



计算机与软件工程学院

上机报告

**（ 2021/2022 学年 第 1 学期 ）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | **数值计算（C++）** | | | | | |
| 课程代码 | **190901319** | | | | | |
| 上机时间 | 20211 | 年 | 11 | 月 | 10 | 日 |
| 指导单位 | 物联网工程系 | | | | | |
| 任课教师 | 李显勇 | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生姓名 | 李子涵 | | |
| 学 号 | 3120190971401 | | |
| 成 绩 |  | 年级专业 | 计算机科学与技术 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称** | **【实验1】拉格朗日插值，牛顿插值** | | | **实验地点** | 8-322 |
| **实验类型** | **验证** | **实验学时** | **2** | **实验日期** | 2021-11-10 |
| **实验目的和要求**   1. 了解**拉格朗日插值**基本理论，编程实现该算法； 2. 了解**牛顿插值**基本理论，编程实现该算法； 3. 提交以下文档（用**学号+姓名**作为文件夹名，将所有内容放于该文件夹中）：   1）实验报告  2）程序代码 | | | | | |
| **实验环境(实验设备)**  windows 10及以上版本；PC；Python 3.7 | | | | | |
| **实验原理及内容**  **一、实验原理**  根据**拉格朗日插值，牛顿插值**相关知识和算法编程完成本实验  **二、实验内容**  已知1920年-1970年美国人口如下   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 年份 | 1920 | 1930 | 1940 | 1950 | 1960 | 1970 | | 人口（千人） | 105711 | 123203 | 131669 | 150697 | 179323 | 203212 |  1. 构造5次Lagrange插值多项式，并用此估计1910年，1965年和1975年的人口数； 2. 用牛顿插值估计1965年和1975年的人口数。   **三、实验过程（可以文字说明+运行结果截图）**  函数代码：  *#Lagrange插值多项式* **def** Lang(val):  x=[pairs[i] **for** i **in** range(len(pairs)) **if** i%2==0]*#x存放插值节点的x值* y=[pairs[i] **for** i **in** range(len(pairs)) **if** i%2!=0]*#y存放插值节点的y值* ans = 0  **for** i **in** range(n):  temp=1*#拉格朗日基函数* **for** j **in** range(i-1):  **if** j!=i:  temp=temp\*(val-x[j])/(x[i]-x[j])  **for** j **in** range(i-1,n):  **if** j>=0 **and** j!=i:  temp=temp\*(val-x[j])/(x[i]-x[j])  ans=ans+temp\*y[i]*#计算插值多项式* print(**"拉格朗日插值多项式："**)  print(**'f({}) = {} (保留三位小数)'**.format(val, format(ans, **'.3f'**)))  *#print(ans)  #牛顿插值多项式* **def** Newton(val):  *#构造差商表，每一列表示一组* x=[pairs[i] **for** i **in** range(len(pairs)) **if** i%2==0]*#x存放插值节点的x值* y=[pairs[i] **for** i **in** range(len(pairs)) **if** i%2!=0]*#y存放插值节点的y值* i = 1  g=[pairs[i] **for** i **in** range(len(pairs)) **if** i%2!=0]*# g[k]表示f[0,1,2,...,k],首先将y值赋给g  #构造差商表* **while**(i<n):  k=0  j=i  **while**(j<n):  g[j]=(y[j]-y[j-1])/(x[j]-x[k])  j+=1  k+=1  y=[g[i] **for** i **in** range(len(g))]*#将此列差商存放进y* i+=1  t=1  i=1  ans=g[0]  **while**(i<n):*#计算插值多项式* t=t\*(val-x[i-1])  ans=ans+t\*g[i]  i+=1  print(**"牛顿插值多项式："**)  print(**'f({}) = {} (保留三位小数)'**.format(val, format(ans, **'.3f'**)))  *#print(ans)*  *#主代码部分，下列过程说明只更改与展示此部分* **if** \_\_name\_\_==**'\_\_main\_\_'**:  pairs=[1920,105711,1930,123203,1940,131669,1950,150697,1960,179323]*#将每个点的x,y值成对列出 eg:(-2,17)(0,1)(1,2)(2,19)* **if**(len(pairs)%2!=0):*#若有奇数个点则给出提示* print(**"missing element"**)  n=int(len(pairs)/2)*#n表示插值点数量，插值多项式阶数* Newton(1965)*#给出要计算的值* Lang(1910)  实验过程：  （1）a.取前五个点构造拉格朗日插值多项式，估计1910年的人口数：  主代码：  **if** \_\_name\_\_==**'\_\_main\_\_'**:  pairs=[1920,105711,1930,123203,1940,131669,1950,150697,1960,179323]*#将每个点的x,y值成对列出 eg:(-2,17)(0,1)(1,2)(2,19)* **if**(len(pairs)%2!=0):*#若有奇数个点则给出提示* print(**"missing element"**)  n=int(len(pairs)/2)*#n表示插值点数量，插值多项式阶数* Lang(1910)  运行结果：    b.取后五个点构造拉格朗日插值多项式，估计1965和1975年的人口数：  主代码：  **if** \_\_name\_\_==**'\_\_main\_\_'**:  pairs=[1930,123203,1940,131669,1950,150697,1960,179323,1970,203212]*#将每个点的x,y值成对列出 eg:(-2,17)(0,1)(1,2)(2,19)* **if**(len(pairs)%2!=0):*#若有奇数个点则给出提示* print(**"missing element"**)  n=int(len(pairs)/2)*#n表示插值点数量，插值多项式阶数* Lang(1965)  Lang(1975)  运行结果：    （2）a.取后五个点，用牛顿插值估计1965年和1975年的人口数：  主代码：  **if** \_\_name\_\_==**'\_\_main\_\_'**:  pairs=[1920,105711,1930,123203,1940,131669,1950,150697,1960,179323]*#将每个点的x,y值成对列出 eg:(-2,17)(0,1)(1,2)(2,19)* **if**(len(pairs)%2!=0):*#若有奇数个点则给出提示* print(**"missing element"**)  n=int(len(pairs)/2)*#n表示插值点数量，插值多项式阶数* Newton(1965)  Newton(1975)  运行结果： | | | | | |

|  |
| --- |
| **四、实验小结**（包括问题和解决方法、心得体会、意见与建议等）  问题：  （1）在牛顿插值多项式中，虽然只需要每列的第一个差商，但是在构造差商表时只用一维数组存储差商会造成覆盖，无法计算出全部差商。  （2）牛顿插值多项式中需要运用多个数组，在C++中对数组的使用需要指定长度，使简单的问题复杂化  解决方法：  （1）看书上的代码思想后发现可以逆向求插商，先求差商表每列最后一个，那么覆盖以后也不会导致上方差商求不出，同时也降低了算法复杂度。  （2）换用Python语言，运用列表来存储插值点，较C++更简单，也更节约时间  心得体会：  用计算机语言将插值算法实现使我更加深了对该算法的理解，并进一步熟悉了求解过程，同时将数学计算过程转换为代码的过程也让我理解到通过数学推导过程可以优化算法，降低其时间与空间复杂度。此外，当遇到问题时首先要独立思考，其次再求助书本或网络。 |