

《数值分析》上机综合大作业成绩

上机实验报告书

**2022 / 2023 学年 第 2 学期**

**课程名称：\_ 数值分析 \_\_\_\_\_\_**

**专业班级：\_ 计算机2103班\_\_\_\_\_\_\_**

**学 号：\_\_\_\_\_\_\_ 210405317\_\_\_ \_\_\_\_\_**

**姓 名：\_\_\_\_\_\_\_\_ 吴奕民\_ \_\_\_\_\_\_\_\_\_**

**指导教师：\_\_\_\_\_\_\_\_\_ 李威\_\_ \_\_\_\_\_\_\_\_\_**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 评分项目 | | 分值比例 | 实际得分 |
| 1 | 出勤情况 | | 10% |  |
| 2 | 技术选择合理性 | | 10% |  |
| 3 | 编程质量和答辩 | 设计方案合理、正确 | 10% |  |
| 对设计过程进行阐述 | 10% |  |
| 程序结果演示 | 10% |  |
| 回答问题 | 10% |  |
| 4 | 设计报告 | 文字描述和说明 | 10% |  |
| 图表清楚正确 | 10% |  |
| 成果体现 | 10% |  |
| 撰写规范性 | 10% |  |
| 总得分/成绩等级 | | | |  |
| 备注： | | | | |

**题目1 插值函数运算**

**1．1 题目的主要研究内容及预期达到的目标**

（1）主要研究内容：设 IMG_256 x属于[-5,5]，在[-5,5]内取n+1个等距节点 IMG_257 构造当n=2,4,6,8,10时的拉格朗日插值多项式 IMG_258 ,并在同一张图上画出f(x)和所有 IMG_258 的图形；

（2）预期目标：通过编程求出各个拉格朗日插值并画出原函数及各个插值函数的图象。

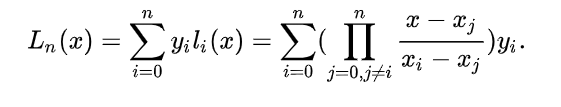
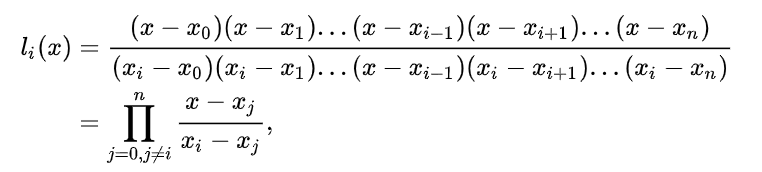
**1．2 题目研究的工作基础或实验条件**

（1）硬件环境：笔记本 Lenovo XiaoXin Pro 13；

（2）软件环境：Window 10 专业版、jupyter-lab、Python3。

**1．3 设计思想**

设计思想为拉格朗日插值法，过程如下：



其中 到 是处于插值区间的各个递增的插值节点，到 是各个插值节点所对应的函数值。

**1．4 流程图**

主程序的流程图如图1所示，插值点生成算法如图2所示，拉格朗日插值算法如图3所示。

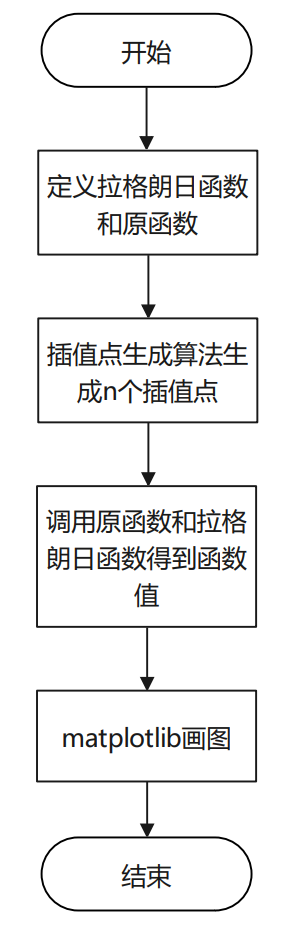


图1 主程序流程图

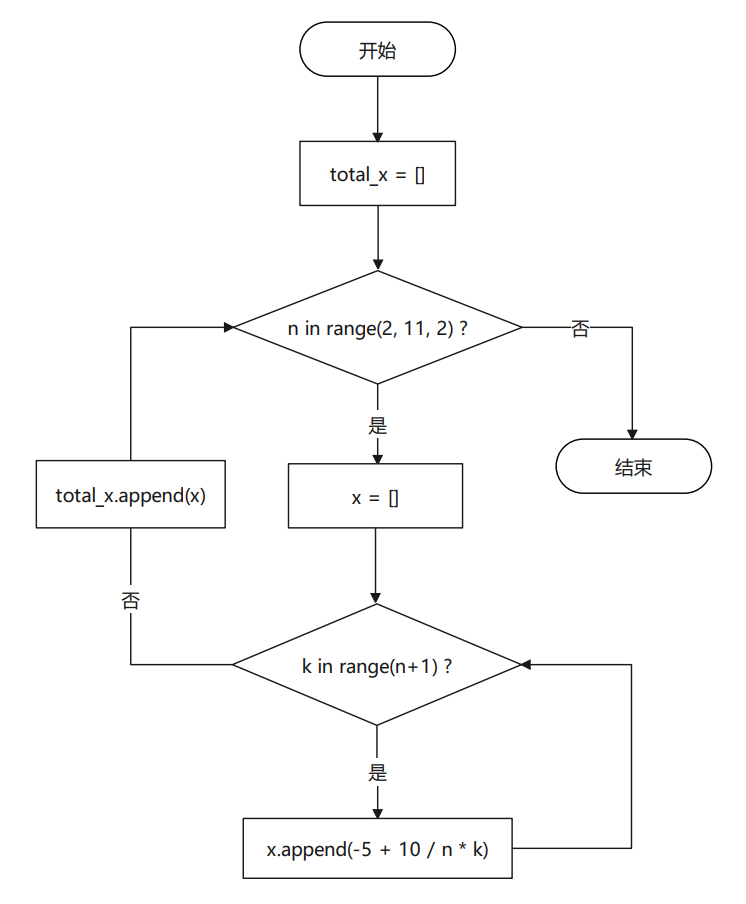


图2 插值点生成算法流程图

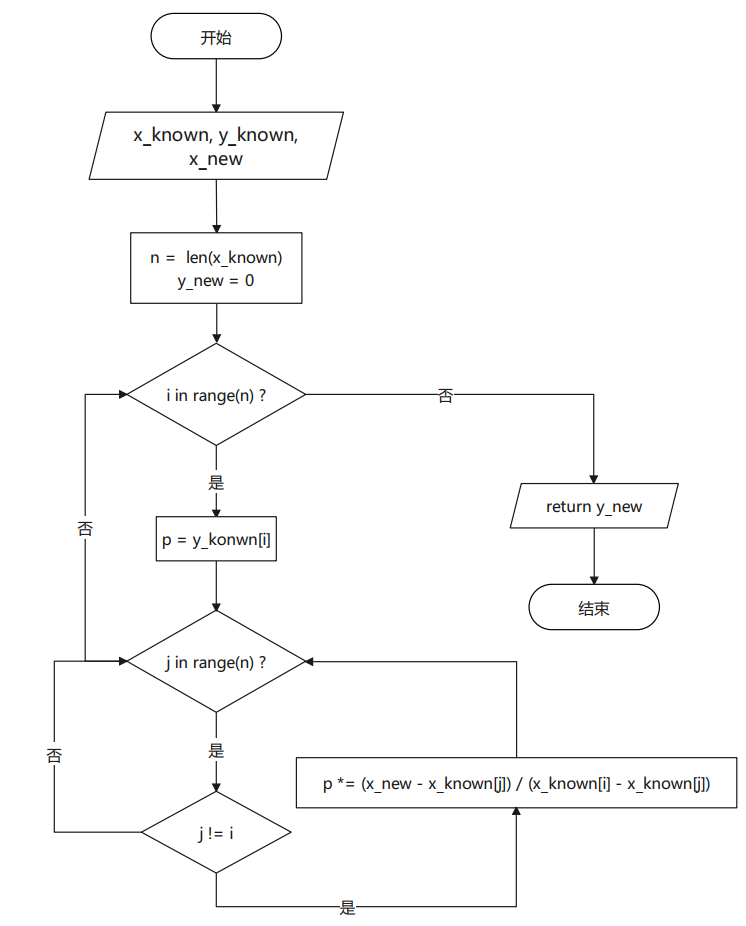


图3 拉格朗日插值算法流程图

**1．5 主要程序代码**

1. import numpy as np
2. import matplotlib.pyplot as plt
3. *# 拉格朗日插值多项式*
4. def lagrange(x\_known, y\_known, x\_new):
5. n = len(x\_known)
6. y\_new = 0
7. for i in range(n):
8. p = y\_known[i]
9. for j in range(n):
10. if j != i:
11. p \*= (x\_new - x\_known[j]) / (x\_known[i] - x\_known[j])
12. y\_new += p
13. return y\_new
14. *# 原函数*
15. def fun(x):
16. return 1 / (1 + x \* x)
17. *# 获取 n=2, 4, 6...插值点*
18. total\_x = []
19. for n in range(2, 11, 2):
20. x = []
21. for k in range(n + 1):
22. x.append(-5 + 10 / n \* k)
23. total\_x.append(x)
24. *# 生成样本点*
25. x0 = np.arange(-5, 5.1, 0.1)
26. *# 生成figure和ax实例*
27. fig, ax = plt.subplots(figsize=(8, 6), layout='constrained')
28. *# 画主函数图像*
29. ax.plot(x0, fun(x0), label='f(x)')
30. *# 画 n=2, 4, 6...插值曲线*
31. for i in range(len(total\_x)):
32. *# 定义已知的数据点*
33. x\_known = np.array(total\_x[i])
34. y\_known = np.array(fun(np.array(total\_x[i])))
35. *# 绘制插值曲线*
36. ax.plot(x0, lagrange(x\_known, y\_known, x0), label=f'n={(i + 1) \* 2}')
37. *# 设置图的属性*
38. ax.set\_xlabel('x')  *# x-label*
39. ax.set\_ylabel('y')  *# y-label*
40. ax.set\_title("Lagrange interpolation")  *# title*
41. ax.legend(loc='upper right', bbox\_to\_anchor=(1.18, 1)) *# 显示图例，对应ax.plot()中的lable属性*
42. ax.grid() *# 显示网格*
43. fig.savefig('lagrange.png', dpi=200) *# 保存图片*

**1．6 运行结果及分析**

运行结果如图4所示。

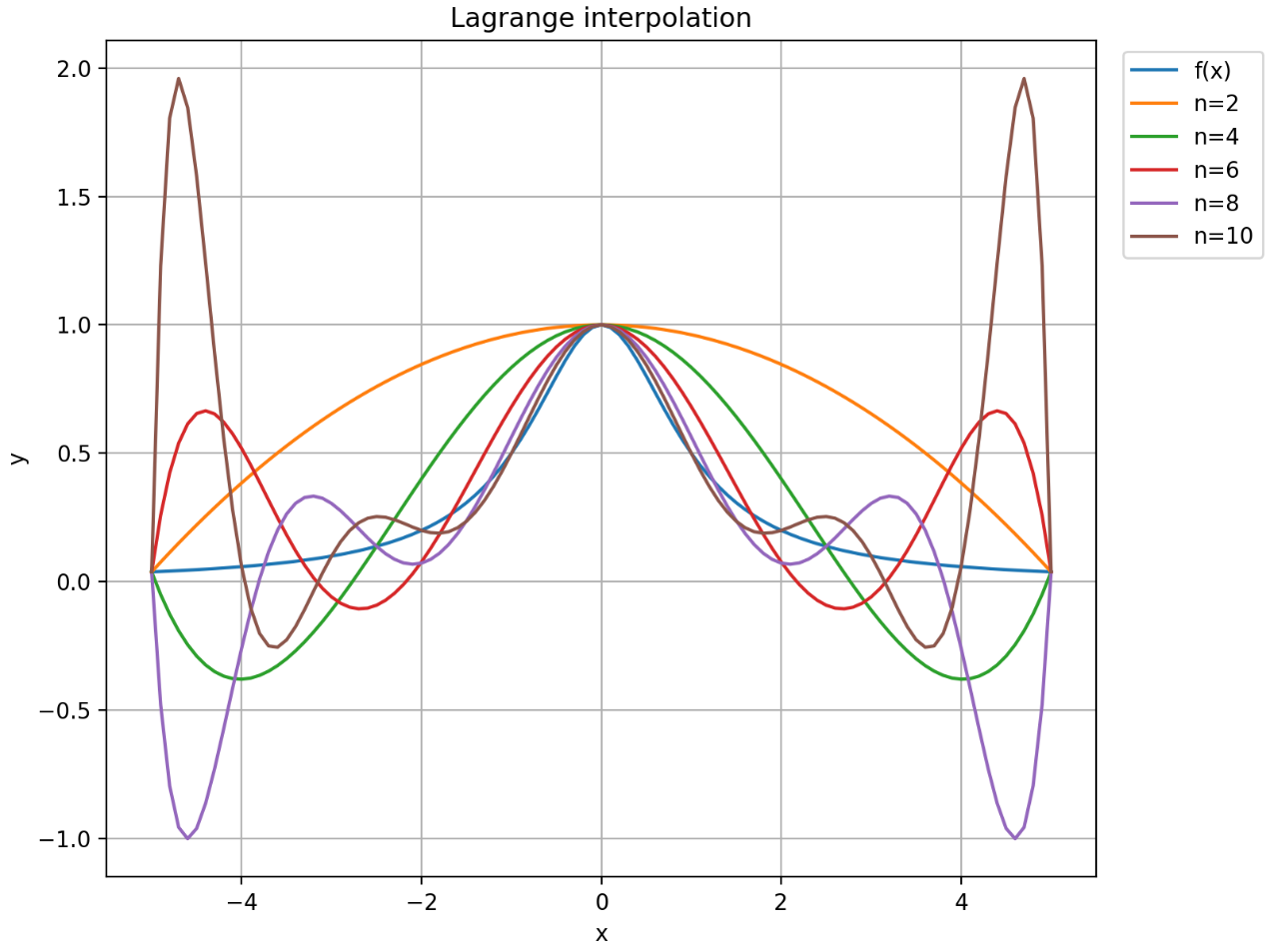


图4 运行结果

从图例可得n等于对应值的插值函数图像。且从图三整体可以看出，当n分别等于2、4、6、8、10时，随着n的增大，虽然插值函数在插值区间中部与原函数的拟合效果越好，但在区间两端出现了龙格现象，大幅地偏离了实际数值。同时，各个曲线与原函数的交点个数与在插值区间内所选节点个数相同。因此，当插值阶数较高时，选用拉格朗日插值法会使得结果误差极大。

**1．7 心得体会**

因为Python3的熟练掌握，与matplotlib库的强大支持，很快地完成了这次任务。

在实验过程中遇到的主要问题和解决方法是：

1. list无法直接进行运算，需要使用np.array转化为数组类型；
2. 使用np.arrange()生成样本点时应该注意上限是开的，需要选择5.1作为上限，才能画出正确的图像。
3. 使用bbox\_to\_anchor参数来精确控制图例位置。

由于自己写过一遍代码，加深了对拉格朗日插值多项式的理解，掌握了

数学公式到编程代码的转化思想，激发了对编程的热爱。