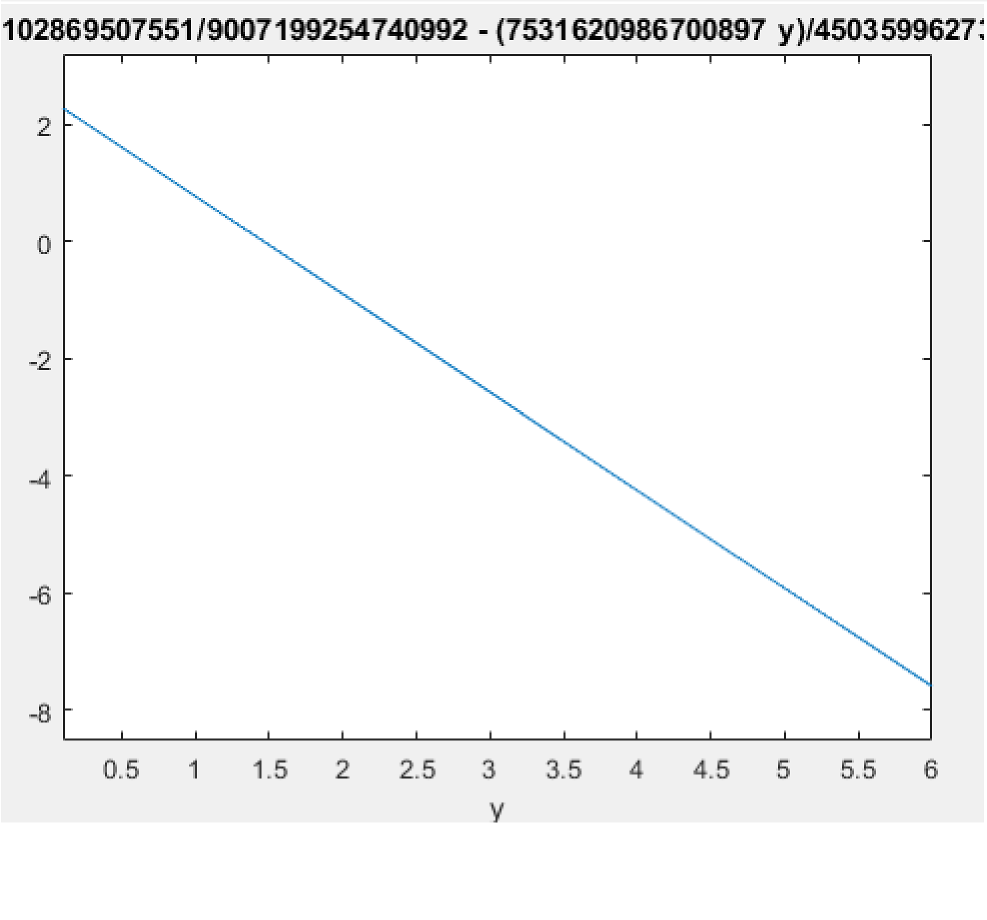
4)

This time the function is analysed and the degree of interpolating polynomial is obtained as 11 so that we can claim that in general the degree does not differ too much, and we can see the absolute error criteria can be obtained around 10 degree of polynomial easily.

```
>> Interpolation
Enter the function in terms of x:1/sqrt(1+((sin(x*x)*sin(x*x))/(x.^2)))
What is the lower bound?1
What is the upper bound?4
What is the absolute error power(n=?) 1.0E-n?4

    a0 a1 a2 ... an

   -1.6724

   -1.1873

   -0.5235

    0.0890

    0.4840

    0.5676

    0.2092

    0.7652

    Xi values:   [xlower,xupper]
    1.0000    2.5000    4.0000    3.9970    1.9990    1.8680    1.6673    1.1777    1.1656    1.0709    1.0484    1.0251

      F[Xi]
    0.7652         0         0         0         0         0         0         0         0         0         0         0
    0.9999    0.1565         0         0         0         0         0         0         0         0         0         0
    0.9974   -0.0017   -0.0527         0         0         0         0         0         0         0         0         0
    0.9978   -0.1307   -0.0862   -0.0112         0         0         0         0         0         0         0         0
    0.9356    0.0311   -0.0809   -0.0106    0.0005         0         0         0         0         0         0         0
    0.9838   -0.3674    0.1872   -0.1257    0.1821    0.2092         0         0         0         0         0         0
    0.9782    0.0277   -1.1910    0.5916   -0.3075    0.5880    0.5676         0         0         0         0         0
    0.7677    0.4300   -0.5829   -0.7404    0.4724   -0.2763    0.6536    0.4840         0         0         0         0
    0.7662    0.1210    0.6159   -1.7066    1.1593   -0.2426   -0.0119    0.4988    0.0890         0         0         0
    0.7615    0.0494    0.6709   -0.0922   -2.0254    3.4315   -1.2557    0.4246    0.0519   -0.5235         0         0
    0.7621   -0.0238    0.6250    0.3546   -0.7220   -1.5903    5.2829   -2.2175    0.8952   -0.5810   -1.1873         0
    0.7632   -0.0501    0.5750    0.3565   -0.0121   -1.1055   -0.5751    6.0151   -2.7702    1.2321   -1.2293   -1.6724

Final degree:
    11
```



1/(sin(x^2)^2/x^2 + 1)^1/2
1
0.95
0.9
0.85
0.8
0.75
-6
-4
-2
0
2
4
6
x



102869507551/9007199254740992 - (7531620986700897 y)/4503599627
2
0
-2
-4
-6
-8
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
y

Conclusion on Results:

Both interval of the function which we are looking for and the function itself changes the closeness of this interpolation method to the real function.

As conclusion, we should consider the error criteria to be able to select new interpolation point each time.

The most important point of this Newton's divided difference method is that it helps you to modify the table each time without calculating the previous steps over and over. So you can just modify on the previous result to obtain better solution, no need for the recalculations like Lagrange Method force you to do.

B. Appendix:

Matlab code of the program is the following:

```
syms x;
func(x) = input('Enter the function in terms of x:');
    %function is taken by the user

f = inline(func);

%started with 3 points so it means that first, second order interpolation
will  be analysed
m = 3;
xlower = input('What is the lower bound?');
xupper = input('What is the upper bound?');
n = input('What is the absolute error power(n=?) 1.0E-n?');  %absolute
error exponent is taken from user
range = (xupper - xlower)/(m-1);
xmid = (xupper + xlower) / 2;
a0 = f(xlower);


X = zeros(1,m); % Memory preallocation

for k=1:m
  X(k) = xlower + (range * (k-1));
end
```

```
while(true)

    T = zeros(1,1000); % Memory preallocation
    r = (xupper - xlower)/999;
    for k=1:998
        T(k) = xlower + (r * k);  %stored the each testing points
    end

    m = length(X);
    Y = zeros(1,m);
    for k=1:m
        Y(k) = f(X(k));  %function values of the X(k) array
    end
    D = zeros(m,m);

    D(:,1) = Y';
    for j=2:m
        for k=j:m
            D(k,j) = (D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));  % stores
divided difference values
        end
    end


    %y = polyval(p,x);

    p = D(m,m);
    for t=1:1000
        for k = 1:m-1
            p = D(m-k,m-k) + ( T(t) - X(m-k))*p;
        end
        if(t == 1)
            max = abs( f(T(t)) - p );  %first value is determined as max
point
            point = T(t);
        else
            if( max < abs( f(T(t)) - p ))   %max point is updated
                max = abs( f(T(t)) - p );
                point = T(t);
            end
        end
    end


    if max < 1.0*10.^-n  %absolute error criteria to break loop
        break;
    else
        X(m+1) = point;  %new interpolation point is determined and added
        xupper = point;
    end
end

disp(' ');
disp('    a0 a1 a2 ... an   ');
disp(' ');
pol = zeros(1,m);
for k = m:-1:1
    disp(D(k,k));
    pol(1,k) = D(k,k);
end
```

```
syms y;
polynom(y) = pol(1,1) + (y*0);
for k = 1:m-1
    polynom(y) = D(m-k,m-k) + (y - X(m-k))*pol(1,k+1);
end

disp('    Xi values:    [xlower,xupper]      ');
disp(X);
disp('      F[Xi]');
disp(D);
disp('Final degree: ');
disp(m-1);  % degree of polynomial which satisfies the absolute error
criteria

%ezplot(func)   % func is plotted
%ezplot(polynom,[min,max]) % polynom is plotted
```