Numerical Analysis Programming Assignment #2 Lagrange 插值

陈宇轩 PB16060738

问题描述

对函数 $f(x)=\frac{1}{4+x+x^2}, x\in[-5,5]$，构造其N次Lagrange插值函数，取

$$\max_{-5\le x\le 5}\left\|f(x)-p(x)\right\|\approx\max_{i}\left|f(y_i)-p(y_i)\right|, y_i=\frac{i}{50}-5, i=0,\cdots 500$$

为近似误差。其中，插值节点（设有 N+1个）取为：

$$(1)\ x_i=-5+\frac{10}{N}i, i=0,1,\cdots N$$

$$(2)\ x_i=-5\cos\left(\frac{2i+1}{2N+2}\pi\right), i=0,1,\cdots N$$

对N= 4，8，16 比较以上两组节点的插值结果。

计算结果

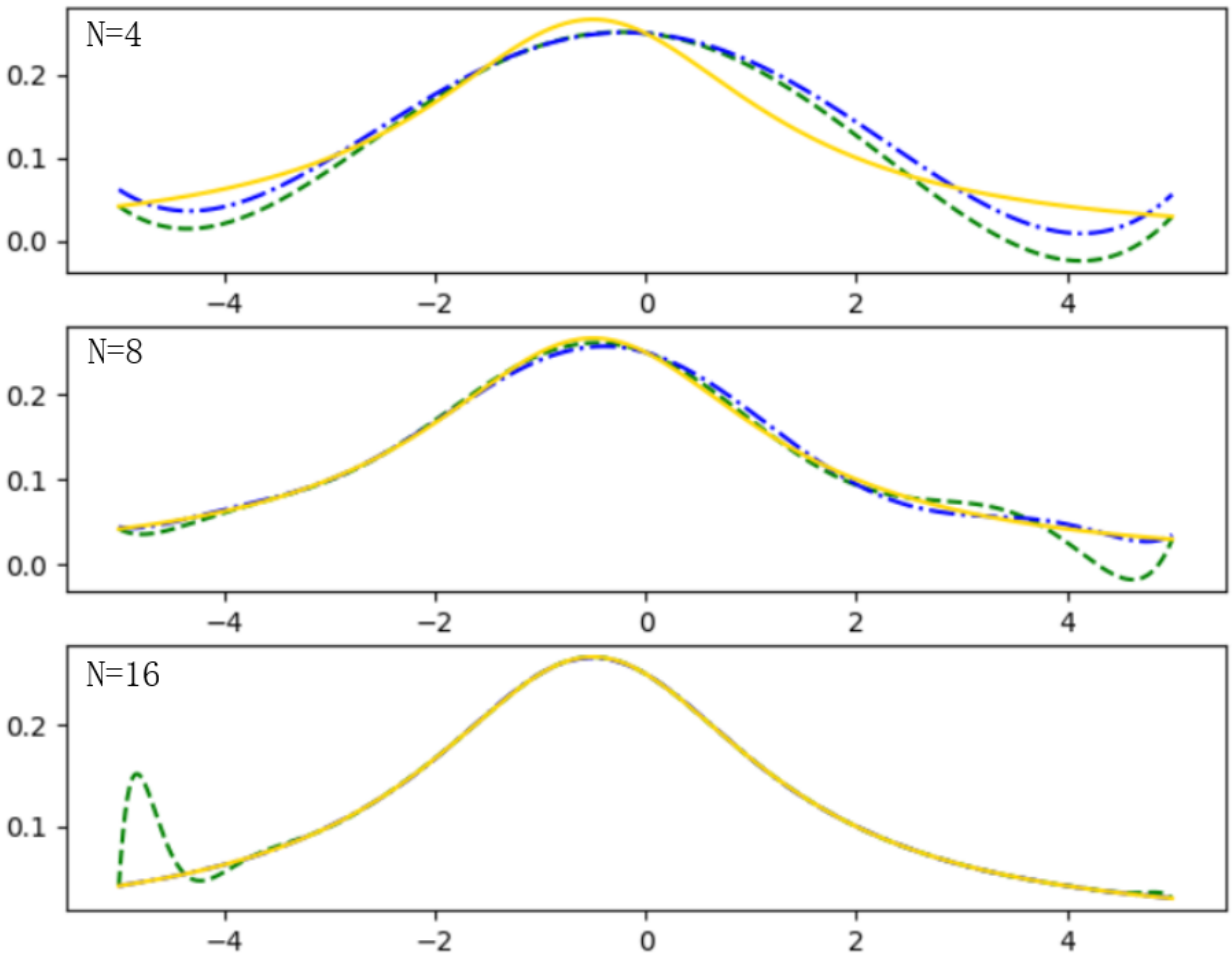
以下表格中数据为对应的N次Lagrange插值函数与原函数f(x)的近似误差。

	N=4	N=8	N=16
第一组节点	6.475332068311e-02	5.156056628633e-02	1.071904407255e-01
第二组节点	5.437602650073e-02	1.426092606546e-02	8.367272096925e-04

作图结果

以下图片分别为N=4,8,16时函数f(x)与其插值函数在[-5,5]上的图像。

其中黄色实线为函数f(x)，绿色短线为第一组插值节点即$x_i=-5+\frac{10}{N}i$ 的插值函数，蓝色短实相间线为第二组插值节点即$x_i=-5\cos(\frac{2i+1}{2N+2}\pi)$ 的插值函数。



结果分析

通过比较近似误差值可以看出：

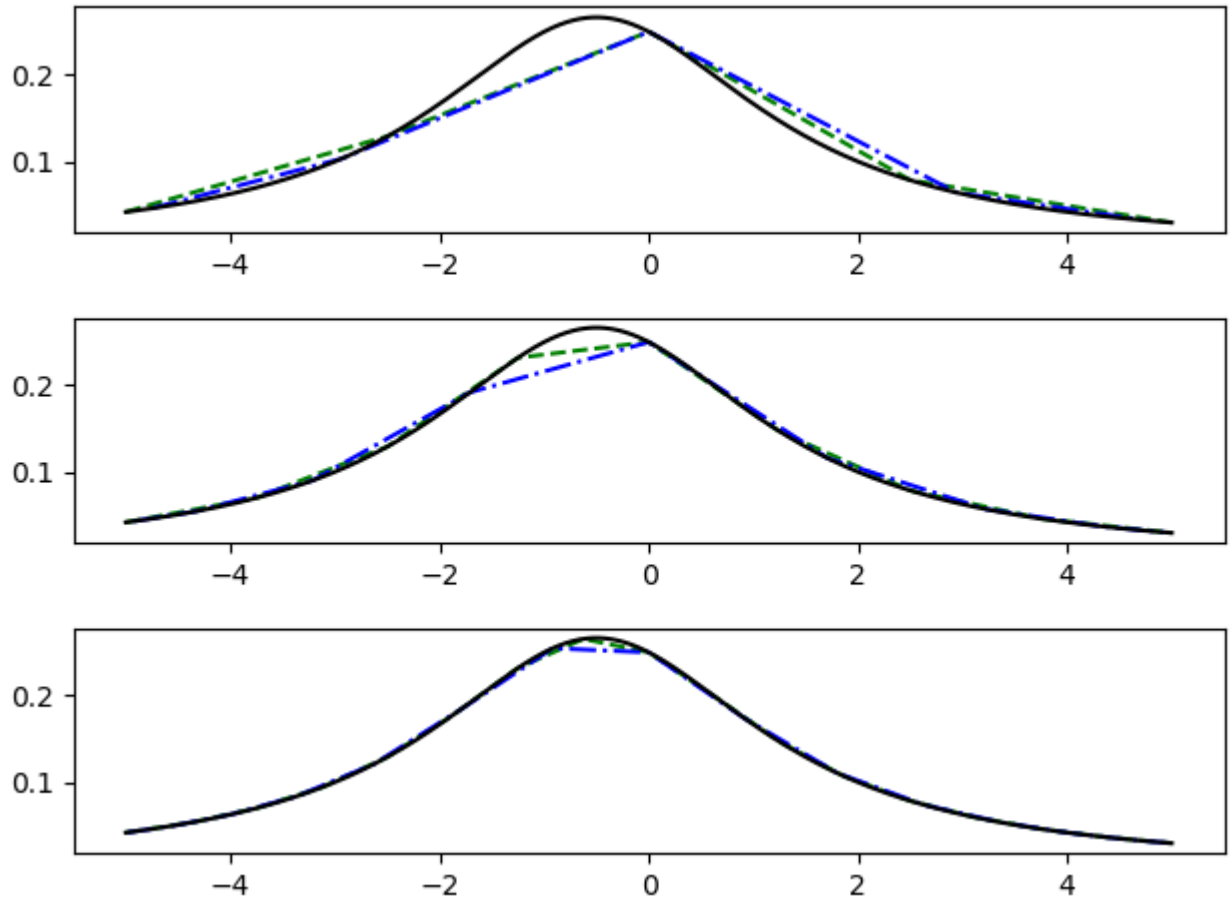
1. N取4，8，16时，第二组插值节点所构造的插值函数的近似误差均比第一组节点的结果好。
2. 随着插值节点个数N的数量的增加，近似误差均越来越小。

通过比较图像可以看出：

1. N=4时，除了峰值部分较为接近，其余部分偏差都很大。 N=8时，插值函数的拟合效果已经很接近函数f(x)。N=16时，第二组节点构造的插值函数已基本与f(x)贴合。
2. 在N=8和N=16时，第一组节点构造的误差函数均在定义域两端出现了较大的误差。可见对于第一组节点这样的取法，步长越小，造成的局部误差在一定范围内可能会更大。

总结

本次实验我选择了python中的matplotlib和numpy库来进行计算和画图。实验过程中曾遇到一个问题，发现图像显示的结果与计算出的偏差值不同。图像如下：



其中很明显N=8时，第一组插值点的偏差反而比第二组的小，这与实际相违背。后来意识到，画图时也只用了N=4,N=8,N=16的x坐标集合，显然不应该用如此少的点作图。于是又改为与f(x)画图方法一样，在[-5,5]上均匀采样了1000个点，重新作图得到了正确的结果。

本次实验，使我对拉格朗日插值法有了更深的理解。

代码实现(Python)

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


def f(x):
    return 1 / (4 + x + x ** 2)


def xset1(n):
    xlist1 = []
    for i in range(n + 1):
        xlist1.append(-5 + 10 / n * i)
    return xlist1
```

```python
def xset2(n):
    xlist2 = []
    for i in range(n + 1):
        xlist2.append(-5 * math.cos((2 * i + 1) / (2 * n + 2) * math.pi))
    return xlist2


def interpolate(x, xlist):
    llist = []
    n = len(xlist)
    result = 0
    for i in range(n):
        l_i = 1
        xi = xlist[i]
        for j in range(n):
            xj = xlist[j]
            if i == j:
                continue
            else:
                l_i = l_i * (x - xj) / (xi - xj)
        llist.append(l_i)
    for i in range(n):
        result = result + llist[i] * f(xlist[i])
    return result


def deviation():
    print("第一组节点，误差为")
    for j in range(2, 5):
        print("n=%d , " % 2 ** j, end='')
        xlist = xset1(2 ** j)
        Max = 0
        for i in range(501):
            yi = i / 50 - 5
            di = math.fabs(f(yi) - interpolate(yi, xlist))
            Max = max(Max, di)
        print("%.12e\n" % Max)

    print("第二组节点，误差为")
    for j in range(2, 5):
        print("n=%d , " % 2 ** j, end='')
        xlist = xset2(2 ** j)
        Max = 0
        for i in range(501):
            yi = i / 50 - 5
            di = math.fabs(f(yi) - interpolate(yi, xlist))
            Max = max(Max, di)
        print("%.12e\n" % Max)


def SelPltPos(xlist):
    x = np.linspace(-5, 5, 1000)
    if len(xlist) == 5:
```

```
        plt.subplot(311)
    elif len(xlist) == 9:
        plt.subplot(312)
    else:
        plt.subplot(313)
    return x


def plot_lagrange():
    x = np.linspace(-5, 5, 1000)
    plt.subplot(311)
    plt.plot(x, f(x), color='gold')
    plt.subplot(312)
    plt.plot(x, f(x), color='gold')
    plt.subplot(313)
    plt.plot(x, f(x), color='gold')


def p1_plot(xlist):
    x = SelPltPos(xlist)
    plt.plot(x, interpolate(x, xlist), color='green', linestyle='--')


def p2_plot(xlist):
    x = SelPltPos(xlist)
    plt.plot(x, interpolate(x, xlist), color='blue', linestyle='-.')


def main():
    p1_plot(xset1(4))
    p1_plot(xset1(8))
    p1_plot(xset1(16))
    p2_plot(xset2(4))
    p2_plot(xset2(8))
    p2_plot(xset2(16))
    plot_lagrange()
    deviation()
    plt.show()


if __name__ == '__main__':
    main()
```