**中国矿业大学计算机学院**

**2021 级本科生课程报告**

课程名称： 最优化理论与方法

报告题目： 《0.618算法原理及实现》

报告时间： 2024.1.10

学生姓名： 杨学通

学 号： 08213129

专 业： 人工智能2021-1班

任课教师： 赵佳琦

**课程报告成绩评定及评分依据**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 评分依据 | 分值 | 实际得分 |
| 1 | 课程报告描述了最优化技术或算法的基本原理 | 20 |  |
| 2 | 课程报告使用python或C++程序设计语言（附源代码） | 40 |  |
| 3 | 给出至少2个测试用例，并且给出可视化的运行结果及分析 | 30 |  |
| 4 | 课程报告给出了个人学习体会 | 10 |  |
| 总分 | | |  |
| 教师评语：  教师签名：  年 月 日 | | | |

**摘 要**

0.618算法是一种用于优化问题的搜索算法，也称为黄金分割法。它基于黄金分割比例（0.618）来进行一位搜索，通过不断缩小搜索范围和调整步长来逼近最优解。该算法适用于连续函数的最优化问题，可以在较短的时间内找到较好的解。它的优点包括收敛速度快、对初始值不敏感、易于实现等。本文通过python的numpy库和sympy库复现了0.618算法，求解不同函数的最优值。

**关键词**：0.618算法；黄金分割；一维搜索；最优化问题；

目录

[1问题描述 1](#_Toc155817813)

[2算法原理及流程 1](#_Toc155817814)

[3运行代码 1](#_Toc155817815)

[4运行截图 3](#_Toc155817816)

[5总结与分析 4](#_Toc155817817)

# 

# 1问题描述

0.618法（黄金分割法）是黄金分割比例（0.618）来对函数进行一维搜索，求解最优值的算法。通过确定搜索方向，调整搜索步长，逐步减小变量的取值范围，从而逼近最优解。该算法适合于一维搜索，具有较快的收敛速度，屏蔽初始值的影响，易于实现等优势。

本次实验要求python实现0.618算法，并通过测试样例，验证0.618算法的正确性，体会0.618算法进行一维搜索的过程。

# 2算法原理及流程

Step1:置初始区间，精度要求，k=1

计算,

计算和

Step2:若，则停止计算，否则：

若，转Step3

若，转Step4

Step3:置，

置，

计算，转Step5

Step4:置，

置，

计算，转Step5

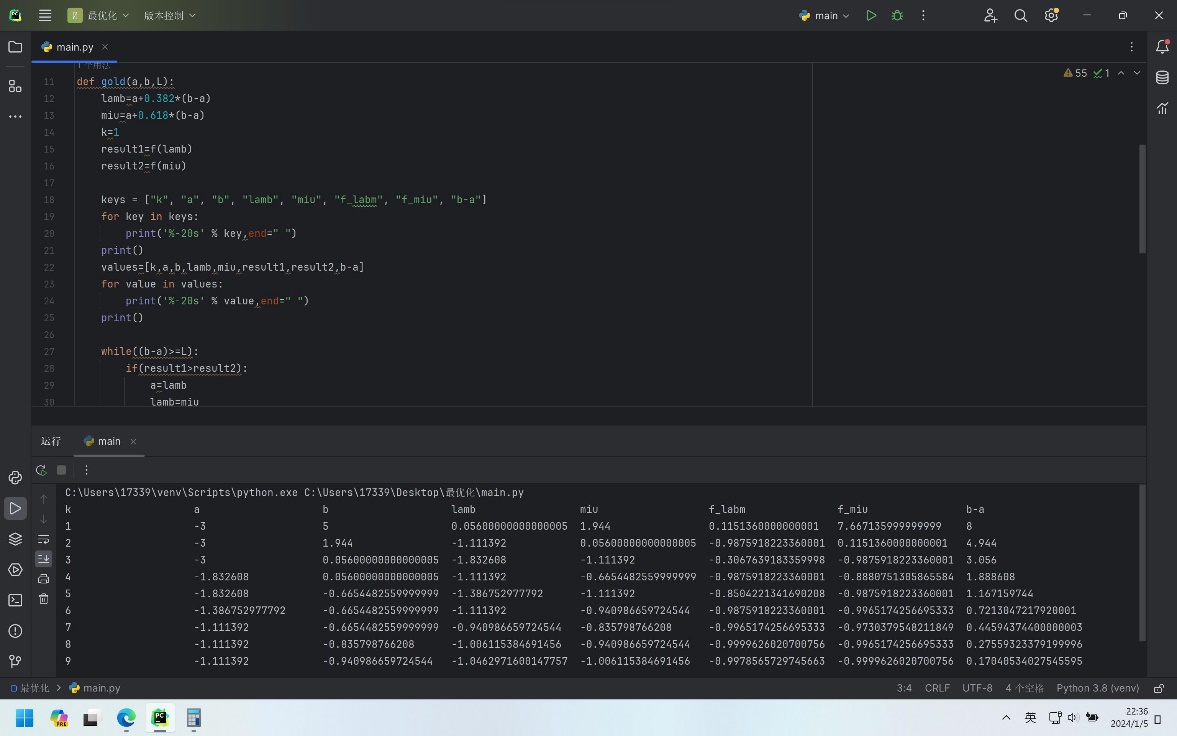
Step5:置k=k+1，返回Step2

# 3运行代码

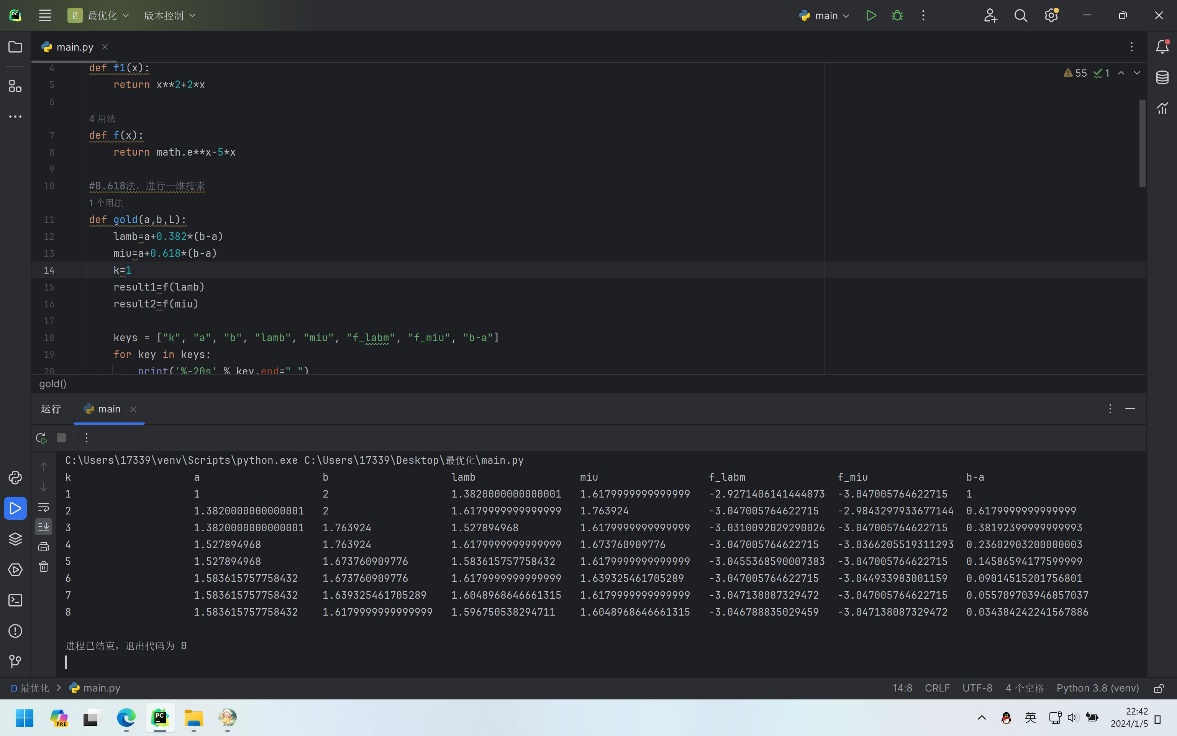
import math  
  
*#定义f(x)*def f1(x):  
 return x\*\*2+2\*x  
  
def f(x):  
 return math.e\*\*x-5\*x  
  
*#0.618法，进行一维搜索*def gold(a,b,L):  
 lamb=a+0.382\*(b-a)  
 miu=a+0.618\*(b-a)  
 k=1  
 result1=f(lamb)  
 result2=f(miu)  
  
 keys = ["k", "a", "b", "lamb", "miu", "f\_labm", "f\_miu", "b-a"]  
 for key in keys:  
 print('%-20s' % key,end=" ")  
 print()  
 values=[k,a,b,lamb,miu,result1,result2,b-a]  
 for value in values:  
 print('%-20s' % value,end=" ")  
 print()  
  
 while((b-a)>=L):  
 if(result1>result2):  
 a=lamb  
 lamb=miu  
 miu=a+0.618\*(b-a)  
 result1 = result2  
 result2=f(miu)  
 k+=1  
 values = [k, a, b, lamb, miu, result1, result2, b - a]  
 for value in values:  
 print('%-20s' % value,end=" ")  
 print()  
 else:  
 b=miu  
 miu=lamb  
 lamb=a+0.382\*(b-a)  
 result2 = result1  
 result1=f(lamb)  
 k+=1  
 values=[k,a,b,lamb,miu,result1,result2,b-a]  
 for value in values:  
 print('%-20s' % value,end=" ")  
 print()  
  
*#[a,b],L*gold(1,2,0.04)

# 4运行截图

实例1：，初始区间，精度L=0.001



实例2：，初始区间，精度L=0.04



# 5总结与分析

通过本次实验，我用python实现了0.618算法。通过使用不同的函数测试，取得很好的效果。同时，我还观察到了一些额外的现象和特点：

初始搜索范围的选择：我发现初始搜索范围的选择对算法的表现有较大的影响。当初始搜索范围过大时，算法可能需要更多的迭代次数才能收敛；而当初始搜索范围过小时，可能会错过全局最优解。

参数调节的灵敏度：我发现在调节0.618算法中的参数时，特别是黄金分割比例和步长的选择会对算法的性能产生显著的影响。合适的参数选择可以加速算法的收敛，而不合适的参数可能导致算法陷入局部最优解。

对不同类型函数的适用性：我测试了0.618算法在不同类型的函数上的表现，发现在某些函数上算法表现更为出色，而在某些函数上可能需要更多的迭代次数才能达到较好的结果。这表明算法的适用性受到目标函数的影响。

通过这些额外的观察和分析，我对0.618算法的特性有了更深入的理解，也为我在实际应用中更好地选择和调节算法提供了一些启示。