计算方法-第四次上机作业

PB19051183 吴承泽

实验内容：

这次实验是实现三次样条插值。通过给定的函数与区间，以大M法作三次样条插值，边界条件均采取自然边界条件，并使用python作出五组三次样条插值函数与原函数。

实验源码：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <windows.h>

#define N 50

#define M0 0
#define Mn 0

#define m0 0
#define mn 0

double A[N + 2][N + 2];
double x[N + 3];
double b[N + 2];

double Sample_Point[N + 1][2];      //Sample_Point[][0]存放x, Sample_Point[][1]存放y
double Range[2][2] = {-2, 2, -2, 4};

double lamda[N + 1];
double miu[N + 1];
double d[N + 1];
double h[N + 1];
FILE *fp;

int cnt = 0;

double func(double x, int k)
{
    switch(k)
    {
        case 0: return (x / (x*x + x + 1));
        case 1: return (x + sin(2 * x))/(1 + exp(-1*x));
        default: exit(1);
    }
}

void Order_Initialization(int k)
{
```

```
    for(int i = 0;i < N + 1; i++)
    {
        Sample_Point[i][0] = (Range[k][1] - Range[k][0])/ N * i + Range[k][0];
    }
    for(int i = 0; i < N + 1; i++)
    {
        Sample_Point[i][1] = func(Sample_Point[i][0], k);
    }
}

int mycmp(const void* p1, const void* p2)
{
    double temp = ((double *)p1)[0] - ((double *)p2)[0];
    if(temp > 0) return 1;
    else if (temp < 0) return -1;
    else return 0;
}

void Ramd_Initialization(int k)
{
    Sleep(1000);
    srand((unsigned int)time(NULL));
    for(int i = 1; i < N; i++)
    {
        Sample_Point[i][0] = (rand() / 32767.0)*(Range[k][1] - Range[k][0]) +
Range[k][0];
    }
    Sample_Point[0][0] = Range[k][0];
    Sample_Point[N][0] = Range[k][1];
    qsort(Sample_Point, N + 1 , sizeof(Sample_Point[0]), mycmp);
    for(int i = 0; i < N + 1; i++)
    {
        Sample_Point[i][1] = func(Sample_Point[i][0], k);
    }
    int c = 1;
}

double Func_Compute(double Solution[][4], int i)
{
    return Solution[i][0] + Solution[i][1] * Sample_Point[i][0] + Solution[i][2]
* pow(Sample_Point[i][0], 2) + Solution[i][3] * pow(Sample_Point[i][0], 3);
}

void Form_Array(int k, int mode)
{
    void Compute_Linear_Equations(int num, int mode);

    int i;
    double Solution[N][4];
    for(i = 0; i < N; i++)
    {
        h[i] = Sample_Point[i + 1][0] - Sample_Point[i][0];
    }
    for(i = 1; i < N; i++)      //lamda[i]从1开始
    {
        lamda[i] = h[i] / (h[i] + h[i - 1]);
        miu[i] = 1 - lamda[i];
    }
```

```
    for(i = 1; i < N; i++)
    {
        d[i] = 6.0 / (h[i] + h[i - 1]) * ((Sample_Point[i + 1][1] -
Sample_Point[i][1]) / h[i] - (Sample_Point[i][1] - Sample_Point[i - 1][1]) / h[i
- 1]);
    }


    if(mode == 0)
    {
        for(i = 1; i < N - 1; i++)
        {
            A[i][i + 1] = lamda[i];
            A[i][i] = 2;
            A[i + 1][i] = miu[i + 1];
        }
        A[N - 1][N - 1] = 2;

        b[1] = d[1] - miu[1] * M0;
        b[N - 1] = d[N - 1] - lamda[N - 1] * Mn;
        for(i = 2; i <  N - 1; i++)
        {
            b[i] = d[i];
        }
        Compute_Linear_Equations(N - 1, mode);

        x[0] = M0;
        x[N] = Mn;
    }
    else
    {
        for(i = 1; i < N + 1; i++)
        {
            A[i][i + 1] = (i == 1? 1 : lamda[i - 1]);
            A[i][i] = 2;
            A[i + 1][i] = (i == N? 1 : miu[i]);
        }
        A[N + 1][N + 1] = 2;

        b[1] = 6.0 / h[0] *((Sample_Point[0][1] - Sample_Point[1]
[1])/(Sample_Point[0][0] - Sample_Point[1][0]) - m0);
        b[N + 1] = 6.0 / h[N - 1] *(mn - (Sample_Point[N - 1][1] -
Sample_Point[N][1])/(Sample_Point[N - 1][0] - Sample_Point[N][0]));

        for(i = 2; i < N + 1; i++)
        {
            b[i] = d[i - 1];
        }

        Compute_Linear_Equations(N - 1, mode);
        int c = 1;
    }

    for(i = 0; i < N; i++)
    {
        Solution[i][3] = (x[i + 1] - x[i]) / (6 * h[i]);
        Solution[i][2] = (Sample_Point[i + 1][0] * x[i] - Sample_Point[i][0]
*x[i + 1]) / (2 * h[i]);
```

```
        Solution[i][1] = (Sample_Point[i + 1][1] - Sample_Point[i][1]) / h[i] +
(x[i] * h[i])/6 - (x[i + 1]*h[i])/6 - (x[i] * Sample_Point[i + 1][0] *
Sample_Point[i + 1][0]) / (2 * h[i])+ (x[i + 1] * Sample_Point[i][0] *
Sample_Point[i][0]) / (2 * h[i]);
        Solution[i][0] = -(Sample_Point[i][0] * Sample_Point[i + 1][1])/h[i] +
(Sample_Point[i + 1][0] * Sample_Point[i][1])/h[i] + (x[i] * pow(Sample_Point[i
+ 1][0],3))/(6*h[i]) - (x[i + 1] * pow(Sample_Point[i][0],3))/(6*h[i]) -
(x[i]*h[i]*Sample_Point[i + 1][0])/6 + (x[i + 1]*h[i]*Sample_Point[i][0])/6;
    }

    //生成python代码部分
    if(cnt == 0)
    {
        fprintf(fp, "x = np.linspace(%lf, %lf, 100)\n",Range[k][0] ,Range[k]
[1]);
        if(k == 0)
            fprintf(fp,"y = x/(x*x+x+1)\n");
        else
            fprintf(fp, "y = (x + np.sin(2*x))/(1+np.exp(-1*x))\n");
        fprintf(fp,"plt.plot(x,y)\n\n");
    }

    fprintf(fp, "x%d = np.linspace(%lf, %lf, 100)\n",cnt, Range[k][0] ,Range[k]
[1]);

    for(i = 0; i < N; i++)
    {
        if(i!= N - 1)
            fprintf(fp, "interval%d%d = [1 if (i>=%lf and i<%lf) else 0 for i in
x%d]\n",cnt,i,Sample_Point[i][0],Sample_Point[i + 1][0], cnt);
        else
            fprintf(fp, "interval%d%d = [1 if (i>=%lf and i<=%lf) else 0 for i
in x%d]\n",cnt,i,Sample_Point[i][0],Sample_Point[i + 1][0], cnt);
    }
    fprintf(fp, "y%d = ",cnt);

    for(i = 0; i < N; i++)
    {
        if(i)
            fprintf(fp,"+");
        fprintf(fp, "
(%lf*x%d**3+%lf*x%d**2+%lf*x%d+%lf)*interval%d%d",Solution[i][3], cnt,
Solution[i][2], cnt, Solution[i][1], cnt, Solution[i][0],cnt, i);
    }
    if(cnt == 0)
    {
        fprintf(fp, "\nvalues = [");
        for(i = 0; i < N; i++)
        {
            fprintf(fp,"%lf, ",Sample_Point[i][0]);
        }
        fprintf(fp,"%lf",Sample_Point[i][0]);
        fprintf(fp, "]\n");
        fprintf(fp, "squares = [");
        for(i = 0; i < N; i++)
        {
            fprintf(fp,"%lf, ",Func_Compute(Solution, i));
        }
```

```
        fprintf(fp,"%lf",Solution[i - 1][0] + Solution[i - 1][1] *
Sample_Point[i][0] + Solution[i - 1][2] * pow(Sample_Point[i][0], 2) +
Solution[i - 1][3] * pow(Sample_Point[i][0], 3));
        fprintf(fp, "]\n");
        fprintf(fp,"plt.scatter(values,squares,color='r',s=20)\n");
    }
    fprintf(fp,"\n");
    fprintf(fp,"plt.plot(x%d,y%d)\n\n",cnt,cnt);
}

void Compute_Linear_Equations(int num, int mode)
{
    int i;
    double v[N + 2], u[N + 2], y[N + 2];
    v[0] = 0;
    A[1][0] = 0;
    for(i = 1; i < num; i++)
    {
        u[i] = A[i][i] - A[i][i - 1] * v[i - 1];
        v[i] = A[i][i + 1] / u[i];
        y[i] = (b[i] - A[i][i - 1] * y[i - 1]) / u[i];
    }
    for(i = num; i > 0; i--)
        x[i] = y[i] - v[i] * x[i + 1];

    if(mode == 1)
        for(i = 0; i < num; i++)
        {
            x[i] = x[i + 1];
        }
}



int main()
{
    fp = fopen("data.txt","w");
    fclose(fp);

    fp = fopen("data.txt","a");

    Order_Initialization(0);
    Form_Array(0, 0);
    cnt++;

    for(int i = 0; i < 4; i++)
    {
        Ramd_Initialization(0);
        Form_Array(0, 0);
        cnt++;
    }
    fprintf(fp,"plt.show()\n\n");

    cnt = 0;
    Order_Initialization(0);
    Form_Array(0, 1);
    cnt++;
```

```
    for(int i = 0; i < 4; i++)
    {
        Ramd_Initialization(0);
        Form_Array(0, 1);
        cnt++;
    }
    fprintf(fp,"plt.show()\n\n");

    cnt = 0;
    Order_Initialization(1);
    Form_Array(1, 0);
    cnt++;

    for(int i = 0; i < 4; i++)
    {
        Ramd_Initialization(1);
        Form_Array(1, 0);
        cnt++;
    }
    fprintf(fp,"plt.show()\n\n");

    cnt = 0;
    Order_Initialization(1);
    Form_Array(1, 1);
    cnt++;

    for(int i = 0; i < 4; i++)
    {
        Ramd_Initialization(1);
        Form_Array(1, 1);
        cnt++;
    }
    fprintf(fp,"plt.show()\n\n");

    fclose(fp);
    return 0;
}
```

实验结果：

生成的python代码在src中的TXT文件内，长度过长不在此贴出：

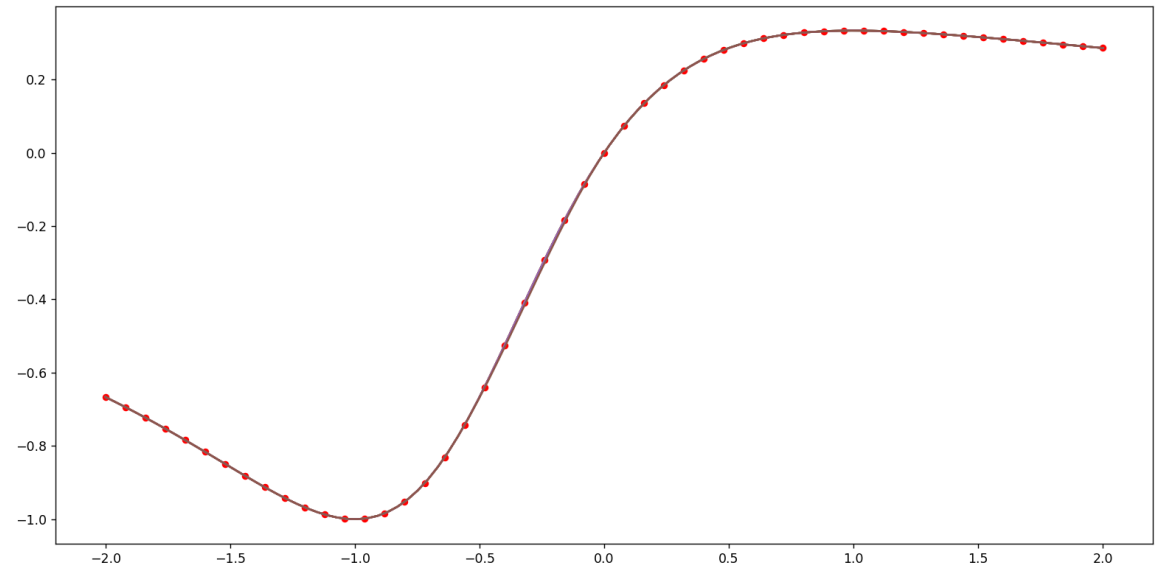
注意：需自行在开头输入以下代码，且需要将生成的python代码分四次粘贴出来

```
import matplotlib.pyplot as plt
import numpy as np
```

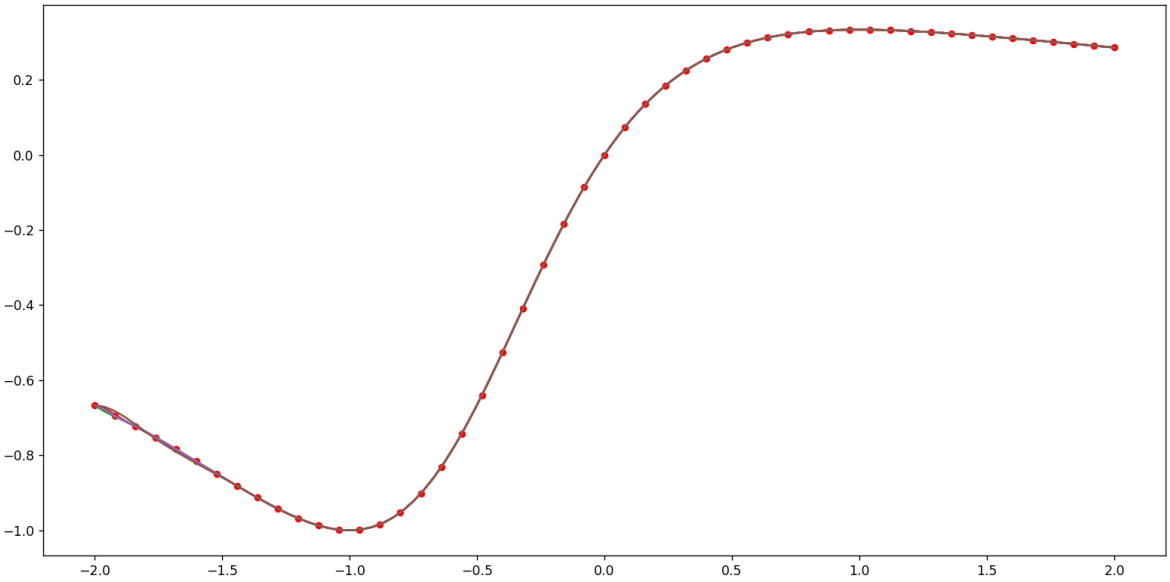
在C程序中生成的三次样条函数插值的系数存储在Solution数组中，插值点存储在Sample_Point中。

作图如下（由于五条曲线与一条原函数高度重合，因此下面四张图拟合出的效果与原函数几乎重叠）

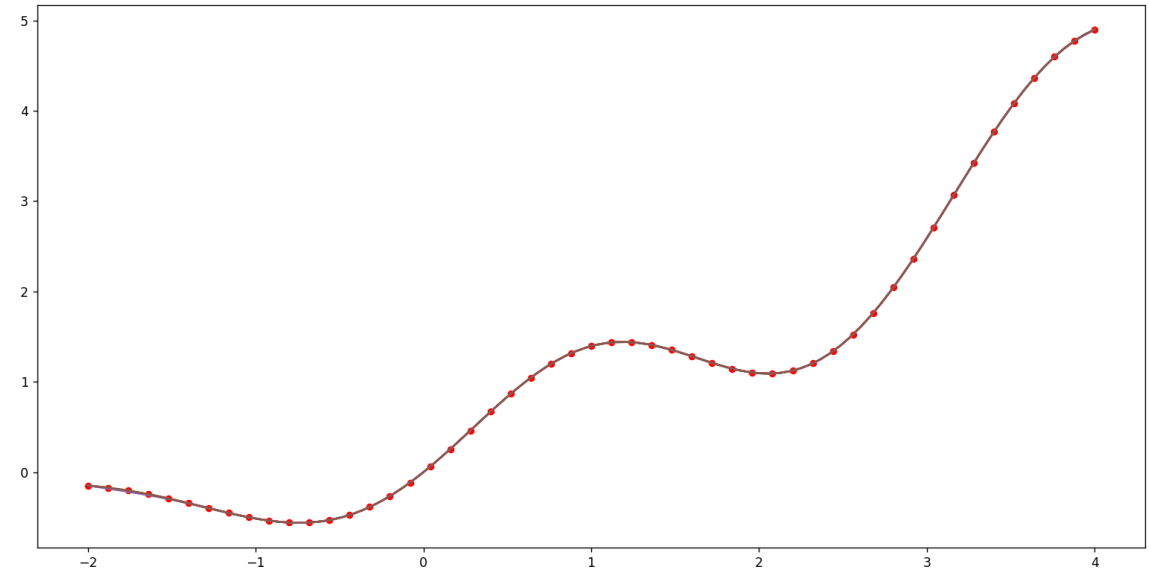
第一个函数的M0Mn均为0的边界条件下生成的图像：



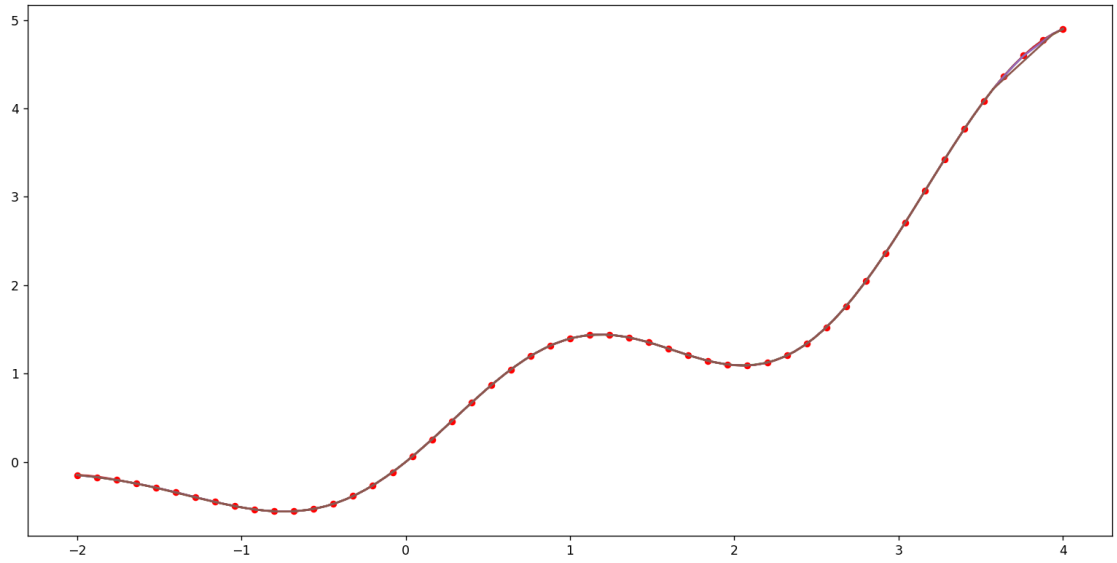
第一个函数的m0mn均为0的边界条件下生成的图像：



第二个函数的M0Mn均为0的边界条件下生成的图像：



第二个函数的m0mn均为0的边界条件下生成的图像：



其中

函数一为f(x) = x / (x^2 + x + 1)

函数二为f(x) = (x+sin(2x))/(1 + e^(-x))

结果分析：

(a)三次样条插值函数的拟合效果很好，在拟合绘制的图中几乎看不出三次样条插值函数与原函数的区别(N = 50)。

(b)对比均匀插值与随机插值，均匀插值的拟合效果比随机插值要更好，因为均匀插值中的区间大小相等，以三次函数拟合曲线的最大偏差的上界会更小。

(c)即使是对于相同的函数，同样是均匀插值，取不同边界条件的拟合效果也不相同。

(d)对于同一函数，边界条件相同，随机插值的效果取决于随机生成点列函数是否均匀，均匀产生的x_i的插值效果比非均匀的效果更好。