**《最优化技术》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2019级计算机科学与技术（卓越）02班** | | | **姓名** | **李燕琴** |
| **实验题目** | 一维搜索算法的应用 | | | | | |
| **实验时间** | **2021年5月7日** | | **实验地点** | **DS3401** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 ■设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确；□源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 一、实验目的  理解掌握一维搜索算法中的黄金分割法，牛顿法，并用于实际问题的求解。 | | | | | | |
| 二、实验项目内容   1. 给定一个函数，利用黄金分割法把区间压缩到长度只有0.23，需给出所有中间结果。 2. 给定一个函数 ，利用牛顿法求解该函数的最小值，需给出中间结果。   注意：所有程序请用python语言实现。只提交本电子文档，注意本文件末尾的文件命名要求；源程序一节请用代码备注的方式说明你的算法和思路；实验结果一节需要提供测试结果截图并给出结果分析。 | | | | | | |
| 三、实验过程或算法（源程序）  ======================第一题：黄金分割法===========================  **import** numpy  **import** matplotlib**.**pyplot **as** plt  **def** fun**(**x**):**  **return** 8**\***numpy**.**exp**(**1**-**x**)+**7**\***numpy**.**log**(**x**)**  # 第一步 确定以为搜索区间，进退法  **def** interval**(**a0**,** step**=**0.3**):**  a1 **=** a0  a2 **=** a0**+**step  # 得到值判断  f1 **=** fun**(**a1**)**  f2 **=** fun**(**a2**)**  **if** fun**(**a1**)<**fun**(**a2**):** # 更换为反向计算  a1**,**a2 **=** a2**,**a1  step **=** **-**step  a3 **=** a2 **+** step  **while** fun**(**a2**)** **>** fun**(**a3**):**  step **=** 2 **\*** step # 2倍以提高速率  a1 **=** a2  a2 **=** a3  a3 **=** a2 **+** step  **if** f1**<**f2**:**  **return** **[**a3**,** a1**]**  **else:**  **return** **[**a1**,** a3**]**  # 第二步 黄金分割法求解  **def** golden**(**a**,**b**,**exa**,**p**=**0.382**):**  cnt **=** 1  **while** **True:**  # print("第"+str(cnt)+"次迭代:a="+str(a)+",b="+str(b))  **print(**"第%d次迭代:a = %.3f, b = %.3f"**%(**cnt**,**a**,**b**))**  a1 **=** a **+** p**\*(**b**-**a**)**  b1 **=** b **-** p**\*(**b**-**a**)**  **if** fun**(**a1**)<**fun**(**b1**):**  b **=** b1  **else:**  a **=** a1  if abs(b-a)<exa:  return (b+a)/2  cnt = cnt+1  if \_\_name\_\_ == "\_\_main\_\_":  x0 = 3  [a,b] = interval(x0)  print("取值区间为["+str(a)+","+str(b)+"]")  x = golden(a,b,0.0001)  # print("x="+str(x)+",f(x)="+str(fun(x)))  print("x = %.3f, f(x) = %.3f"%(x,fun(x)))  # x = numpy.linspace(0, 3,1000)  # y = fun(x)  # plt.plot(x,y)  # plt.show()  =======================第二题：牛顿迭代法==========================  **import** numpy **as** np  **def** f**(**x**):**  **return** 60**-**10**\***x**[**0**]-**4**\***x**[**1**]+pow(**x**[**0**],**2**)+pow(**x**[**1**],**2**)-**x**[**0**]\***x**[**1**]**  **def** jacobian**(**x**):**  # [dx1, dx2] 第一阶导数  **return** np**.**array**([-**10**+**2**\***x**[**0**]-**x**[**1**],** **-**4**+**2**\***x**[**1**]-**x**[**0**]])**  **def** hessian**(**x**):**  # [[dx1x1,dx1x2],  # [dx2x1,dx2x2]] 第二阶导数  **return** **[[**2**,-**1**],[-**1**,**2**]]**  # 牛顿法求解  **def** newton**(**x0**,**e**):**  # linalg=linear+algebralinalg=linear+algebra 线性代数 算法  cnt **=** 1  max\_cnt **=** 100  x1 **=** x0  delta **=** 1  **while** cnt**<**max\_cnt**:**  x2 **=** x1 **-** np**.**dot**(**np**.**linalg**.**inv**(**hessian**(**x1**)),** jacobian**(**x1**))** # dot 点积，矩阵乘法，迭代公式  **print(**"第%d次迭代，x = "**%(**cnt**)** **+** **str(**x2**))**  **if** **sum((**x1**-**x2**)\*\***2**)<**e**:**  **return** **(**x1**+**x2**)/**2  x1 **=** x2  cnt **+=** 1  **print(**"x0 = " **+** **str(**x0**)** **+** "时，牛顿法求解函数最小值不收敛"**)**  **exit(-**1**)**  if \_\_name\_\_ == "\_\_main\_\_":  x0 = np.array([0,0])  x = newton(x0,1e-3)  print(x) | | | | | | |
| 四、实验结果及分析和（或）源程序调试过程  =======================第一题，黄金分割法==========================  实验结果输出如下，可以看到，经过五次迭代后，，此时    ======================第二题，牛顿迭代法===========================  如下图所示，牛顿迭代法迭代速度很快，当选择作为初始迭代点的时候，只需一次迭代即可到达极小值，对应。 | | | | | | |

注：电子文档命名要求：学号+姓名+实验序号