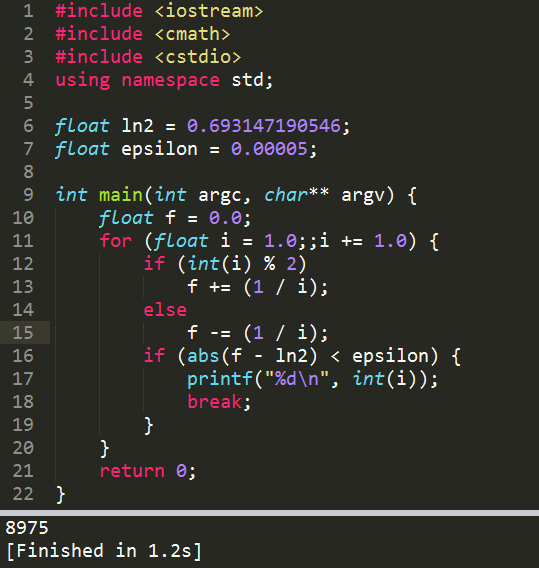
一、

实验代码与运行结果如图：代码lab1-1.cpp



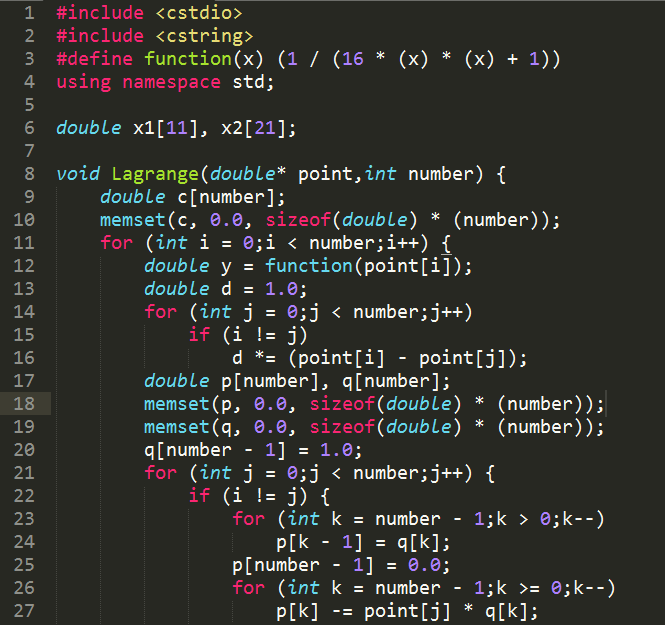
由估计式1 / (n + 1) < 0.5 x 10 -4解得n=19999，比程序运行结果大一倍有余。

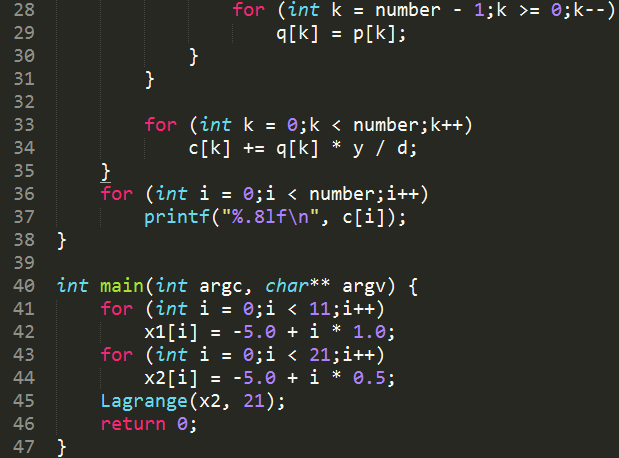
推测可能的几点原因：

1. 误差的估计式仅给出了误差的上限，实际的误差值与该理论值存在一定的差异；
2. 单精度浮点数在运算过程中存在一定的舍入误差，且当n值较大时，1/n值愈加趋近于0，单精度浮点数能表示的精度有限，且随着累加结果的增大，会出现大数吞噬小数的情况，这些因素都使得误差会累计。

二、

计算拉格朗日插值多项式系数的程序如下：代码lab1-2.cpp,对不同的n需要修改源代码中调用Lagrange（）函数的参数。



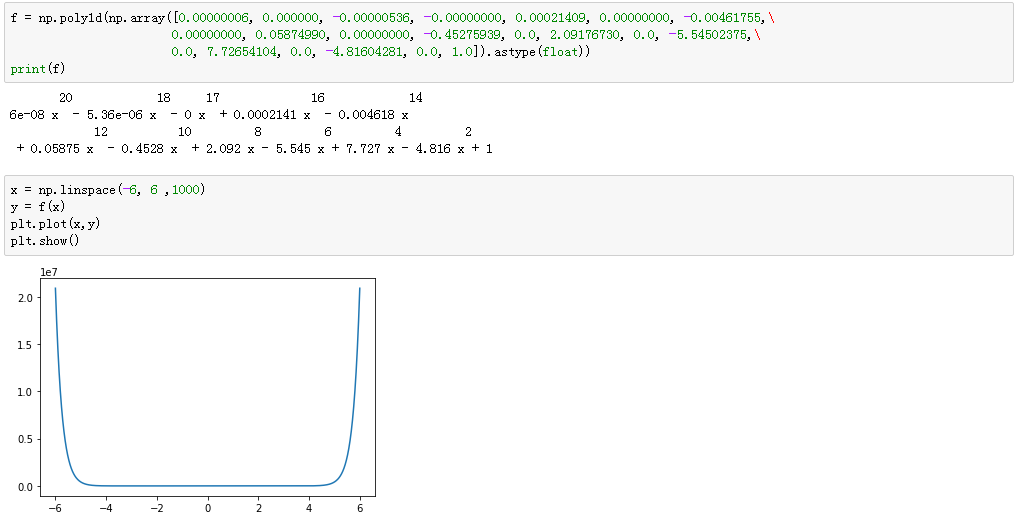


利用求出的系数绘制函数图像如下：

绘图工具：jupyter notebook + python3.5 + matplotlib2.2，下同。

拉格朗日多项式的系数可见如下python代码：文件见lab1.ipynb

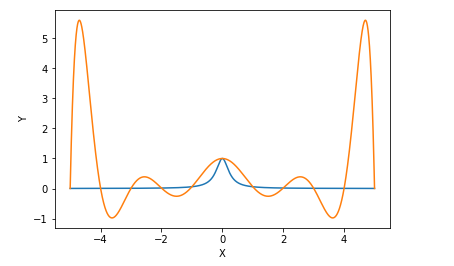


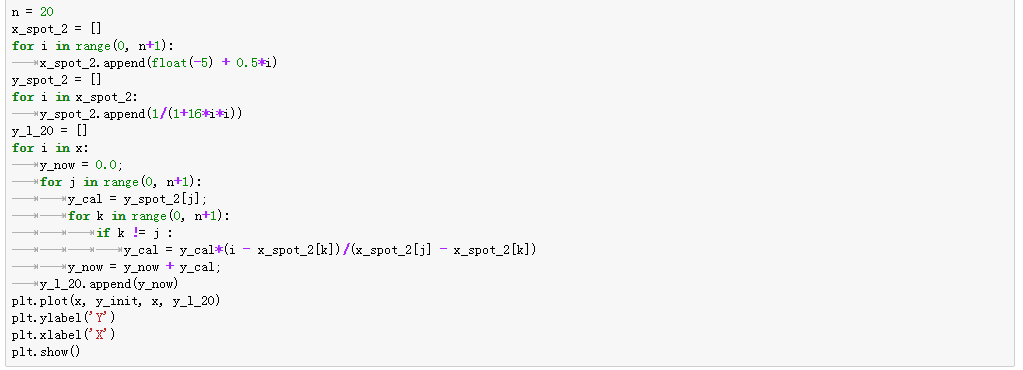


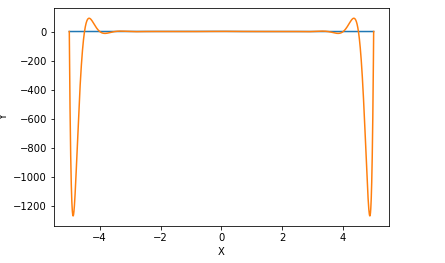
奇数次项系数为0是因为原函数是偶函数所致。

不采用计算系数的方法而直接绘制函数图像的代码且显示如下：







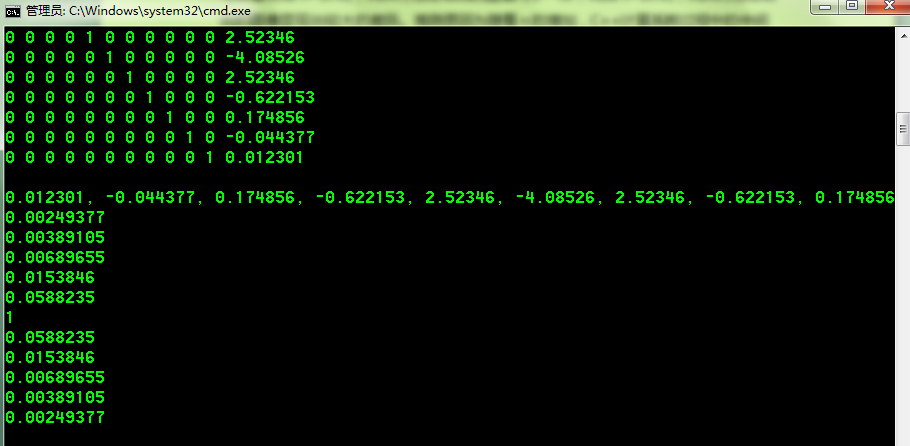


可以看出当n=10时，两种方法绘制出的图像几乎一样，而当n=20时，两种方法绘制出的图像显现出较大的差异。推测原因为随着n的增加，C++计算系数过程中的中间量越来越小，double类型的舍入误差不断累计，导致误差较大，甚至在函数的最高次项系数的正负上也出现了差异。

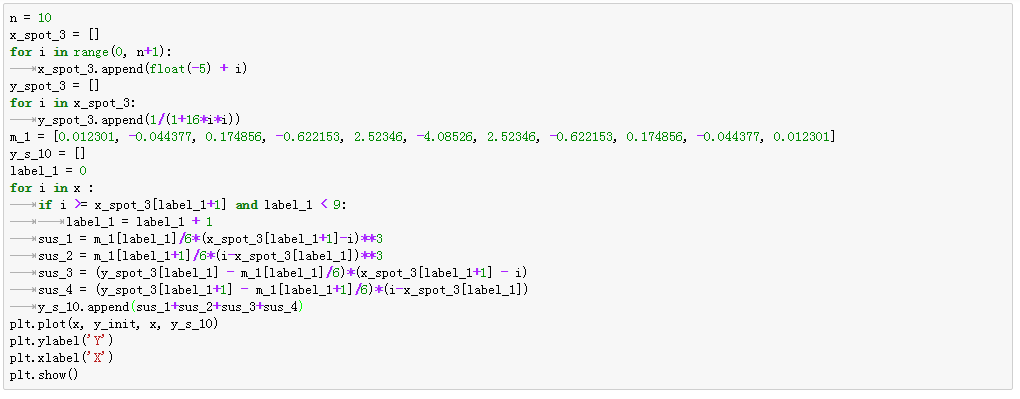
样条插值利用二阶导数为参数，然后采用高斯消元法求解出参数即可。

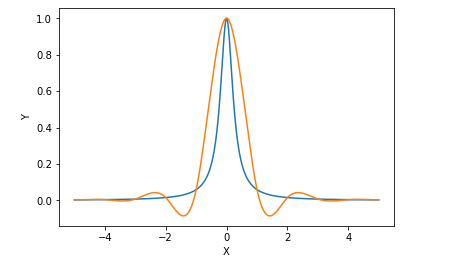
代码lab1-3.cpp

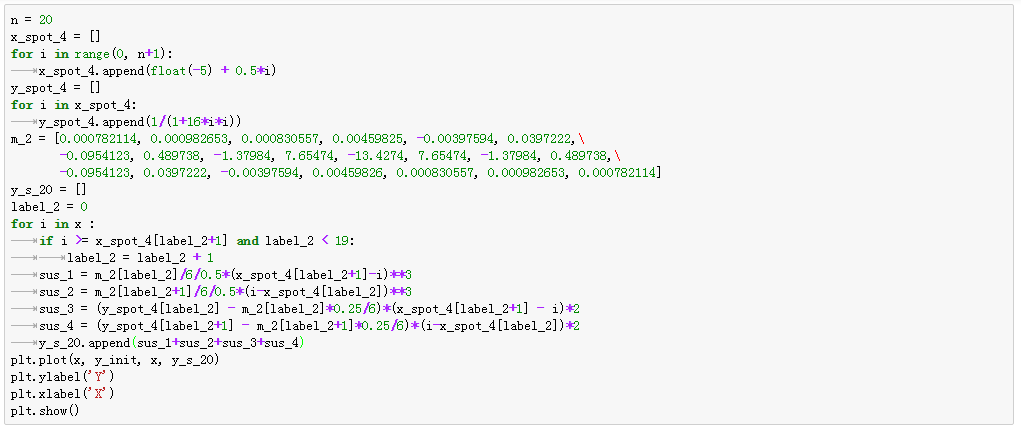
输入插值段数（10或20），将输出高斯消元法的过程，以及Mi和yi的数值。

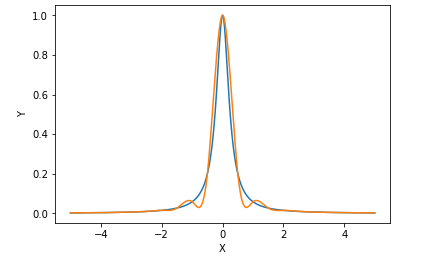


绘制图像如下：









在x=4.8时，L10 = 5.1513 L20 = -1080.7 S10 = 0.0036028 S20 = 0.0027008

F = 0.0027053

观察可得，Lagrange插值在边缘处出现了病态性质，且随着插值点数的增多，插值多项式的次数增大，这种病态现象越来越明显。而样条插值则没有出现病态现象，且随着插值点数的增多，误差越来越小。且在计算Lagrange插值多项式的系数时，随着次数的升高，多项式的系数将越来越小，计算过程中累计的误差也会越来严重。