MA 322: Midsem Assignment

Due on Sunday, October 5, 2015

Jiten Chandra kalita

Silvi Pandey-130123045

Contents

PROBLEM 1

Draw a spiral using parametric spline functions

SOLUTION

Steps involved in making the spiral :-

(1)Spiral was plotted on a graph paper and 10 nodal points were taken i.e. 10 (x,y) coordinates were taken to generate the whole spiral.

(2)The points were chosen such that the spiral could be generated accurately i.e. the points were the curvature of the spiral changes a considerable amount were necessarily included among the nodal points.

(3)These points were parameterized taking t as a parameter which varies from 0 to 9(i.e. equally spaced).

(4)Cubic spline was used to generate x as a function of the parameter t.Similarly, y was generated as a function of t.

(5)With t varying from 0 to 9 with a difference of 0.01,the x(t) and y(t) points corresponding to these parameters are taken and **GNUPLOT** is used to plot x(t) against y(t) and the spiral is generated.

CODE

```
 #include <iostream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
     OBJECTIVE
     To generate a spiral like structure using cubic spline and parameterization.

     NUMBER OF PARAMETERS VALUES USED
     10 (0-9)

     METHOD
     1)Spiral was plotted on a graph sheet and 10 co-ordinates were taken as nodal
       points.Co-ordinates were chosen such that the structure of the spiral could
       be retrieved properly.
     2)With t as a parameter which varies from 0 to 9 (equally spaced) , x and y
       both are inter-polated using cubic spline.
     3)Plot (x(t),y(t)) from t = 0 to 9 with difference of 0.01. i.e for t = 0,0.01
    ,0.02....9 (x(t),y(t)) are plotted and the spiral is generated.

     LANGUAGES/SOFTWARE USED
     1)C++
     2)Gnuplot

*/
using namespace std;
int low = 0;
int high = 9;
int points = 10;
double h = (double)(high - low)/(points - 1);

double functionx(double t);
```

```
double functiony(double t);
void ThomasMethod(double c[],double a[]);
double poly(double x,int j,double a,double b,double c,double d);

int main()
{
     FILE *fp=fopen("plotSpiral.txt","w");
     double coefficientx[points-1][4],coefficienty[points-1][4];
     double ax[points],cx[points-2],ay[points],cy[points-2];
     for(int i=0;i<points;i++)
          ax[i]=functionx(low+i*h);
     for(int i=0;i<points;i++)
          ay[i]=functiony(low+i*h);
     ThomasMethod(cx,ax);
     ThomasMethod(cy,ay);
     for(int i=0;i<points-1;i++)
     {
          coefficientx[i][0]=ax[i];
          if(i==0)
               coefficientx[i][2]=0;
          else
               coefficientx[i][2]=cx[i-1];
     }
     for(int i=0;i<points-1;i++)
     {
          if(i!=0)
               coefficientx[i][1]=(ax[i+1]-ax[i])/h-h*(2*cx[i-1]+cx[i])/3;
          else
               coefficientx[i][1]=(ax[i+1]-ax[i])/h-h*cx[i-1]/3;
     }
     for(int i=0;i<points-1;i++)
     {
          if(i==points-2)
               coefficientx[i][3]=-coefficientx[i][2]/(3*h);
          coefficientx[i][3]=(coefficientx[i+1][2]-coefficientx[i][2])/(3*h);
     }
     cout<<"Coefficients for x co-ordinate"<<endl;
     for(int i=0;i<points-1;i++)
     {
          printf("%0.6lf %0.6lf %0.6lf %0.6lf\n",coefficientx[i][0],coefficientx[i][1],
          coefficientx[i][2],coefficientx[i][3]);
     }
     cout<<endl;
     for(int i=0;i<points-1;i++)
     {
          coefficienty[i][0]=ay[i];
          if(i==0)
               coefficienty[i][2]=0;
          else
               coefficienty[i][2]=cy[i-1];
     }
     for(int i=0;i<points-1;i++)
     {
```

```
            if(i!=0)
                coefficienty[i][1]=(ay[i+1]-ay[i])/h-h*(2*cy[i-1]+cy[i])/3;
            else
                coefficienty[i][1]=(ay[i+1]-ay[i])/h-h*cy[i-1]/3;
        }
        cout<<"Coefficients for y co-ordinate"<<endl;
        for(int i=0;i<points-1;i++)
        {
            if(i==points-2)
                coefficienty[i][3]=-coefficienty[i][2]/(3*h);
            coefficienty[i][3]=(coefficienty[i+1][2]-coefficienty[i][2])/(3*h);
        }
        for(int i=0;i<points-1;i++)
        {
            printf("%0.6lf %0.6lf %0.6lf %0.6lf\n",coefficienty[i][0],coefficienty[i][1],
            coefficienty[i][2],coefficienty[i][3]);
         }
        double t;
        int j;
        for(t=0;t<=9;t=t+0.01)
         {
            j = floor(t);
            fprintf(fp,"%lf %lf\n",poly(t,j,coefficientx[j][0],coefficientx[j][1],
            coefficientx[j][2],coefficientx[j][3]),poly(t,j,coefficienty[j][0],
            coefficienty[j][1],coefficienty[j][2],coefficienty[j][3]));
            }
}

//Cubic polynomial
double poly(double x,int j,double a,double b,double c,double d)
{
        return a+b*(x-(low+j*h))+c*(x-(low+j*h))*(x-(low+j*h))+d*(x-(low+j*h))*
        (x-(low+j*h))*(x-(low+j*h));
}

//Value of x for t = 0,1,2..,9
//Nodal points
double functionx(double t)
{
     if(t == 0)
         return 3;
     if(t == 1)
         return 3.5;
     if(t == 2)
         return 4;
     if(t == 3)
         return 3;
     if(t == 4)
         return 2;
     if(t == 5)
         return 3;
     if(t == 6)
         return 5;
```

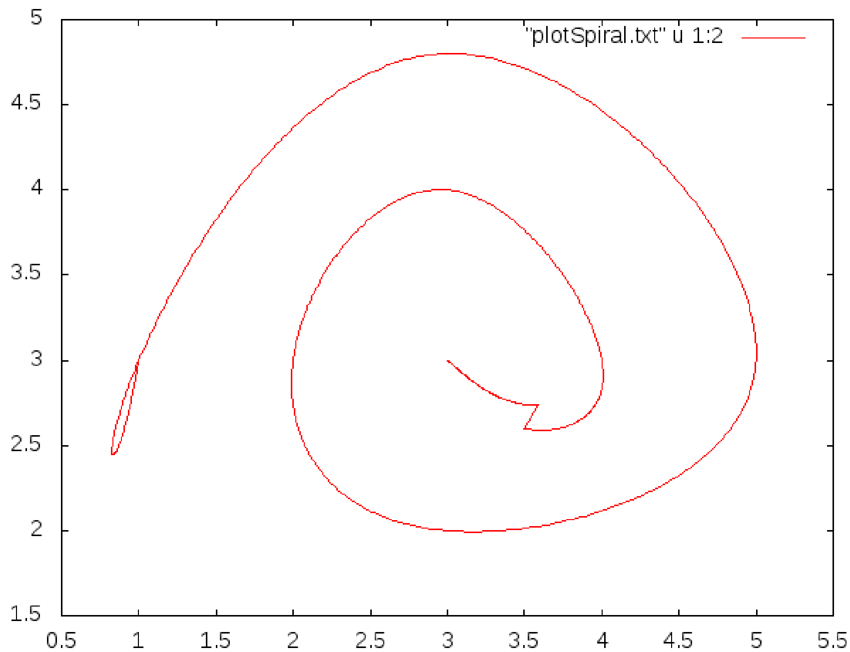
```
    if(t == 7)
        return 3;
    if(t == 8)
        return 1;
    if(t == 9)
        return 1;
}

//Value of y for t = 0,1,2,..9
//Nodal points
double functiony(double t)
{
    if(t == 0)
        return 3;
    if(t == 1)
        return 2.6;
    if(t == 2)
        return 3;
    if(t == 3)
        return 4;
    if(t == 4)
        return 3;
    if(t == 5)
        return 2;
    if(t == 6)
        return 3;
    if(t == 7)
        return 4.8;
    if(t == 8)
        return 3;
    if(t == 9)
        return 2;
}

//Thomas method to find the coefficients.
void ThomasMethod(double c[],double a[])
{
     double   A[points-2],B[points-2],C[points-2],F[points-2];
     for(int i=0;i<points-2;i++)
     {
          A[i]=h;
          B[i]=4*h;
          C[i]=h;
          F[i]=(3*(a[i+2]-a[i+1])-3*(a[i+1]-a[i]))/h;
     }
     for(int i=1;i<points-2;i++)
     {
          B[i]=B[i]-C[i]*A[i]/B[i-1];
          F[i]=F[i]-F[i-1]*A[i]/B[i-1];
     }
     c[points-3]=F[points-3]/B[points-3];
     for(int i=points-4;i>=0;i--)
     {
```

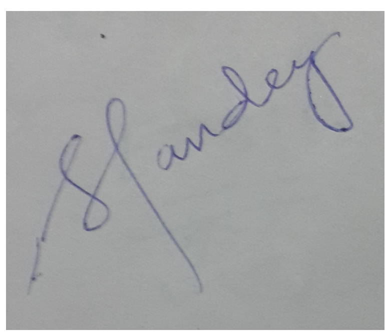
```
            c[i]=(F[i]-C[i]*c[i+1])/B[i];
      }
}
```

RESULT



PROBLEM 2

Generate your own signature using minimum number of nodal points.The following signature is used to generate the signature.



SOLUTION

Steps involved in making the signature :-

(1)Signature was plotted on a graph paper and firstly 12 nodal points were taken i.e. 12 (x,y) coordinates were taken to generate the first part of the signature because the signature is not continuous.Pen is lifted after the first two alphabets.Two separate set of cubic splines are used to generate the two parts separately and then are plotted together.For the second part,28 more points are taken and the process is repeated.

(2)The points were chosen such that the signature could be generated accurately i.e. the points were the curvature of the spiral changes a considerable amount were necessarily included among the nodal points.Also the crossing points are taken as many times they are crossed

(3)These points were parameterized taking t as a parameter which varies from 0 to 11(i.e. equally spaced)

and from 0 to 27 for the second part.
(4)Cubic spline was used to generate x as a function of the parameter t.Similarly, y was generated as a function of t.
(5)With t varying from 0 to 11 with a difference of 0.01,the x(t) and y(t) points corresponding to these parameters are taken and **GNUPLOT** is used to plot x(t) against y(t) and the first part of the signature is generated .In the same plot,second part of the signature is also plotted varying t from 0 to 27 with a difference of 0.01.

CODE
PART 1

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
     OBJECTIVE
     To generate signature using cubic spline and parameterization.

     NUMBER OF PARAMETERS VALUES USED
     12 (0-11)

     PART 1 OF SIGNATURE
     My signature is not continuous.I lift pen after first two alphabets.This is for
        the first part of the signature.

     METHOD
     1)Signature was plotted on a graph sheet and 12 co-ordinates were taken as nodal
       points.Co-ordinates were chosen such that the structure of the signature could
       be retrieved properly.
     2)With t as a parameter which varies from 0 to 11 (equally spaced) , x and y
       both are inter-polated using cubic spline.
     3)Plot (x(t),y(t)) from t = 0 to 11 with difference of 0.01. i.e for t = 0,0.01,
     0.02....11 (x(t),y(t)) are plotted and the first part of the signature is generated.

     LANGUAGES/SOFTWARE USED
     1)C++
     2)Gnuplot

*/

using namespace std;
int low = 0;
int high = 11;
int points = 12;
double h = (double)(high - low)/(points - 1);

double functionx(double t);
double functiony(double t);
void ThomasMethod(double c[],double a[]);
double poly(double x,int j,double a,double b,double c,double d);
```

```
int main()
{
     FILE *fp=fopen("plotSignature1.txt","w");
     double coefficientx[points-1][4],coefficienty[points-1][4];
     double ax[points],cx[points-2],ay[points],cy[points-2];
     for(int i=0;i<points;i++)
          ax[i]=functionx(low+i*h);
    for(int i=0;i<points;i++)
          ay[i]=functiony(low+i*h);
     ThomasMethod(cx,ax);
     ThomasMethod(cy,ay);
     for(int i=0;i<points-1;i++)
     {
          coefficientx[i][0]=ax[i];
          if(i==0)
               coefficientx[i][2]=0;
          else
               coefficientx[i][2]=cx[i-1];
     }
     for(int i=0;i<points-1;i++)
     {
          if(i!=0)
               coefficientx[i][1]=(ax[i+1]-ax[i])/h-h*(2*cx[i-1]+cx[i])/3;
          else
               coefficientx[i][1]=(ax[i+1]-ax[i])/h-h*cx[i-1]/3;
     }
     for(int i=0;i<points-1;i++)
     {
          if(i==points-2)
               coefficientx[i][3]=-coefficientx[i][2]/(3*h);
          coefficientx[i][3]=(coefficientx[i+1][2]-coefficientx[i][2])/(3*h);
     }
     for(int i=0;i<points-1;i++)
    {
        printf("%0.6lf %0.6lf %0.6lf %0.6lf\n",coefficientx[i][0],coefficientx[i][1],
               coefficientx[i][2],coefficientx[i][3]);
    }
    cout<<endl;
     for(int i=0;i<points-1;i++)
     {
          coefficienty[i][0]=ay[i];
          if(i==0)
               coefficienty[i][2]=0;
          else
               coefficienty[i][2]=cy[i-1];
     }
     for(int i=0;i<points-1;i++)
     {
          if(i!=0)
               coefficienty[i][1]=(ay[i+1]-ay[i])/h-h*(2*cy[i-1]+cy[i])/3;
          else
               coefficienty[i][1]=(ay[i+1]-ay[i])/h-h*cy[i-1]/3;
     }
```

```
     for(int i=0;i<points-1;i++)
     {
          if(i==points-2)
               coefficienty[i][3]=-coefficienty[i][2]/(3*h);
          coefficienty[i][3]=(coefficienty[i+1][2]-coefficienty[i][2])/(3*h);
     }
     for(int i=0;i<points-1;i++)
    {
        printf("%0.6lf %0.6lf %0.6lf %0.6lf\n",coefficienty[i][0],coefficienty[i][1],
               coefficienty[i][2],coefficienty[i][3]);
    }
    double t;
    int j;
    for(t=0;t<=11;t=t+0.01)
     {
     j = floor(t);
     fprintf(fp,"%lf %lf\n",poly(t,j,coefficientx[j][0],coefficientx[j][1],
                             coefficientx[j][2],coefficientx[j][3]),
                            poly(t,j,coefficienty[j][0],coefficienty[j][1],
                                coefficienty[j][2],coefficienty[j][3]));
     }
}
double poly(double x,int j,double a,double b,double c,double d)
{
     return a+b*(x-(low+j*h))+c*(x-(low+j*h))*(x-(low+j*h))+d*(x-(low+j*h))*
     (x-(low+j*h))*(x-(low+j*h));
}
double functionx(double t)
{
    if(t == 0)
        return 1;
    if(t == 1)
        return 1;
    if(t == 2)
        return 1;
    if(t == 3)
        return 0.7;
    if(t == 4)
        return 1;
    if(t == 5)
        return 2;
    if(t == 6)
        return 1.5;
    if(t == 7)
        return 2;
    if(t == 8)
        return 2.2;
    if(t == 9)
        return 1.8;
    if(t == 10)
        return 2.2;
    if(t == 11)
        return 3.6;
```

```
}
double functiony(double t)
{
    if(t == 0)
        return 0.5;
    if(t == 1)
        return 2;
    if(t == 2)
        return 2.8;
    if(t == 3)
        return 3.5;
    if(t == 4)
        return 2.8;
    if(t == 5)
        return 2.2;
    if(t == 6)
        return 1;
    if(t == 7)
        return 2.2;
    if(t == 8)
        return 3.2;
    if(t == 9)
        return 4.4;
    if(t == 10)
     return 3.2;
    if(t == 11)
     return 1;
}
void ThomasMethod(double c[],double a[])
{
     double    A[points-2],B[points-2],C[points-2],F[points-2];
     for(int i=0;i<points-2;i++)
     {
          A[i]=h;
          B[i]=4*h;
          C[i]=h;
          F[i]=(3*(a[i+2]-a[i+1])-3*(a[i+1]-a[i]))/h;
     }
     for(int i=1;i<points-2;i++)
     {
          B[i]=B[i]-C[i]*A[i]/B[i-1];
          F[i]=F[i]-F[i-1]*A[i]/B[i-1];
     }
     c[points-3]=F[points-3]/B[points-3];
     for(int i=points-4;i>=0;i--)
     {
          c[i]=(F[i]-C[i]*c[i+1])/B[i];
     }
}
```

PART 2

```
#include <iostream>
#include <stdio.h>
```

```
#include <math.h>
#include <stdlib.h>

/*
     OBJECTIVE
     To generate signature using cubic spline and parameterization.

     NUMBER OF PARAMETERS VALUES USED
     28 (0-27)

     PART 2 OF SIGNATURE
     My signature is not continuous.I lift pen after first two alphabets.This is for
        the second part of the signature.

     METHOD
     1)Signature was plotted on a graph sheet and 28 co-ordinates were taken as nodal
       points.Co-ordinates were chosen such that the structure of the signature could
       be retrieved properly.
     2)With t as a parameter which varies from 0 to 27 (equally spaced) , x and y
       both are inter-polated using cubic spline.
     3)Plot (x(t),y(t)) from t = 0 to 27 with difference of 0.01. i.e for t = 0,0.01,
     0.02....27 (x(t),y(t)) are plotted and the second part of the signature is generated.
     4)Both the parts of the signature are plotted together using GNUPLOT and the final
      signature is obatined.

     LANGUAGES/SOFTWARE USED
     1)C++
     2)Gnuplot

*/


using namespace std;
int low = 0;
int high = 27;
int points = 28;
double h = (double)(high - low)/(points - 1);

double functionx(double t);
double functiony(double t);
void ThomasMethod(double c[],double a[]);
double poly(double x,int j,double a,double b,double c,double d);

int main()
{
     FILE *fp=fopen("plotSignature2.txt","w");
     double coefficientx[points-1][4],coefficienty[points-1][4];
     double ax[points],cx[points-2],ay[points],cy[points-2];
     for(int i=0;i<points;i++)
          ax[i]=functionx(low+i*h);
    for(int i=0;i<points;i++)
          ay[i]=functiony(low+i*h);
     ThomasMethod(cx,ax);
```

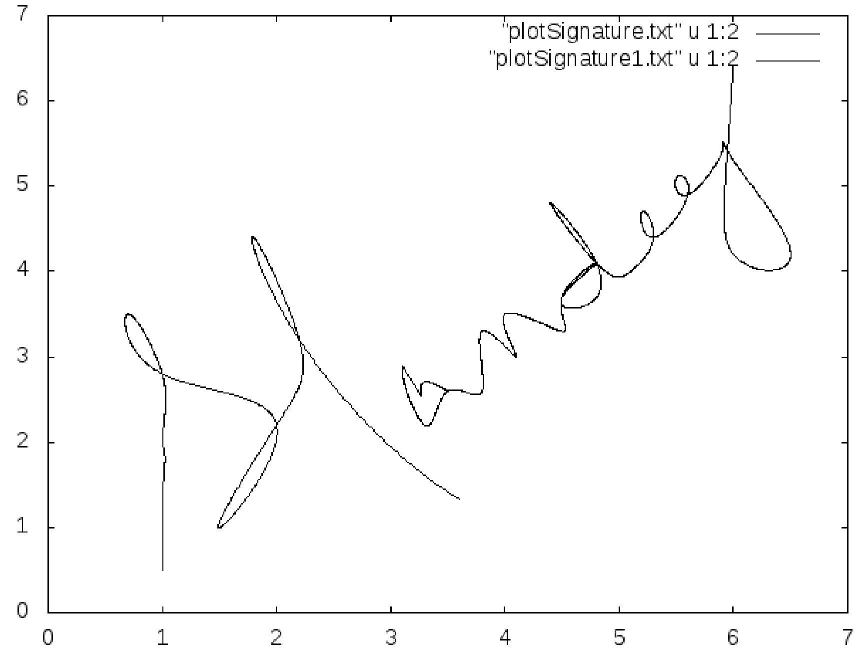
```
    ThomasMethod(cy,ay);
    for(int i=0;i<points-1;i++)
    {
        coefficientx[i][0]=ax[i];
        if(i==0)
            coefficientx[i][2]=0;
        else
            coefficientx[i][2]=cx[i-1];
    }
    for(int i=0;i<points-1;i++)
    {
        if(i!=0)
            coefficientx[i][1]=(ax[i+1]-ax[i])/h-h*(2*cx[i-1]+cx[i])/3;
        else
            coefficientx[i][1]=(ax[i+1]-ax[i])/h-h*cx[i-1]/3;
    }
    for(int i=0;i<points-1;i++)
    {
        if(i==points-2)
            coefficientx[i][3]=-coefficientx[i][2]/(3*h);
        coefficientx[i][3]=(coefficientx[i+1][2]-coefficientx[i][2])/(3*h);
    }
    for(int i=0;i<points-1;i++)
   {
      printf("%0.6lf %0.6lf %0.6lf %0.6lf\n",coefficientx[i][0],coefficientx[i][1],
            coefficientx[i][2],coefficientx[i][3]);
   }
   cout<<endl;
    for(int i=0;i<points-1;i++)
    {
        coefficienty[i][0]=ay[i];
        if(i==0)
            coefficienty[i][2]=0;
        else
            coefficienty[i][2]=cy[i-1];
    }
    for(int i=0;i<points-1;i++)
    {
        if(i!=0)
            coefficienty[i][1]=(ay[i+1]-ay[i])/h-h*(2*cy[i-1]+cy[i])/3;
        else
            coefficienty[i][1]=(ay[i+1]-ay[i])/h-h*cy[i-1]/3;
    }
    for(int i=0;i<points-1;i++)
    {
        if(i==points-2)
            coefficienty[i][3]=-coefficienty[i][2]/(3*h);
        coefficienty[i][3]=(coefficienty[i+1][2]-coefficienty[i][2])/(3*h);
    }
    for(int i=0;i<points-1;i++)
   {
      printf("%0.6lf %0.6lf %0.6lf %0.6lf\n",coefficienty[i][0],coefficienty[i][1],
            coefficienty[i][2],coefficienty[i][3]);
```

```
    }
    double t;
    int j;
    for(t=0;t<=27;t=t+0.01)
     {
     j = floor(t);
     fprintf(fp,"%lf %lf\n",poly(t,j,coefficientx[j][0],coefficientx[j][1],
                             coefficientx[j][2],coefficientx[j][3]),
                            poly(t,j,coefficienty[j][0],coefficienty[j][1],
                                coefficienty[j][2],coefficienty[j][3]));
     }
}
double poly(double x,int j,double a,double b,double c,double d)
{
     return a+b*(x-(low+j*h))+c*(x-(low+j*h))*(x-(low+j*h))+d*(x-(low+j*h))*
     (x-(low+j*h))*(x-(low+j*h));
}
double functionx(double t)
{
    if(t == 0)
        return 3.5;
    if(t == 1)
        return 3.1;
    if(t == 2)
        return 3.3;
    if(t == 3)
        return 3.5;
    if(t == 4)
        return 3.8;
    if(t == 5)
        return 3.8;
    if(t == 6)
        return 4.1;
    if(t == 7)
        return 4;
    if(t == 8)
        return 4.5;
    if(t == 9)
        return 4.5;
    if(t == 10)
        return 4.8;
    if(t == 11)
        return 4.5;
     if(t == 12)
     return 4.7;
     if(t == 13)
     return 4.8;
     if(t == 14)
     return 4.4;
     if(t == 15)
     return 4.8;
     if(t == 16)
     return 5.1;
```

```
     if(t == 17)
     return 5.3;
     if(t == 18)
     return 5.2;
     if(t == 19)
     return 5.3;
     if(t == 20)
     return 5.6;
     if(t == 21)
     return 5.5;
     if(t == 22)
     return 5.6;
     if(t == 23)
     return 5.9;
      if(t == 24)
     return 6;
       if(t == 25)
     return 6.5;
       if(t == 26)
     return 6;
       if(t == 27)
     return 6;
}
double functiony(double t)
{
    if(t == 0)
        return 2.6;
    if(t == 1)
        return 2.9;
    if(t == 2)
        return 2.2;
    if(t == 3)
        return 2.6;
    if(t == 4)
        return 2.6;
    if(t == 5)
        return 3.3;
    if(t == 6)
        return 3;
    if(t == 7)
        return 3.5;
    if(t == 8)
        return 3.3;
    if(t == 9)
        return 3.7;
    if(t == 10)
     return 4.1;
    if(t == 11)
     return 3.7;
     if(t == 12)
     return 3.6;
     if(t == 13)
     return 4.1;
```

```
    if(t == 14)
    return 4.8;
    if(t == 15)
    return 4.1;
    if(t == 16)
    return 4;
    if(t == 17)
    return 4.4;
    if(t == 18)
    return 4.7;
    if(t == 19)
    return 4.4;
    if(t == 20)
    return 4.9;
    if(t == 21)
    return 5.1;
    if(t == 22)
    return 4.9;
    if(t == 23)
    return 5.4;
     if(t == 24)
    return 5.3;
     if(t == 25)
    return 4.2;
     if(t == 26)
    return 4.2;
     if(t == 27)
    return 5.3;
}
void ThomasMethod(double c[],double a[])
{
    double   A[points-2],B[points-2],C[points-2],F[points-2];
    for(int i=0;i<points-2;i++)
    {
        A[i]=h;
        B[i]=4*h;
        C[i]=h;
        F[i]=(3*(a[i+2]-a[i+1])-3*(a[i+1]-a[i]))/h;
    }
    for(int i=1;i<points-2;i++)
    {
        B[i]=B[i]-C[i]*A[i]/B[i-1];
        F[i]=F[i]-F[i-1]*A[i]/B[i-1];
    }
    c[points-3]=F[points-3]/B[points-3];
    for(int i=points-4;i>=0;i--)
    {
        c[i]=(F[i]-C[i]*c[i+1])/B[i];
    }
}
```

RESULT


"plotSignature.txt" u 1:2
"plotSignature1.txt" u 1:2
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7