**第1次上机实验报告**

**学号：3119009062 班级：软件工程（2） 姓名：刘龙**

目录

[实验2：减少运算次数的实验结果分析 1](#_Toc98479336)

[实验3：求解非线性方程的二分法实现 2](#_Toc98479337)

# 实验2：减少运算次数的实验结果分析

【实验目的】比较不同算法求多项式的运算次数与用时。

设计2种不同的算法计算以下函数的值，分别测试*x* = 0.1，1，2。

 (1)

【实验条件】

计算机配置：64 位操作系统, 基于 x64 的处理器

CPU：Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz

内存大小：8GB

操作系统：Windows

【算法介绍】

算法1：

蛮力法，按照给出等式，从左往右以此计算，得到结果

算法2：

秦九韶算法

【实验结果及分析】

表1 算法比较结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *x* | 算法 | 函数结果*f* | 乘法次数 | 加法次数 | 用时(秒) |
| 0.1 | 算法1 | 1.234567901234568 | 5000050000 | 100000 | 0.25435543060302734 |
| 算法2 | 1.234567901234568 | 100000 | 100000 | 0.022983789443969727 |
| 1 | 算法1 | 5000150001 | 5000050000 | 100000 | 0.1744983196258545 |
| 算法2 | 5000150001 | 100000 | 100000 | 0.02692437171936035 |
| 2 | 算法1 |  | 5000050000 | 100000 | 0.24238801002502441 |
| 算法2 |  | 100000 | 100000 | 0.3191087245941162 |

结果讨论：

使用秦九韶算法对于数据量为n的数据，无论是加法还是乘法，运行的次数都是n次，大大提高了计算的速率

【算法1源代码】

|  |
| --- |
| 把源代码贴于此处  from time import \*  import numpy as np  # f(x)=1+2x+3x^2+L+100001x^100000  # 赋初值  for x in [0.1, 1, 2]:  f = 1  add = 0 # 统计加法次数  multi = 0 # 统计乘法次数  start = time()  for i in range(100000):  f += (i + 2) \* (np.power(x, i + 1))  add += 1  multi += i + 1  end = time()  print('测试值为：', x)  if (x == 2):  print('二进制的位数:', bin(f))  else:  print('结果为：', f)  print('程序运行的时间：(s)', end - start)  print('加法的次数:', add, '乘法的次数:', multi) |

【算法2源代码】

|  |
| --- |
| 把源代码贴于此处  from time import \*  # f(x)=1+2x+3x^2+L+100001x^100000  # 赋初值  for x in [0.1, 1, 2]:  f = 100001  add = 0 # 统计加法次数  multi = 0 # 统计乘法次数  start = time()  for i in range(100000, 0, -1):  f = x \* f + i  add += 1  multi += 1  end = time()  print('测试值为：', x)  if (x == 2):  print('二进制的位数:', bin(f))  else:  print('结果为：', f)  print('程序运行的时间：(s)', end - start)  print('加法的次数:', add, '乘法的次数:', multi) |

# 实验3：求解非线性方程的二分法实现

【实验目的】

使用二分法求解方程x\*x-2=0的根

【实验结果及分析】

表2 二分法求解过程数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | 下限*x*low | 上限*x*up | (*x*up – *x*low)/2 | *f*((*x*up – *x*low)/2)  的正负性 |
| 0 | 1.3 | 1.5 | 1.4 | < 0 |
| 1 | 1.50000000000000 | 1.40000000000000 | 1.45000000000000 | <0 |
| 2 | 1.45000000000000 | 1.40000000000000 | 1.42500000000000 | >0 |
| 3 | 1.42500000000000 | 1.40000000000000 | 1.41250000000000 | >0 |
| 4 | 1.42500000000000 | 1.41250000000000 | 1.41875000000000 | <0 |
| 5 | 1.41875000000000 | 1.41250000000000 | 1.41562500000000 | >0 |
| 6 | 1.41562500000000 | 1.41250000000000 | 1.41406250000000 | >0 |
| 7 | 1.41562500000000 | 1.41406250000000 | 1.41484375000000 | <0 |
| 8 | 1.41484375000000 | 1.41406250000000 | 1.41445312500000 | >0 |
| 9 | 1.41445312500000 | 1.41406250000000 | 1.41425781250000 | >0 |
| 10 | 1.41425781250000 | 1.41406250000000 | 1.41416015625000 | >0 |
| 11 | 1.41425781250000 | 1.41416015625000 | 1.41420898437500 | <0 |
| 12 | 1.41425781250000 | 1.41420898437500 | 1.41423339843750 | <0 |
| 13 | 1.41423339843750 | 1.41420898437500 | 1.41422119140625 | >0 |
| 14 | 1.41422119140625 | 1.41420898437500 | 1.41421508789062 | >0 |
| 15 | 1.41421508789062 | 1.41420898437500 | 1.41421203613281 | >0 |
| 16 | 1.41421508789062 | 1.41421203613281 | 1.41421356201172 | <0 |
| 17 | 1.41421508789062 | 1.41421356201172 | 1.41421432495117 | <0 |
| 18 | 1.41421432495117 | 1.41421356201172 | 1.41421394348145 | >0 |
| 19 | 1.41421394348145 | 1.41421356201172 | 1.41421375274658 | >0 |
| 20 | 1.41421375274658 | 1.41421356201172 | 1.41421365737915 | >0 |
| 21 | 1.41421365737915 | 1.41421356201172 | 1.41421360969543 | >0 |
| 22 | 1.41421360969543 | 1.41421356201172 | 1.41421358585358 | >0 |
| 23 | 1.41421358585358 | 1.41421356201172 | 1.41421357393265 | >0 |
| 24 | 1.41421357393265 | 1.41421356201172 | 1.41421356797218 | >0 |
| 25 | 1.41421356797218 | 1.41421356201172 | 1.41421356499195 | >0 |
| 26 | 1.41421356499195 | 1.41421356201172 | 1.41421356350183 | >0 |
| 27 | 1.41421356350183 | 1.41421356201172 | 1.41421356275678 | >0 |
| 28 | 1.41421356275678 | 1.41421356201172 | 1.41421356238424 | >0 |
| 29 | 1.41421356238425 | 1.41421356201172 | 1.41421356219798 | >0 |
| 30 | 1.41421356238425 | 1.41421356219798 | 1.41421356229112 | <0 |
| 31 | 1.41421356238425 | 1.41421356229112 | 1.41421356233768 | <0 |
|  |  |  |  |  |

结果讨论：

一共迭代31次，最终得到的近似解为：

1.41421356233768

【算法源代码】

|  |
| --- |
| 把源代码贴于此处  # 画图 y=x\*x-2  # x = np.linspace(-5, 5, 100)  # y = x \* x - 2  # plt.figure()  # plt.plot(x, y)  # plt.plot([-4, 4], [0, 0])  # plt.show()  a = 2  LIMIT = 1e-20  def f(x):  return x \* x - a  xlow = float(input('请输入x值下限'))  xup = float(input('请输入x值上限'))  iter = 0  while (xup - xlow) \* (xup - xlow) > LIMIT:  # 计算重点值  middle = (xup + xlow) / 2  iter += 1  temp = f(middle)  if (temp > 0):  xup = middle  if (temp < 0):  xlow = middle  if (temp == 0):  print('方程根为', middle)  print('{:.15g} {:.15g} {:.15g}'.format(iter, xlow, xup)) |

附加题：对于大量的输出数据，有什么便捷的方法把列表数据输出，方便贴到word文档呢？或者直接把数据矩阵输出到.txt文件中呢？

调用xlwt库和excel有关的函数创建excel文件，详细代码如下所示

import xlwt

a = 2

LIMIT = 1e-20

def f(x):

return x \* x - a

xlow = float(input('请输入x值下限'))

xup = float(input('请输入x值上限'))

iter = 0

# 将数据写入excel表格

excel = xlwt.Workbook(encoding='utf-8', style\_compression=0)

sheet = excel.add\_sheet('数据', cell\_overwrite\_ok=True)

col = ['迭代次数', '上限', '下限', '中间值', 'f(中间值)正负性']

for i in range(len(col)):

sheet.write(0, i, col[i])

middle = (xup + xlow) / 2

while (xup - xlow) \* (xup - xlow) > LIMIT:

# 计算重点值

iter += 1

temp = f(middle)

if (temp > 0):

xup = middle

if (temp < 0):

xlow = middle

if (temp == 0):

print('方程根为', middle)

print('{:.15g} {:.15g} {:.15g}'.format(iter, xlow, xup))

flag = '>0'

if (temp < 0):

flag = '<0'

middle = (xup + xlow) / 2

datalist = [iter, xup, xlow, middle, flag]

for j in range(5):

sheet.write(iter, j, datalist[j])

savepath = 'excel表格.xls'

excel.save(savepath)