数值分析实验

实验一 误差与插值法

2013011393 计 35 冯栩

一、 实验目的

详见实验指导书。

二、 算法描述

使用了不完整的高斯消元法以求解第二题。而第一题则是直接使用循环判断误差是否符合精度要求即可。

三、 程序清单

见 四 3 附件说明。

四、 实验结果

1、

根据实验的要求，编写了一个使用两种方法检测 n 在多少时能够满足精度要求。程序如下（完整程序附在 1_1.cpp 中）：

```
#include <iostream>
#include <cstdlib>
#include <cmath>

using namespace std;

int main()
{
    int n=1;
    float err_1 = 0, err_2 = 0;
    float est = 0;
    do
    {
```

```
        if( n % 2 == 1)
        {
            est += 1.0/n;
        }
        else
        {
            est -= 1.0/n;
        }
        err_2 = abs(0.693147190546 - est);
        n++;
    }
    while(err_2 >= 0.5e-4);
    cout << est << ' ' << n << endl;
    n = 1;
    do
    {
        err_1 = 1.0/(n+1);
        n++;
    }
    while(err_1 >= 0.5e-4);
    cout << n << endl;
}
```

根据结果输出，根据数列展开计算法，n 为 8976。但是根据理论计算，当 n 为 20000 时才能满足精度要求。具体分析如下：

由估计有

$$|x_n - ln2| < \frac{1}{n+1}$$

但是也有

$$|x_n - ln2| < \frac{1}{n+1} - \frac{1}{n+2} + \frac{1}{n+3} < \frac{1}{n+1}$$

由此不断的增添余项，我们不难发现实际的误差是要小于我们估计的值的，

因此不难解释为什么使用数列叠加的 n 到 8976 就满足了精度，但是使用 $\frac{1}{n+1}$就会在 n 为 20000 时才满足精度要求。

2、

（1）按照 Lagrange 插值，直接使用书上的公式进行带入，即

$$L_{10}(x) = \sum_{I=0}^{10} f(x_i)\, l_i(x)$$

$$L_{20}(x) = \sum_{I=0}^{20} f(x_i)\, l_i(x)$$

（2）根据书上的例题以及具体的实现操作，需要通过编程来解出所需要的 M_i。因此选择使用 c++编写高斯消元的办法来求解（具体实现详见 1_2_s.cpp）。程序会将高斯消元求解过程以及最后的所有 M 值输出到屏幕上

当 n=10 时,

i	0	1	2	3	4
M_i	0.012301	-0.044377	0.174856	-0.622153	2.52346
y_i	0.00249377	0.00389105	0.00689655	0.0153846	0.0588235
5	6	7	8	9	10
-4.08526	2.52346	-0.622153	0.174856	-0.044377	0.012301
1	0.0588235	0.0153846	0.00689655	0.00389105	0.00249377

由于数值是对称的我们可以相应的对于 n=20 的表格做出简化，

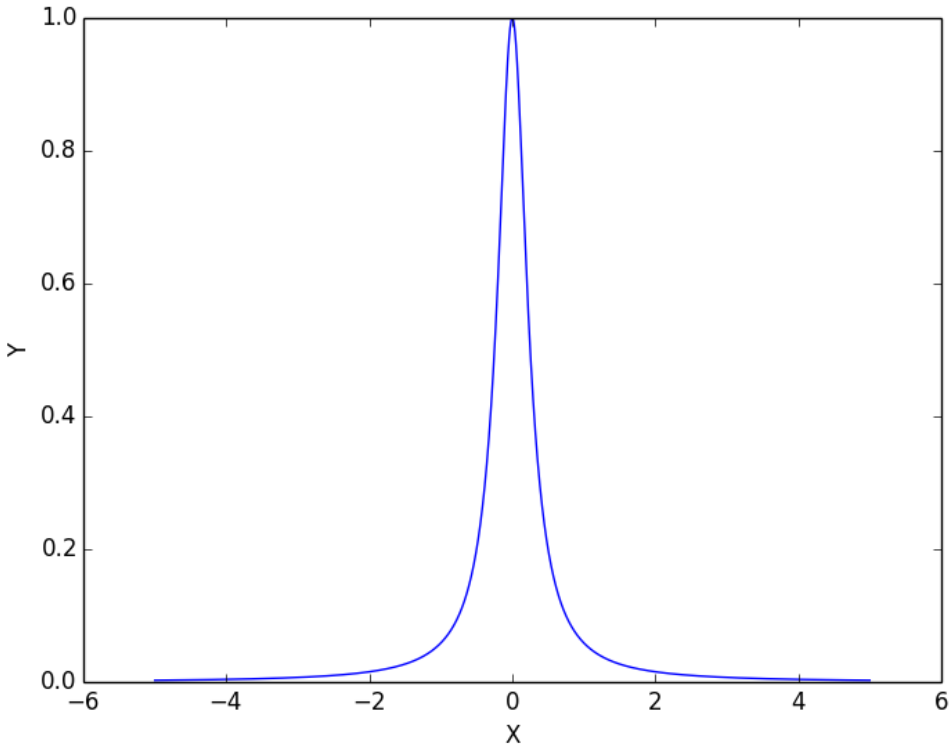
i	0	1	2	3	4
M_i	0.000782114	0.000982653	0.000830557	0.0045982	-0.00397594
y_i	0.0024937	0.0030769	0.0038910	0.0050761	0.0068965
5	6	7	8	9	10
0.0397222	-0.0954123	0.489738	-1.37984	7.65474	-13.4274
0.00990099	0.0153846	0.027027	0.0588235	0.2	1

之后根据三次样条插值的分段函数进行带入相应值即可。

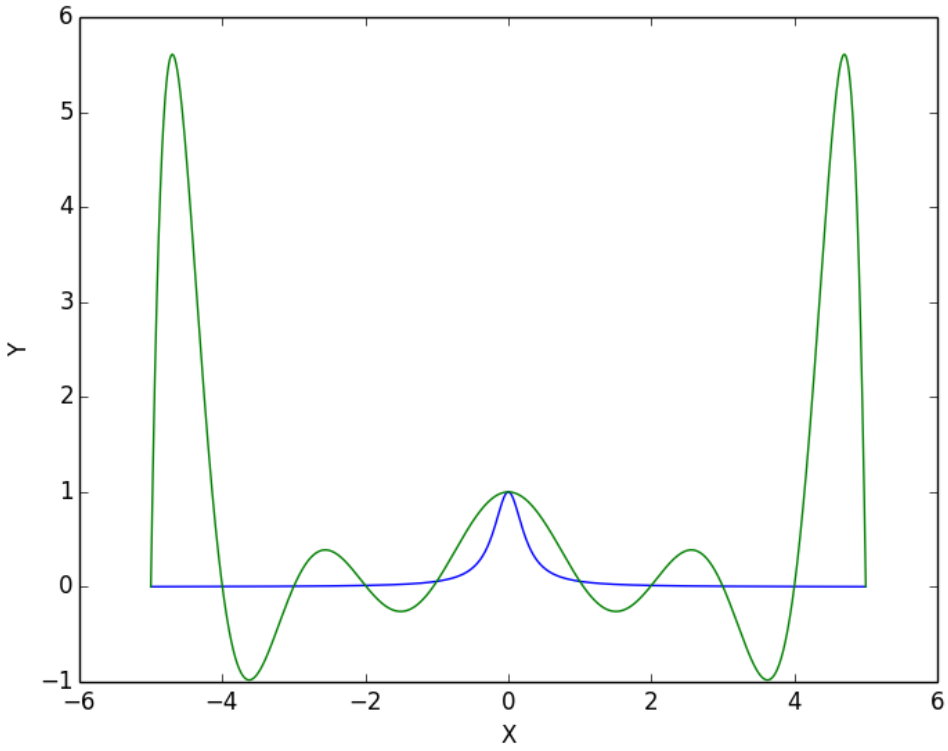
（3）由于需要画图，我是用了 python 的第三方库 matplotlib 用来进行图形

的绘制。在 paint.py 中分别有四个函数的绘制信息，是通过在[-5,5]中取 10000 个点来进行的绘图。

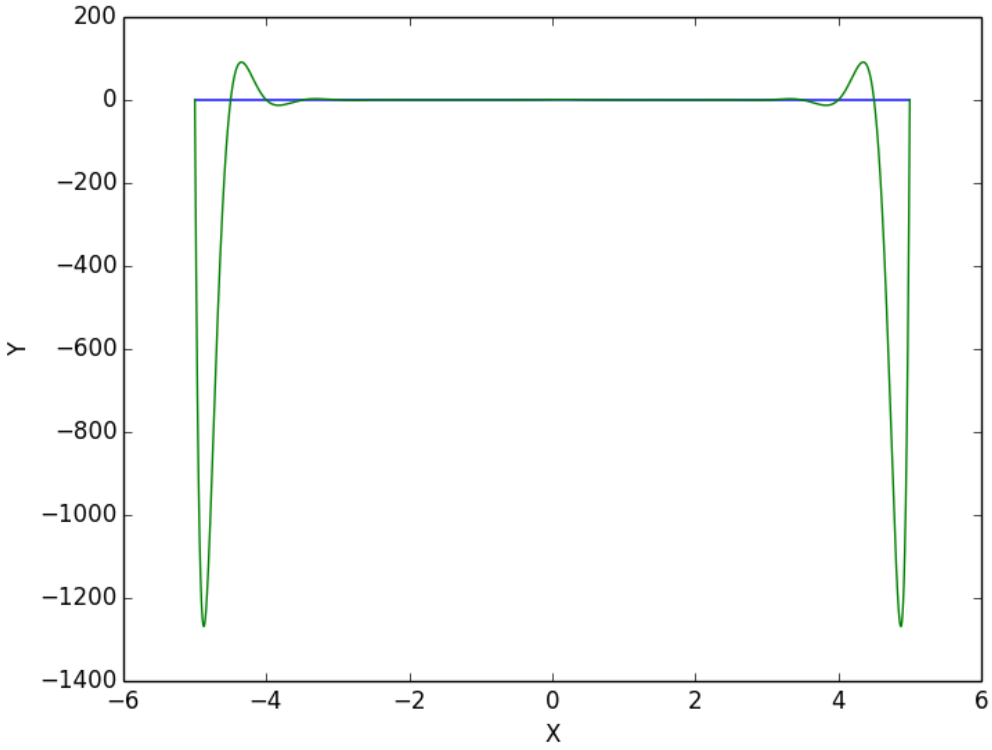
f(x)



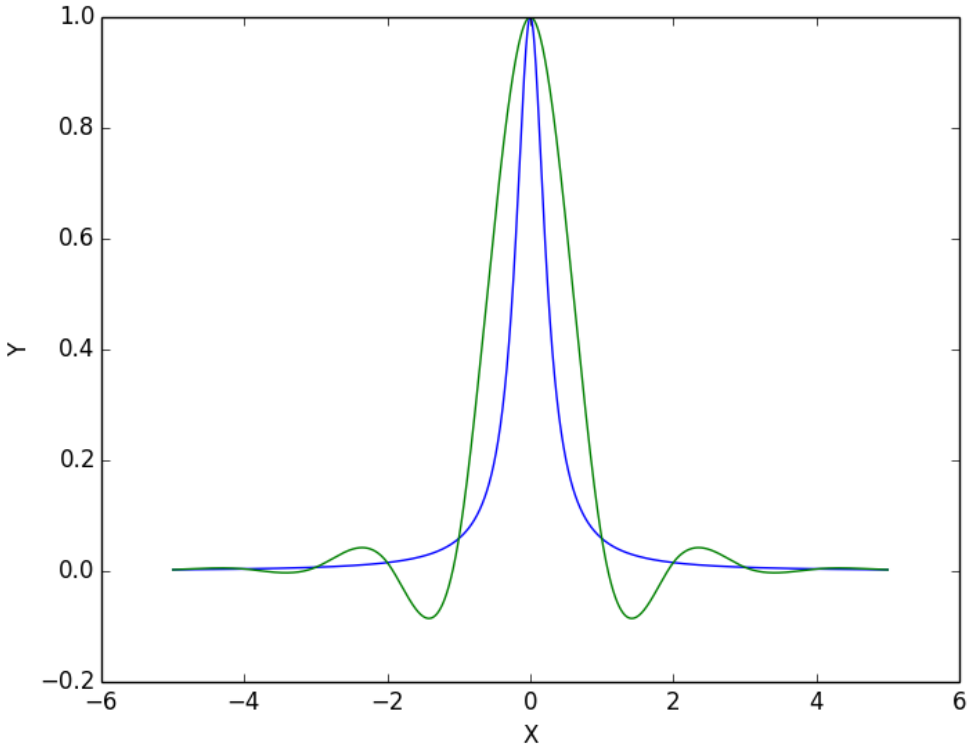
L_{10}


6
5
4
3
2
1
0
−1
Y
−6
−4
−2
0
2
4
6
X

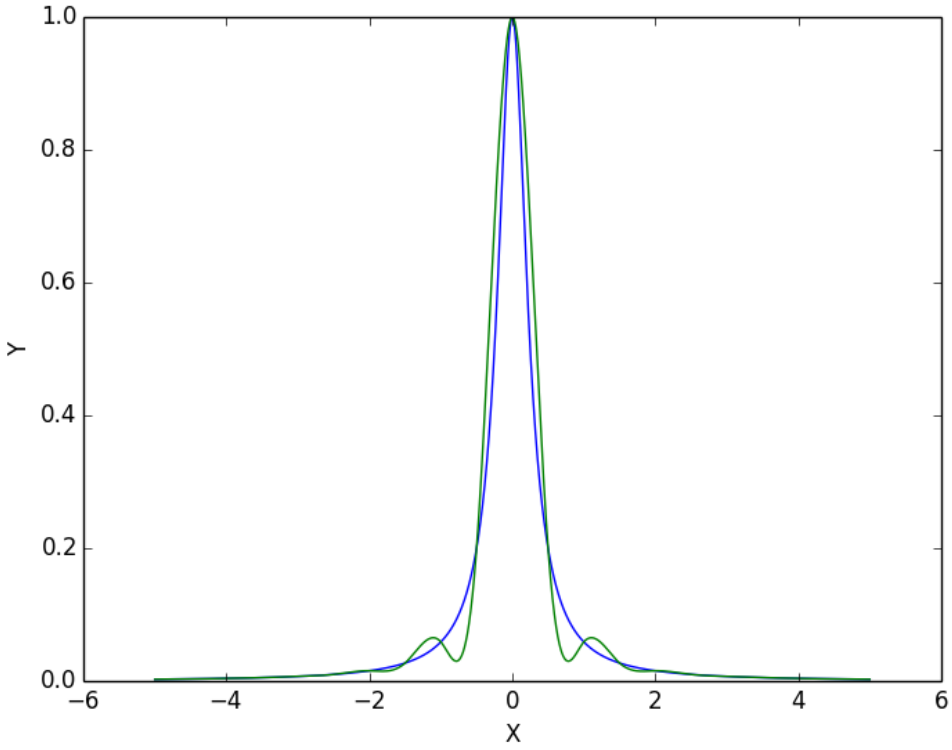
L_{20}


200
0
−200
−400
−600
−800
−1000
−1200
−1400
Y
−6
−4
−2
0
2
4
6
X

S_{10}



S_{20}



从图中我们可以发现，使用 Lagrange 插值法尽管在中间的时候会比较趋近，但是随着节点的增多，即次数的变高，在边缘的地方误差就会变得越来越大，这也就是 Runge 现象：高次插值的病态性质。反观三次样条插值，一是因为其最高次数是三次，因此不会出现较大的病态性；二则是由于三次样条函数是分段函数，因此在每段都有其较好的拟合性质。而且从 n=10 和 20 的比较当中我们不难看出，随着点数的增多，三次样条函数的拟合性质变得越来越好，误差也会变得越来越小。

（4）将 paint.py 进行一定的修改使其输出在 4.8 时各个函数的值即可

$f(x)$	L_{10}	L_{20}	S_{10}	S_{20}
0.0027053	5.1513	−1080.7	0.0036028	0.0027008

在 4.8 出的函数值也向我们说明了之前的猜测，即使用 Lagrange 插值法点数越多病态性越明显。而三次样条插值则是随着点数增多，误差越小。这其实是由于高次函数例如 L_{10}，尽管在-5 和-4 两点上都是原函数值，但是由于高次函数零点的性质，在这两点之间的极值就会发生一个巨变（因为最高次为 10），因此出现病态性质。但是三次样条函数最高次数只有三次，而且还是分段拟合，因此效果就会号上很多。

3、附件说明

1_1.cpp

直接编译运行即可，会输出两种误差估计方式下 n 在到达多少时符合精度要求

1_2_s.cpp

编译运行后输入 n 的值（10、20），之后会输出高斯消元过程以及最后的解 M_i，并且输出各点 $f(x_i)$值。

paint.py

使用 python 绘图，需要对需要的块进行反注释来进行运行。

五、 实验心得

第一题还算是简单，但是到了第二题就很费时间，主要并不是高斯消元或者是其他地方，而是将那些计算的系数导入到 python 中进行画图特别费时间。但是这次实验确实也收获了很多，比如十分直观的了解了龙格现象，以及三次样条

插值优良的性质等。