Assignment Week 0

Question 2 And Question 3

```
//Code For Legrange Interpolation i.e Q2 and evaluation on 100 uniformly spaced values i.e Q3

#include<stdio.h>

#include<conio.h>

#include<math.h>


/*Function to evaluate Pi(x)*/


float Pi(int i, int n, float xx[n], float x){

   int j;

   double product=1;

   for(j=0;j<=n;j++){

      if(j!=i)

         product=product*(x-xx[j])/(xx[i]-xx[j]);

   }

      return product;

}


/*Function to evaluate lintp(x) where lintp is the Lagrange interpolating polynomial of degree n*/


float lintp(int n, float xx[n], float yy[n], float x){

   double sum=0;

   int i;

   for(i=0;i<=n;i++){

      sum=sum+Pi(i,n,xx,x)*yy[i];

   }

   return sum;

}

main(){
    int i,n;  //n is the degree
    int pi=3.14159;
    printf("Enter the number of data-points:\n");
    scanf("%d",&n);  //no. of data-points is n
    //Arrays to store the (n+1) x and y data-points of size n+1
    float xx[n];
    float yy[n];
    //The sine fucntion is sampled at xx data points(I have taken it in the radian)
    //yy are the corresponding values of function sampled at xx
    printf("Enter the xx data-points:\n");
    for(i=0;i<=n;i++){
        xx[i]=i*2*M_PI/n;
        yy[i]=sin(xx[i]);
        printf("\n %f",yy[i]);
    }

    float x;  //value of x for which interpolated value is required
    printf("\nValues of x and its interpolated value y(x):\n");
    //Evaluating the function on 100 uniformly spaced values between 0 and 2*pi
    for(i=0;i<100;i++){
        x=i*M_PI*2/99;
    printf("\n %f",x);
    printf(" %f",lintp(n,xx,yy,x));
    printf("\n");
    }
}
```

Output.txt file of Question 3 with each line containg as x y

```
0.000000 -0.000000

0.063467 0.064054

0.126933 0.127591

0.190400 0.190423

0.253866 0.252350

0.317333 0.313169

0.380799 0.372668

0.444266 0.430634

0.507732 0.486856

0.571199 0.541121

0.634665 0.593221

0.698132 0.642952

0.761598 0.690119

0.825065 0.734533
```

0.888531	0.776014
0.951998	0.814394
1.015464	0.849516
1.078931	0.881235
1.142397	0.909420
1.205864	0.933955
1.269330	0.954738
1.332797	0.971680
1.396263	0.984713
1.459730	0.993780
1.523196	0.998844
1.586663	0.999882
1.650129	0.996889
1.713596	0.989876
1.777063	0.978872

1.840529	0.963921
1.903996	0.945082
1.967462	0.922434
2.030929	0.896066
2.094395	0.866087
2.157862	0.832619
2.221328	0.795796
2.284795	0.755767
2.348261	0.712696
2.411728	0.666756
2.475194	0.618133
2.538661	0.567023
2.602127	0.513633
2.665594	0.458177
2.729060	0.400879

```
2.792527 0.341970

2.855993 0.281687

2.919460 0.220272

2.982926 0.157972

3.046393 0.095038

3.109859 0.031722

3.173326 -0.031722

3.236792 -0.095038

3.300259 -0.157972

3.363725 -0.220272

3.427192 -0.281687

3.490659 -0.341970

3.554125 -0.400879

3.617592 -0.458177

3.681058 -0.513633
```

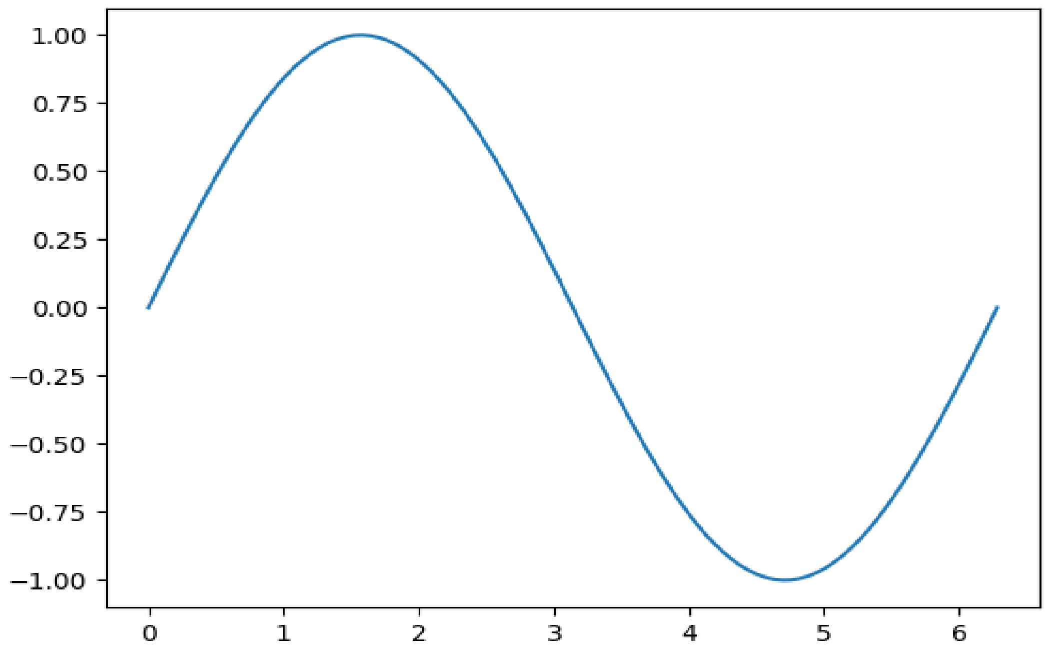
3.744524	-0.567023
3.807991	-0.618133
3.871458	-0.666756
3.934924	-0.712696
3.998391	-0.755767
4.061857	-0.795796
4.125324	-0.832619
4.188790	-0.866087
4.252257	-0.896066
4.315723	-0.922434
4.379190	-0.945082
4.442656	-0.963921
4.506123	-0.978872
4.569589	-0.989876
4.633056	-0.996889

4.696522	-0.999882
4.759989	-0.998844
4.823455	-0.993780
4.886922	-0.984713
4.950388	-0.971680
5.013855	-0.954738
5.077322	-0.933955
5.140788	-0.909420
5.204255	-0.881235
5.267721	-0.849516
5.331188	-0.814394
5.394654	-0.776014
5.458120	-0.734533
5.521587	-0.690119
5.585053	-0.642952

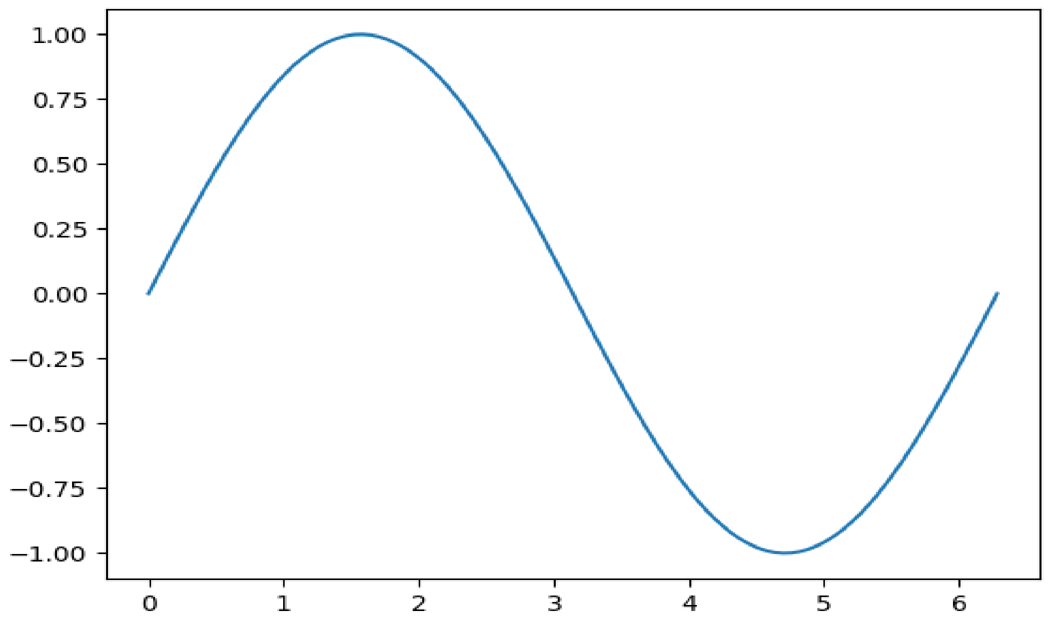
```
5.648520 -0.593221

5.711987 -0.541121

5.775453 -0.486856

5.838920 -0.430634

5.902386 -0.372668

5.965853 -0.313169

6.029319 -0.252350

6.092786 -0.190423

6.156252 -0.127591

6.219719 -0.064054

6.283185 0.000000
```

Question 4 Code And Graph

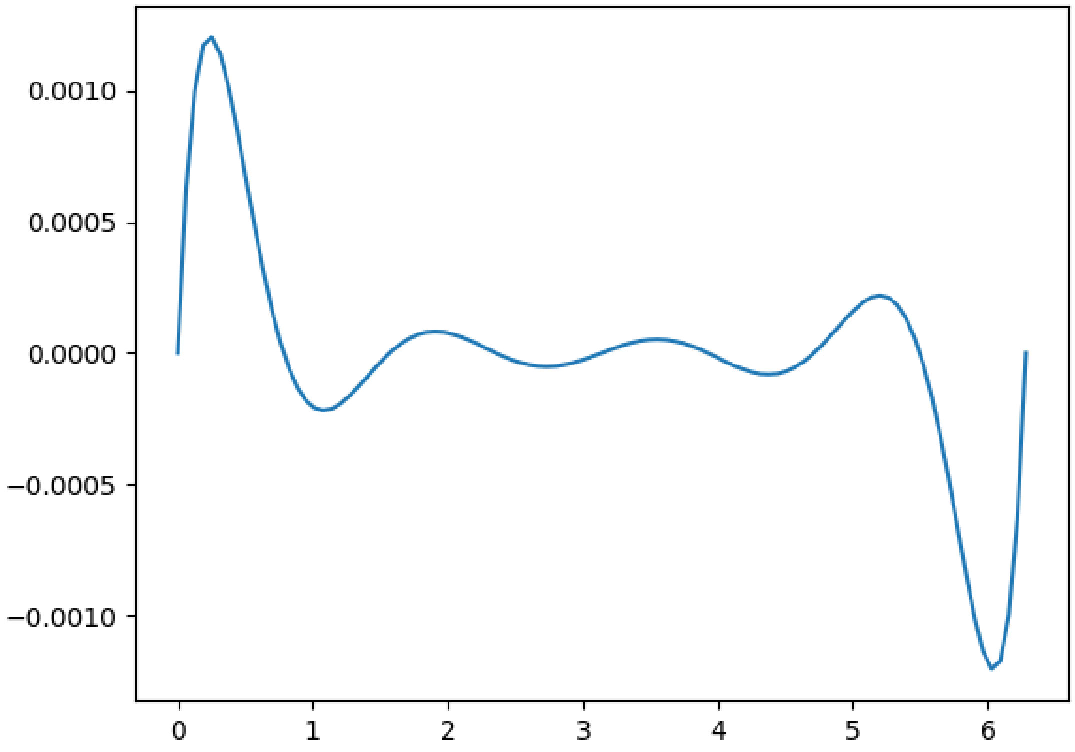
```
Microsoft Windows [Version 10.0.18363.1556]

(c) 2019 Microsoft Corporation. All rights reserved.


C:\Users\Abdul Wahid>ipython --pylab

Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]

Type 'copyright', 'credits' or 'license' for more information

IPython 7.26.0 -- An enhanced Interactive Python. Type '?' for help.

Using matplotlib backend: TkAgg


In [1]: import matplotlib.pyplot as plt


In [2]: import numpy as np


In [3]: x,y=np.loadtxt("./output.txt", usecols=(0,1),unpack=True )


In [4]: x

Out[4]:

array([0.      , 0.063467, 0.126933, 0.1904  , 0.253866, 0.317333,

       0.380799, 0.444266, 0.507732, 0.571199, 0.634665, 0.698132,

       0.761598, 0.825065, 0.888531, 0.951998, 1.015464, 1.078931,

       1.142397, 1.205864, 1.26933 , 1.332797, 1.396263, 1.45973 ,

       1.523196, 1.586663, 1.650129, 1.713596, 1.777063, 1.840529,

       1.903996, 1.967462, 2.030929, 2.094395, 2.157862, 2.221328,

       2.284795, 2.348261, 2.411728, 2.475194, 2.538661, 2.602127,

       2.665594, 2.72906 , 2.792527, 2.855993, 2.91946 , 2.982926,

       3.046393, 3.109859, 3.173326, 3.236792, 3.300259, 3.363725,

       3.427192, 3.490659, 3.554125, 3.617592, 3.681058, 3.744524,

       3.807991, 3.871458, 3.934924, 3.998391, 4.061857, 4.125324,

       4.18879 , 4.252257, 4.315723, 4.37919 , 4.442656, 4.506123,
       4.569589, 4.633056, 4.696522, 4.759989, 4.823455, 4.886922,
       4.950388, 5.013855, 5.077322, 5.140788, 5.204255, 5.267721,
       5.331188, 5.394654, 5.45812 , 5.521587, 5.585053, 5.64852 ,
       5.711987, 5.775453, 5.83892 , 5.902386, 5.965853, 6.029319,
       6.092786, 6.156252, 6.219719, 6.283185])

In [5]: y
Out[5]:
array([ 0.      , 0.063424,  0.126592,  0.189251,  0.251148,  0.312033,
        0.371662,  0.429795,  0.486197,  0.540641,  0.592908,  0.642788,
        0.690079,  0.734592,  0.776146,  0.814576,  0.849725,  0.881453,
        0.909632,  0.934148,  0.954902,  0.971812,  0.984808,  0.993838,
        0.998867,  0.999874,  0.996855,  0.989821,  0.978802,  0.963842,
        0.945001,  0.922354,  0.895994,  0.866025,  0.83257 ,  0.795762,
        0.75575 ,  0.712694,  0.666769,  0.618159,  0.56706 ,  0.513677,
        0.458226,  0.400931,  0.34202 ,  0.281733,  0.220311,  0.158001,
        0.095056,  0.031728, -0.031728, -0.095056, -0.158001, -0.22031 ,
       -0.281733, -0.34202 , -0.400931, -0.458227, -0.513677, -0.56706 ,
       -0.618159, -0.666769, -0.712694, -0.75575 , -0.795762, -0.83257 ,
       -0.866025, -0.895994, -0.922354, -0.945001, -0.963842, -0.978802,
       -0.989821, -0.996855, -0.999874, -0.998867, -0.993838, -0.984808,
       -0.971812, -0.954902, -0.934148, -0.909632, -0.881453, -0.849725,
       -0.814576, -0.776146, -0.734592, -0.690079, -0.642788, -0.592908,
       -0.540641, -0.486197, -0.429795, -0.371662, -0.312033, -0.251148,
       -0.189251, -0.126592, -0.063424,  0.      ])

In [6]: plt.plot(x,y)
Out[6]: [<matplotlib.lines.Line2D at 0x1858ce9f408>]

In [7]: X=np.linspace(0,2*np.pi,100,endpoint=True)
In [8]: S=np.sin(X)


In [9]: plt.plot(X,S)

Out[9]: [<matplotlib.lines.Line2D at 0x18583a67e08>]


In [10]: Err=np.subtract(y,S)


In [11]: plt.plot(x,Err)

Out[11]: [<matplotlib.lines.Line2D at 0x18583ad8288>]


In [12]: plt.yscale('log')


In [13]: plt.plot(x,Err)

Out[13]: [<matplotlib.lines.Line2D at 0x185918f83c8>]


In [14]:
```



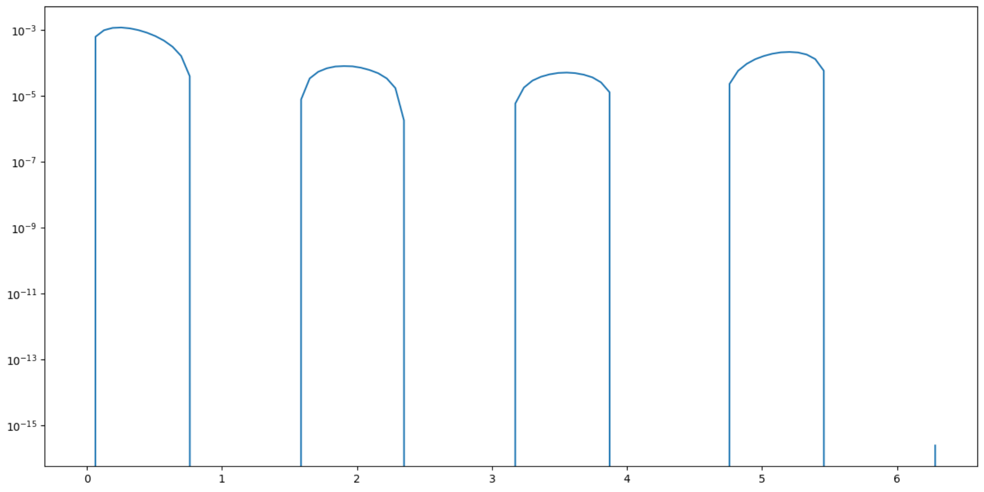
Plot1: INTERPOLATED SINE GRAPH



PLOT 2: ACTUAL SINE GRAPH



PLOT 3: ERROR FUNCTION VS x WHEN y-SCALE IS A NORMAL SCALE



PLOT 4: ERROR FUNCTION WHEN y-SCALE IS A LOGARITHMIC SCALE

Question 5 Code And Graph

C Code

```
#include<stdio.h>

#include<conio.h>

#include<math.h>


/*Function to evaluate Pi(x)*/


float Pi(int i, int n, float XX[n+1], float x){

   int j;

   double product=1;

   for(j=0;j<=n;j++){

      if(j!=i)

         product=product*(x-XX[j])/(XX[i]-XX[j]);

   }

      return product;

}


/*Function to evaluate lintp(x) where lintp is the Lagrange interpolating polynomial of degree n*/


float lintp(int n, float XX[n+1], float YY[n+1], float x){

   double sum=0;

   int i;

   for(i=0;i<=n;i++){

      sum=sum+Pi(i,n,XX,x)*YY[i];

   }

   return sum;

}
```

```
main(){
   int i,n=3,N;  //n is the number of neighbouring point using which we have to interpolate the
function
   float pi=3.14159;

   printf("Enter the number of data-points:\n");
   scanf("%d",&N);  //no. of data-points is N
   //Arrays to store the (N+1) x and y data-points of size N+1
   //XX are the array of 4 i.e n+1 nearest point for a point x at which we want the interpolated value
   // xx are the array of the N+1 points
   float XX[n];
   float YY[n];
   float xx[N];
   float yy[N];
   //printf("Enter the x data-points:\n");

   //printf("The xx data-points:\n");

   for(i=0;i<=N;i++){
      //scanf("%f",&xx[i]);
      xx[i]=i*2*pi/N;
      yy[i]=sin(xx[i]);
     // printf("\n %f",xx[i]);
      //printf(" %f",yy[i]);
     // printf("\n");
   }

   float x;  //value of x for which interpolated value is required
   //printf("\nValues of x and its interpolated value y(x):\n");
   for(int k=0;k<100;k++){
      x=k*pi*2/99;
```

```
printf("\n %f",x);


//Logic for Searching of the point x
unsigned long ju,jm,jl,j;
int ascnd;
int t;
jl=0;
ju=N;
ascnd=(xx[N] >= xx[1]);
while (ju-jl > 1) {
    jm=(ju+jl) >> 1;
    if (x >= xx[jm] == ascnd){
        jl=jm;
    }
    else{
        ju=jm;
    }
}
if (x == xx[0]) j=0;
else if(x == xx[N]) j=N;
else if(x >= xx[N]) j=N;
else if(x <= xx[0]) j=0;
else j=jl;


//Logic for the Generating the 4 neighbouring Points of x
 if(j==0){
    for(i=0;i<=3;i++){
        XX[i]=xx[j];
        YY[i]=yy[j];
        j=j+1;
```

```
      }
}

else if(j==(N-1)){
          t=j+1;
          for(i=0;i<=3;i++){
              XX[i]=xx[t];
              YY[i]=yy[t];
              t=t-1;
          }
}

else if(j==N) {
      for(i=0;i<=3;i++){
         XX[i]=xx[j];
         YY[i]=yy[j];
         j=j-1;
      }
}
else {
          t=j-1;
          for(i=0;i<=3;i++){
              XX[i]=xx[t];
              YY[i]=yy[t];
              t=t+1;
          }
}
//printf("\nThe nearest 4 Values of x are: ");
//for(i=0;i<=3;i++){
     //printf("\n %f",XX[i]);
```

```
            //printf("\n");
        //}


        printf(" %f",lintp(n,XX,YY,x)); // Calling lintp function and generating the interpolated value
        printf("\n");
    }

}
```

Output.txt For Question 5

```
0.000000 0.000006

 0.063466 0.105540

 0.126933 0.204378

 0.190399 0.296637

 0.253866 0.382469

 0.317332 0.462017

 0.380799 0.535426

 0.444265 0.602840

 0.507732 0.664402

 0.571198 0.720257
```

0.634665	**0.770548**
0.698131	**0.815419**
0.761598	**0.855014**
0.825064	**0.889477**
0.888531	**0.918952**
0.951997	**0.943582**
1.015463	**0.963513**
1.078930	**0.978886**
1.142396	**0.989847**
1.205863	**0.996540**
1.269329	**0.999107**
1.332796	**0.997694**
1.396262	**0.992443**
1.459729	**0.983500**
1.523195	**0.971007**

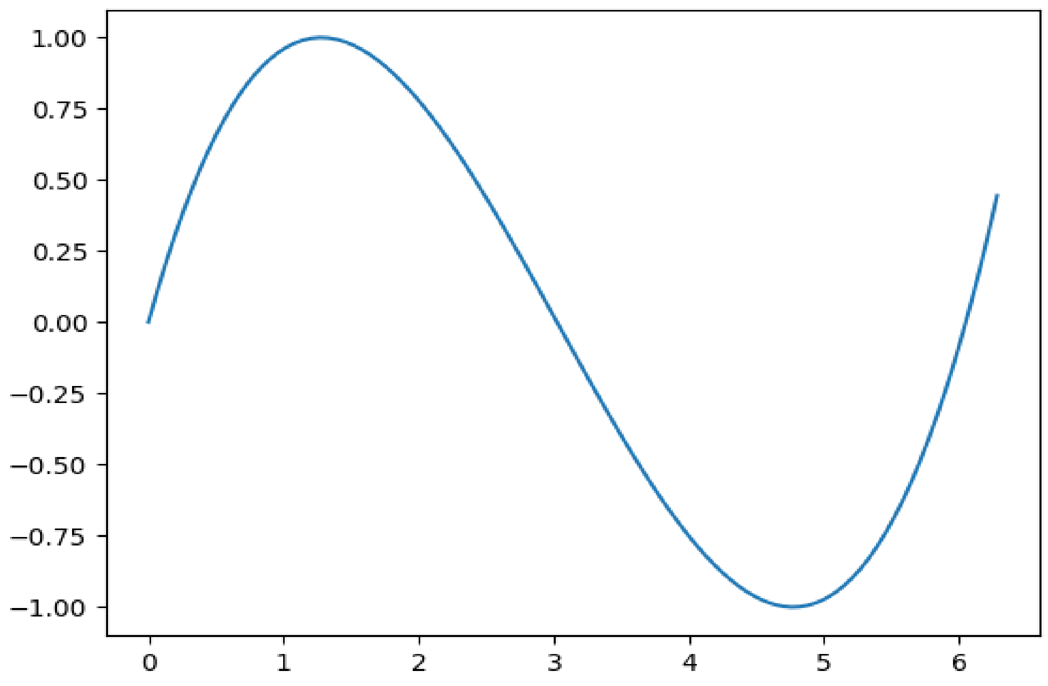
1.586662	**0.955108**
1.650128	**0.935948**
1.713595	**0.913671**
1.777061	**0.888420**
1.840528	**0.860339**
1.903994	**0.829572**
1.967461	**0.796264**
2.030927	**0.760557**
2.094393	**0.722595**
2.157860	**0.682524**
2.221326	**0.640486**
2.284793	**0.596625**
2.348259	**0.551086**
2.411726	**0.504012**
2.475192	**0.455547**

2.538659	**0.405835**
2.602125	**0.355020**
2.665592	**0.303245**
2.729058	**0.250656**
2.792525	**0.197395**
2.855991	**0.143606**
2.919457	**0.089433**
2.982924	**0.035021**
3.046391	**-0.019487**
3.109857	**-0.073947**
3.173323	**-0.128215**
3.236790	**-0.182147**
3.300256	**-0.235600**
3.363723	**-0.288429**
3.427189	**-0.340490**

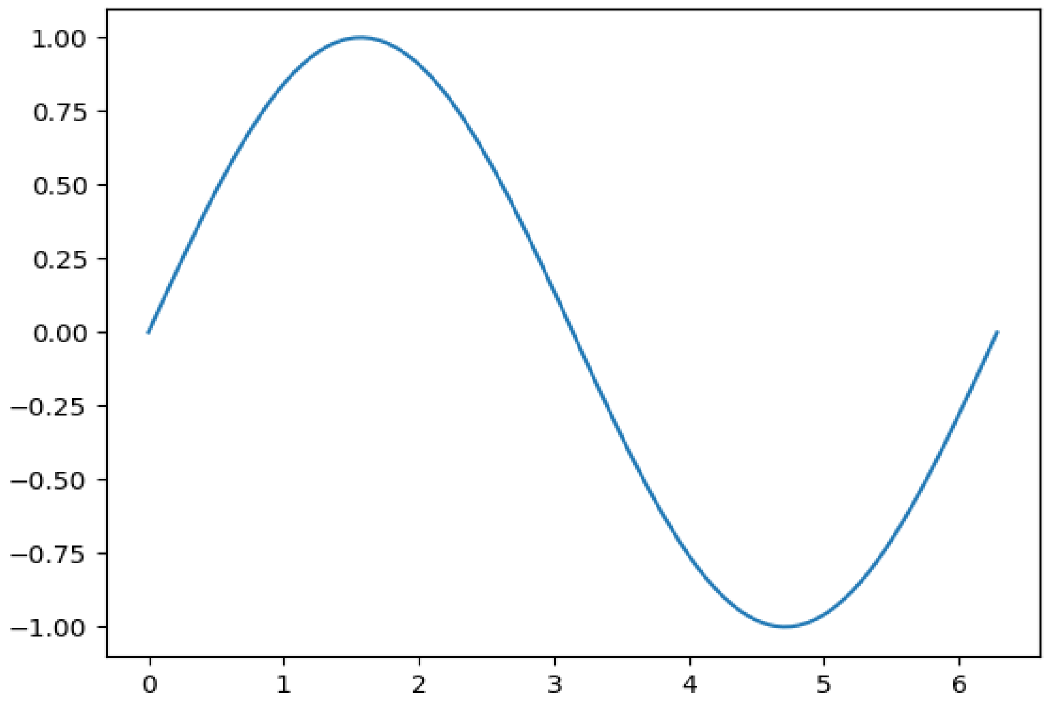
```
3.490656 -0.391641

3.554122 -0.441736

3.617589 -0.490631

3.681055 -0.538184

3.744522 -0.584250

3.807988 -0.628685

3.871454 -0.671346

3.934921 -0.712087

3.998387 -0.750767

4.061854 -0.787240

4.125320 -0.821362

4.188787 -0.852991

4.252253 -0.881981

4.315720 -0.908190

4.379186 -0.931473
```

4.442653	-0.951686
4.506119	-0.968685
4.569586	-0.982327
4.633052	-0.992467
4.696518	-0.998963
4.759985	-1.001669
4.823452	-1.000442
4.886918	-0.995138
4.950385	-0.985613
5.013851	-0.971723
5.077317	-0.953325
5.140784	-0.930274
5.204250	-0.902427
5.267717	-0.869640
5.331183	-0.831768

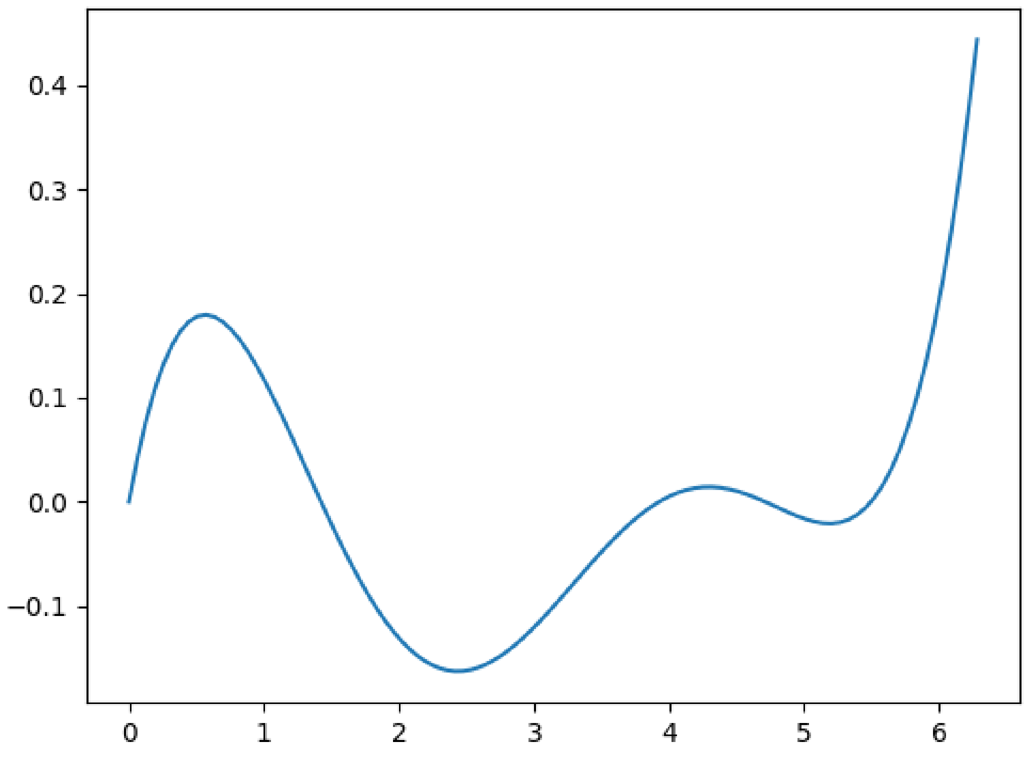
5.394650	-0.788668
5.458116	-0.740197
5.521583	-0.686209
5.585049	-0.626562
5.648516	-0.561111
5.711982	-0.489712
5.775448	-0.412223
5.838915	-0.328497
5.902381	-0.238392
5.965848	-0.141765
6.029315	-0.038470
6.092781	0.071635
6.156247	0.188695
6.219714	0.312854
6.283180	0.444256



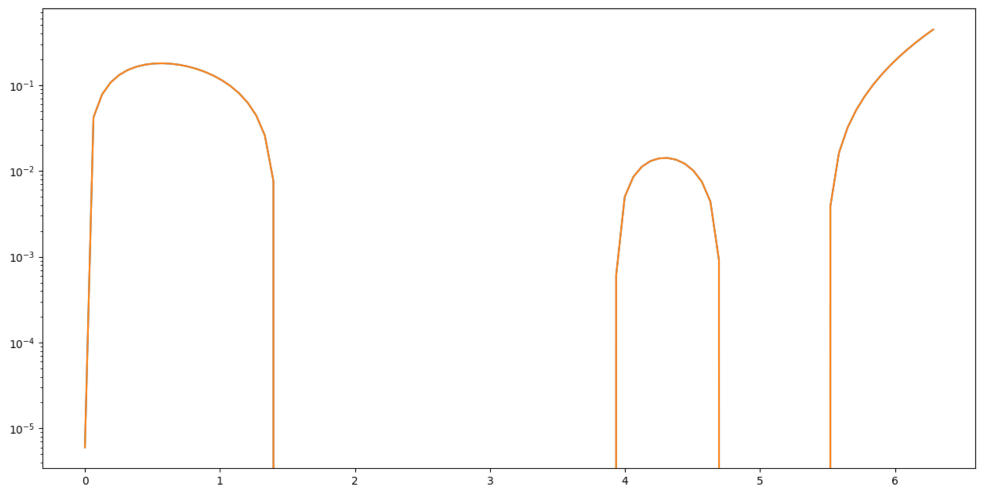
Plot 1: Interpolated Graph of Sine



Plot 2: ACTUAL SINE GRAPH



Plot3 : Error on Normal y-Scale



Plot 4: Error on Logarithmic y-Scale