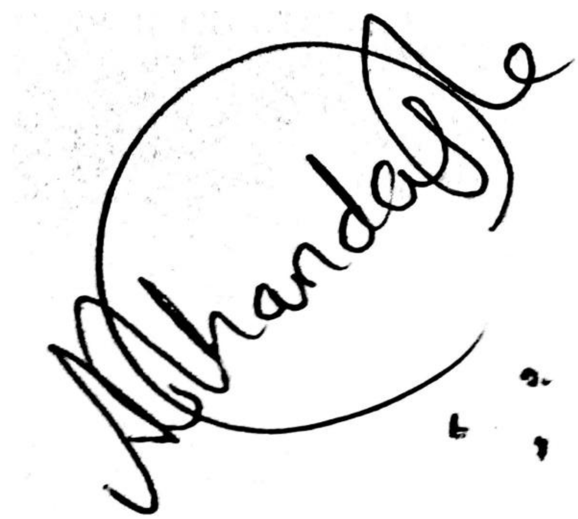
SCIENTIFIC COMPUTING

MID-SEM ASSIGNMENT

ANURAAG KHANDAGLE – 130123007

DATE – 04/10/2015

SIGNATURE 1



SIGNATURE 2

TABLE OF CONTENTS

Contents

ALGORITHM AND CODE

- In the cubic spline method, the function is constructed using –

$S_j = a_j + b_j (x-x_j) + c_j(x- x_j)^2 + d_j (x- x_j)^3$, given the values of a's, we find the values of b, c, d for different intervals along the entire curve of the function.

- In the assignement , we use Parametric Cubic Spline.
- We read the x and y coordinates from an excel file which are actually the points that are extracted from the image. We run a paramter t for the given n points and plot x vs t and y vs t.

t	0	1	$\dots$	n
x	x_0	x_1	$\dots$	x_n
y	y_0	y_1	$\dots$	y_n

- That means, we construct a cublic spline both for x coordinates and y coordinates. Let the cubic spline for x coordinates be X(t) and Y(t) for y coordinates.That is, we have actually generate the values of bx,cx,dx and by,cy and dy.
- Now in order to know the value of X(t) and Y(t), we divide each interval of t into 10 equal parts and calculate the X(t) and Y(t) in the following way as seen in the code :

```
x[i]= (dx[temp]*pow(tp,3))+ (cx[temp]*pow(tp,2))+ (bx[temp]*pow(tp,1))+ x[temp];
    y[i]= (dy[temp]*pow(tp,3))+ (cy[temp]*pow(tp,2))+ (by[temp]*pow(tp,1))+ y[temp];
```

temp represents the index of the different intervals.

tp represents t-t[i], where t[i-1]<=t<=t[i].

- Example : If we have 20 points , then we have 19 intervals.We divide each interval into 10 parts in order to plot the curve.
- Now each interval of the paramater has unique values of a, b, c and d. Now, x-x[j] here is equivalent to t-t[j] both for X(t) and Y(t) where, t[i-1] <=t<=t[i].
- Therefore, in this we have a plot of X(t) and Y(t). Now, these values of X(t) and Y(t) actually plot the function.

ALGORITHM AND CODE

```cpp
/*                              SCIENTIFIC COMPUTING
                                  MIDSEM ASSIGNMENT
****************************************************************************************

NAME - ANURAAG KHANDAGLE
ROLL NUMBER - 130123007


****************************************************************************************
*/
#include <iostream>
#include<bits/stdc++.h>                 // library to reduce the time
#include<cmath>                         //library to use the modf function - to extract the fractional part of a number
using namespace std;

int main()
{
    int ch;
    char shift;

    ifstream fin;                       // to access the .csv file for reading the extracted points
    ofstream fout;                      // to output the points in a .csv file inorder to plot the image
    cout<<"\nEnter the choice : 1) Spiral curve 2) Signature1 (requires less than 20 points) 3) Signature2 (requires more than 20 points)  :";
    cin >>ch;
    int n;                              // number of points taken
    if(ch==1)
    {

    fin.open("q1data.csv");                      //opening the .csv file to read the points
    fout.open("q1ans.csv");
    cout<<"\nEnter the total number of points extracted : ";
    cin>>n;
    }
    if(ch==2)
    {
    fin.open("q2data.csv");
```

ALGORITHM AND CODE

```
fout.open("q2ans.csv");
cout<<"\nEnter the total number of points extracted : ";
cin>>n;
}
if(ch==3)
{
    fin.open("q3data.csv");
fout.open("q3ans.csv");
cout<<"\nEnter the total number of points extracted : ";
cin>>n;
}
float betay[n], betax[n];                                    //variables to store intermediate values
    float check;
 float xpoint[n+1], ypoint[n+1], par[n+1], h[n+1];          //par is the parameter
 float l[n], mu[n], zx[n], zy[n];                            //variables to store intermediate values
 float bx[n], by[n], cx[n], cy[n], dx[n], dy[n];           // b,c and d values for X and Y
 float xout[n*10], yout[n*10];                              // final points to plot the image
  float s[n*10];

//  int n=19;

for(int i=0; i<n+1; i++)
{
  par[i]= 0.0 + i;
  h[i]=1.0;                   //  1 since h[i]=par[i+1]-par[i] is always equal to 1
}

for(int i=0; i<n+1; i++)
{
    fin>> xpoint[i];          //reading the x coordinates of the image from a .csv file
    fin>> shift;               //to read the commas or space between cooordinates
    fin>> ypoint[i];          //reading the y coordinates
}

for(int i=1; i<n; i++)
```

ALGORITHM AND CODE

```
{
    betay[i]= 3*(ypoint[i+1]-2*ypoint[i]+ypoint[i-1]);        //calculating betax and betay for X and Y in the same loop
    betax[i]= 3*(xpoint[i+1]-2*xpoint[i]+xpoint[i-1]);
}

l[0]= 1.0;                                                    //initialising the internediate variables
mu[0]= 0.0;
zy[0]= 0.0;
zx[0]= 0.0;
for(int i=1; i<n; i++)                                         //loop to calculate the intermediate values
{
    l[i]= 2*(par[i+1]-par[i-1]) - (mu[i-1]);
    mu[i]= (1/l[i]);
    zx[i]= (betax[i]-(1*zx[i-1]))/l[i];
    zy[i]= (betay[i]-(1*zy[i-1]))/l[i];
}
l[n]=1.0;
zx[n]=0.0;
zy[n]=0.0;

cx[n]=0.0;
cy[n]=0.0;
for(int i=(n-1); i>=0; i--)                                    //loop to calculate the b,c and d values for X and Y respectively
{
   cx[i]=  zx[i] - (mu[i]*cx[i+1]);
   bx[i]=  (xpoint[i+1]-xpoint[i]) - (cx[i+1]+2*cx[i])/3;
   dx[i]=  (cx[i+1]-cx[i])/3;
   cy[i]=  zy[i] - (mu[i]*cy[i+1]);
   by[i]=  (ypoint[i+1]-ypoint[i]) - (cy[i+1]+2*cy[i])/3;
   dy[i]=  (cy[i+1]-cy[i])/3;
}

s[0]=0.0;
for(int i=1; i<((n)*10); i++)                          // dividing the par intervals into 10 parts for the final plotting
{
    s[i]= 0.1*(i-1);
```

ALGORITHM AND CODE

```
    }

    int temp;

    for(int i=0; i< (n*10); i++)
    {
        temp= int(s[i]);
       // cout<<temp<<endl;
        check=temp;

        float tp=modf(s[i],&check);             //using modf function to find the fractional part (cmath.h)
        //tp= s[i]- temp;
        //cout<<tp<<endl;
        for(int j=0;j<n;j++)
        if(temp==j)
        {
        xout[i]= (dx[temp]*pow(tp,3))+ (cx[temp]*pow(tp,2))+ (bx[temp]*pow(tp,1))+ xpoint[temp];
        yout[i]= (dy[temp]*pow(tp,3))+ (cy[temp]*pow(tp,2))+ (by[temp]*pow(tp,1))+ ypoint[temp];
         }
    }
    for(int i=0; i<(n-2)*10; i++)
    {
        fout<< xout[i]<<","<< -1*yout[i]<<endl;
    }
    if(ch==1)
    cout<<"\n The coordinates to plot the spiral curve using cubic spline are put in q1ans.csv";
    if (ch==2)
    cout<<"\n The coordinates to plot the signature1 using cubic spline are put in q2ans.csv";
    if (ch==3)
    cout<<"\n The coordinates to plot the signature2 using cubic spline are put in q2ans.csv";
    fin.close();
    fout.close();
    return 0;
}
```

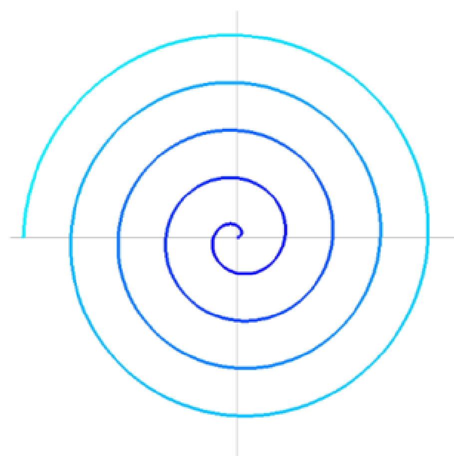
QUESTION -01

OBJECTIVE-

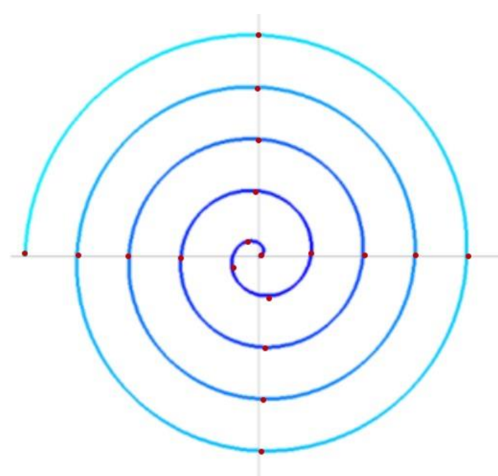
- To draw a spiral and reproduce it using parametric spline functions

POINTS EXTRACTION

- Image extracted using – WebPlotDigitizer
- Image before extraction of points



- Image after extraction of points. The red points indicate the points selected

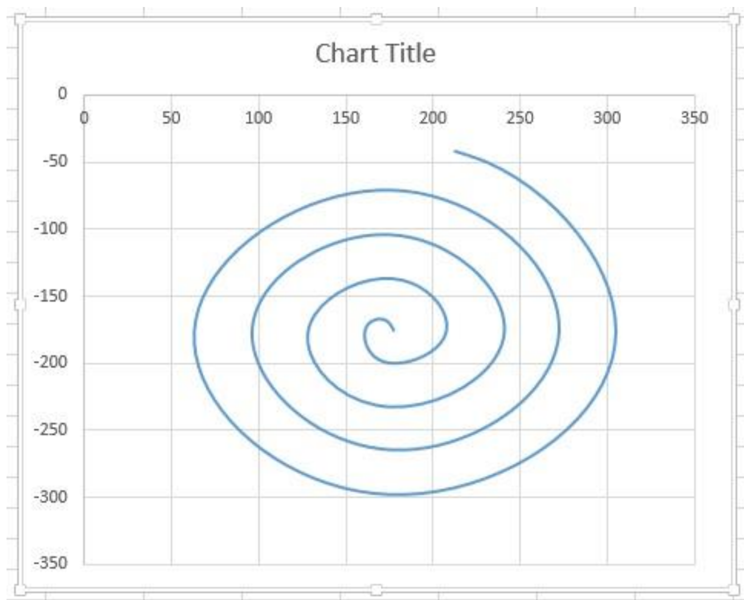


QUESTION -01

- Total points extracted =19

PLOTTING THE IMAGE USING CUBIC SPLINE

- Image after running the Cubic Spline algorithm for the spiral curve given



- Image plotted using Microsoft Excel

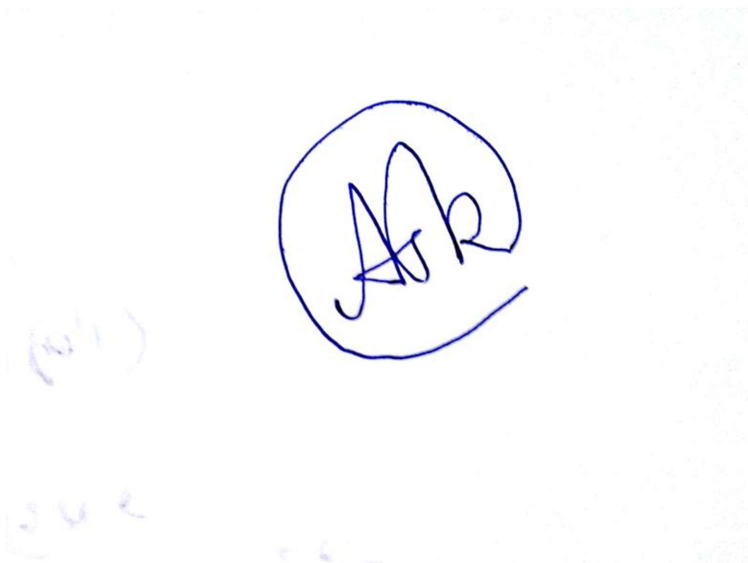
QUESTION 02

OBJECTIVE-

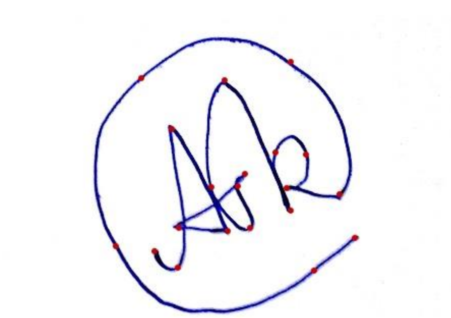
- To plot our own signature using maximum 20 knots and cubic spline

POINTS EXTRACTION

- Points extracted using WebPlotDigitizer
- Image before extraction



- Image after extracting the points. The red points indicate the points selected



- Total points extracted - 20

QUESTION 02

PLOTTING THE IMAGE USING CUBIC SPLINE

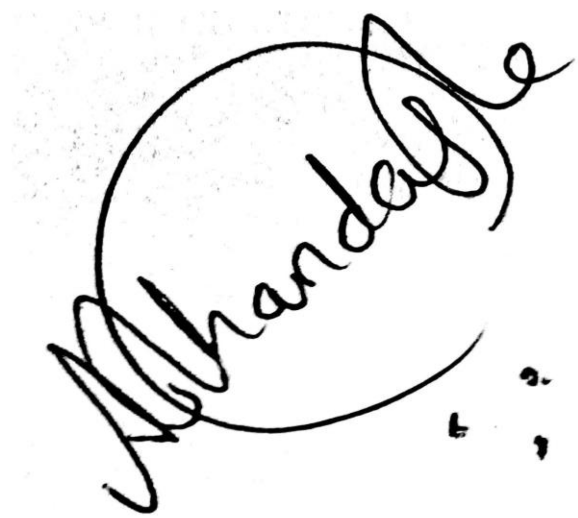
- Image after running the Cubic Spline algorithm for the spiral curve given

- Image plotted using Microsoft Excel.

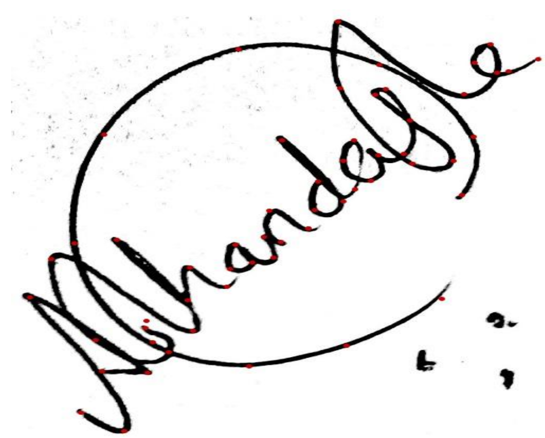
QUESTION 02

Signature 2 (more than 20 points)

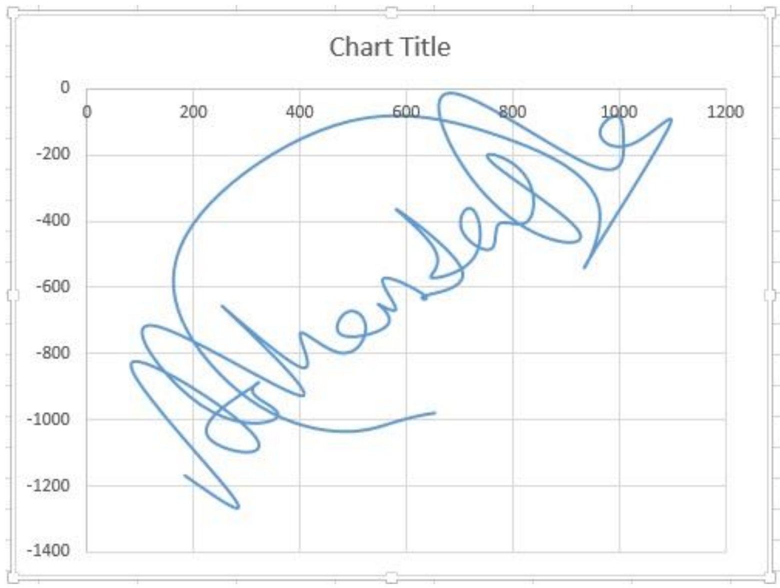
- Original Image



- Image after extracting the points



QUESTION 02



- Image after plotting

CONCLUSION

- The parametric spline equation were successfully found out and also the image was plotted
- The plot requires more number of initial points for curves having a lot of curvatures.
- Cubic lines prove to be the best for continous curves.