《数值计算与最优化技术》实验报告

年级、专业、班级	2021 级计算机科学与技术卓越 2 班	姓名	文红兵
实验题目	线性方程组和非线性方程的求解		
实验时间	2023 年 4 月 9 日	实验地点	DS3305
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

1. 理解并掌握求解线性方程组的方法（列主元法，雅可比迭代法，高斯-赛德尔迭代法），并用程序实现。
2. 理解并掌握非线性方程根（二分法，牛顿迭代法和割线法）的求解方法，并用程序实现。

二、实验项目内容

1）用列主元法求解下列线性方程组：

$$\begin{bmatrix} 1.1348 & 3.8326 & 1.1651 & 3.4017 \\ 0.5301 & 1.7875 & 2.5330 & 1.5435 \\ 3.4129 & 4.9317 & 8.7643 & 1.3142 \\ 1.2371 & 4.9998 & 10.6721 & 0.0147 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 9.5342 \\ 6.3941 \\ 18.4231 \\ 16.9237 \end{bmatrix}$$

报告创建时间：

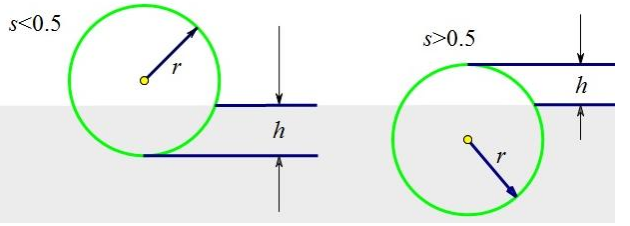
2）使用雅可比迭代法或高斯-赛德尔迭代法对下列方程组进行求解

$$\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2 \\ -x_1 + 10x_2 - 2x_3 = 8.3 \\ -x_1 - x_2 + 5x_3 = 4.2 \end{cases}$$

3）一个实心球体浮在水面上，根据阿基米德原理，球体的浮力和球体排出水的重量相等。令 $V_s = (4/3)\pi r^3$ 为球的体积，V_ω为球体局部浸入水中时排出水的体积。在静态平衡的状态下，球的重量和水的浮力相等，

$$\rho g V_s = \rho_w g V_w \quad 或 \quad s V_s = V_w$$

其中是ρ球体密度，g 是重力加速度，ρ_w 是水的密度，$s = \rho/\rho_w$ 是球体材料的比重，它决定球体沉入水面的深度。



如图，球体高度为 h 部分的体积为 $V_h=(\pi/3)(3rh^2-h^3)$.假设球体半径为 r=10cm, 密度为ρ=0.638，求 h？

（请自行取合适的初始值和初始区间，选用一种非线性方程求根算法求解。）

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）

（1）用列主元法求解下列线性方程组

```
import numpy as np
import copy

def Guess(A, b):  # A 系数矩阵，b 常数，返回 x 解
    n = len(b)
    for i in range(0, n - 1):  # 循环 n - 1 次，i 代表此时是第 i 行
        # 找出这一列最大的元素
        max_pos = i
        max_val = abs(A[i, i])
        for j in range(i + 1, n):
            if max_val < abs(A[j, i]):
                max_pos = j
                max_val = abs(A[j, i])
        print("第", i, "次最大的主元是：", A[max_pos, i])
        # 交换 A 和 b
        tp = copy.copy(A[i, :])  # 直接交换引用传递会出错
        A[i, :] = A[max_pos, :]
        A[max_pos, :] = tp
        b[i], b[max_pos] = b[max_pos], b[i]
        for j in range(i + 1, n):  # 循环消元，从 i 的下一行开始
            l = -A[j, i] / A[i, i]
            A[j, i:n] = A[j, i:n] + l * A[i, i:n]
            b[j] = b[j] + b[i] * l
    # 回代
    b[n - 1] = b[n - 1] / A[n - 1, n - 1]
    for i in range(n - 2, -1, -1):
        b[i] = (b[i] - np.dot(A[i, i + 1:n], b[i + 1:n])) / A[i, i]

    return b

if __name__ == '__main__':
    A = np.array([[1.1348, 3.8326, 1.1651, 3.4017],
                  [0.5301, 1.7875, 2.5330, 1.5435],
                  [3.4129, 4.9317, 8.7643, 1.3142],
                  [1.2371, 4.9998, 10.6721, 0.0147]], dtype="float64")
    b = np.array([9.5342, 6.3941, 18.4231, 16.9237], dtype="float64")
    print(Guess(A, b))  # 结果
```

（2）使用雅可比迭代法或高斯-赛德尔迭代法对下列方程组进行求解

1、雅可比迭代

```
import numpy as np


def Jacobi(A, b):
    """
    构造雅可比迭代式：X(k+1) = BX(k) + g
    其中 B：E - inv(D)*A,雅可比矩阵
       g：inv(D)*b
       D：A 的对角矩阵
    :param A：系数矩阵
    :param b：常数矩阵
    :return：x 的解
    """

    n = len(b)
    D = np.diag(np.diag(A))  # D 是 A 的对角矩阵
    D_inv = np.linalg.inv(D)  # D_inv 是 D 矩阵的逆矩阵
    B = np.eye(n) - np.dot(D_inv, A)  # B 矩阵是 雅可比矩阵
    g = np.dot(D_inv, b)  # inv(D)*b
    x = np.array([[1]] * n)  # 初始值都设为 1
    iter = 100  # 迭代 100 次

    for i in range(0, iter):
        x = np.dot(B, x) + g
        print("第", i, "次迭代：", x.reshape(n))
    return x.reshape(n)


if __name__ == '__main__':

    A = np.array([[10, -1, -2],
                  [-1, 10, -2],
                  [-1, -1, 5]], dtype="float64")
    b = np.array([[7.2], [8.3], [4.2]], dtype="float64")

    print(Jacobi(A, b))
```

2、高斯-赛德尔迭代

```
import numpy as np


def Gauss_Seidel(A, b):
    """
    构造迭代式：X(k+1) = BX(k) + g
    其中 B：-inv(L + D) * U ，  G-S 矩阵
       g：inv(L + D) * b
       D：A 的对角矩阵
       L：A 的下三角矩阵，但是没有对角元素
       U：A 的上三角矩阵，  但是没有对角元素
    :param A：系数矩阵
    :param b：常数矩阵
    :return：x 的解
    """
    n = len(b)
    D = np.diag(np.diag(A))  # A 的对角矩阵
    L = np.tril(A, k=-1)  # A 的下三角矩阵，但是没有对角元素
    U = np.triu(A, k=1)  # A 的上三角矩阵，  但是没有对角元素
    tp = np.linalg.inv(L+D)
    B = np.dot(-tp, U)  # G-S 矩阵
    g = np.dot(tp, b)  # inv(L + D) * b
    x = np.array([[1]]*n)  # 初始值，全部设置为 1
    iter = 100  # 迭代 100 次
    for i in range(0, iter):
        x = np.dot(B, x) + g
        print("第",i ,"次迭代：", x.reshape(n))
    return x.reshape(n)

if __name__ == '__main__':

    A = np.array([[10, -1, -2],
                  [-1, 10, -2],
                  [-1, -1, 5]], dtype="float64")
    b = np.array([[7.2], [8.3], [4.2]], dtype="float64")

    print(Gauss_Seidel(A, b))
```

（3）求 h

```
import math

def get_h(P_liquid, P, R):
    """
    :param P_liquid: 液体密度
    :param P: 球体密度
    :param R: 球体半径
    :return: 返回高度
    """
    s = P / P_liquid
    V_ball = 4 / 3 * math.pi * R * R * R   # 球的体积
    V_down = s * V_ball  # 液体以下体积
    V_up = V_ball - V_down  # 液体以上体积
    V = 0
    if s >= 0.5:
        V = V_up
    else:
        V = V_down
    # 二分法寻找唯一根
    left = 0
    right = R
    e = 1e-9   # 精度
    cnt = 0
    while right - left > e:
        mid = (left + right) / 2
        cnt += 1
        print("第", cnt, "次迭代：", mid)
        y = 3 * R * mid**2 - mid**3 - 3 * V / math.pi
        if y > 0:
            right = mid
        else:
            left = mid
    return (right + left) / 2
if __name__ == '__main__':
    P = 0.638  # 球体的密度
    P_liquid = 1  # 液体的密度
    R = 10  # 球体的半径
    print(get_h(P_liquid, P, R))
```

四、实验结果及分析和（或）源程序调试过程

（1）用列主元法求解下列线性方程组

```
第 0 次最大的主元是： 3.4129
第 1 次最大的主元是： 3.212168932579331
第 2 次最大的主元是： -6.8656974881180455
[1. 1. 1. 1.]
```

得到方程的解是：1 ， 1， 1 ， 1

（2）使用雅可比迭代法或高斯-赛德尔迭代法对下列方程组进行求解

1、雅可比迭代

```
第 0 次迭代： [1.02 1.13 1.24]
第 1 次迭代： [1.081 1.18  1.27 ]
第 2 次迭代： [1.092  1.1921 1.2922]
第 3 次迭代： [1.09765 1.19764 1.29682]
第 4 次迭代： [1.099128 1.199129 1.299058]
第 5 次迭代： [1.0997245 1.1997244 1.2996514]
第 6 次迭代： [1.09990272 1.19990273 1.29988978]
第 7 次迭代： [1.09996823 1.19996823 1.29996109]
第 8 次迭代： [1.09998904 1.19998904 1.29998729]
第 9 次迭代： [1.09999636 1.19999636 1.29999562]
第 10 次迭代： [1.09999876 1.19999876 1.29999854]
第 11 次迭代： [1.09999958 1.19999958 1.2999995 ]
第 12 次迭代： [1.09999986 1.19999986 1.29999983]
第 13 次迭代： [1.09999995 1.19999995 1.29999994]
第 14 次迭代： [1.09999998 1.19999998 1.29999998]
第 15 次迭代： [1.09999999 1.19999999 1.29999999]
第 16 次迭代： [1.1 1.2 1.3]
第 17 次迭代： [1.1 1.2 1.3]
[1.1 1.2 1.3]
```

得到方程的解是：1.1， 1.2 ， 1.3

2、高斯-赛德尔迭代

```
第 0 次迭代： [1.02   1.132  1.2704]
第 1 次迭代： [1.08728   1.192808  1.2960176]
第 2 次迭代： [1.09848432 1.19905195 1.29950725]
第 3 次迭代： [1.09980665 1.19988212 1.29993775]
第 4 次迭代： [1.09997576 1.19998513 1.29999218]
第 5 次迭代： [1.09999695 1.19999813 1.29999902]
第 6 次迭代： [1.09999962 1.19999976 1.29999988]
第 7 次迭代： [1.09999995 1.19999997 1.29999998]
第 8 次迭代： [1.09999999 1.2        1.3       ]
第 9 次迭代： [1.1 1.2 1.3]
第 10 次迭代： [1.1 1.2 1.3]
[1.1 1.2 1.3]
```

得到方程的解是：1.1， 1.2 ， 1.3

分析：

列主元法和雅可比迭代法的对比：

1、列主元法是通过不断地选取消元过程中的主元来避免舍入误差的扩散，因此可以得到精确的解。而雅可比迭代法则是通过不断地逐个更新未知量的值来逼近方程组的解。

2、列主元法的计算量较大，但通常只需要一次消元就能得到结果。而雅可比迭代法需要进行多次迭代才能收敛到正确的解，每次迭代的计算量较小，但总计算量可能会比列主元法更大。

3、列主元法对矩阵系数的大小没有限制，而雅可比迭代法要求方程组的矩阵系数必须严格满足对角占优条件，否则可能无法收敛。

雅可比迭代法和高斯-赛德尔迭代法的对比：

1、在每次迭代中，高斯赛德尔迭代法通过使用最新计算得到的未知量来更新下一次迭代的所有未知量，而雅可比迭代法则是通过使用上一次迭代得到的所有未知量来计算下一次迭代的每个未知量。

3、高斯赛德尔迭代法对于某些满足条件的线性方程组可以比雅可比迭代法更快地收敛到正确的解，因为它可以利用已经计算好的未知量来更快地逼近正确解。

（3）求 h

```
第 1 次迭代： 5.0
第 2 次迭代： 7.5
第 3 次迭代： 8.75
第 4 次迭代： 8.125
第 5 次迭代： 8.4375
第 6 次迭代： 8.28125
第 7 次迭代： 8.203125
第 8 次迭代： 8.1640625
第 9 次迭代： 8.14453125
第 10 次迭代： 8.134765625
第 11 次迭代： 8.1396484375
第 12 次迭代： 8.13720703125
第 13 次迭代： 8.138427734375
第 14 次迭代： 8.1390380859375
第 15 次迭代： 8.13873291015625
第 16 次迭代： 8.138580322265625
第 17 次迭代： 8.138504028320312
第 18 次迭代： 8.138465881347656
第 19 次迭代： 8.138484954833984
第 20 次迭代： 8.138494491577148
8.13849925994873
```

设置精度为 10^{-5}，采用二分法迭代答案是 8.13849925994873

具体分析如下：

求解高度 h 的公式如下

$$3rh^2 - h^3 - \frac{3V}{\pi} = 0$$

令

$$f(h) = 3rh^2 - h^3 - \frac{3V}{\pi} (0 < h < r)$$

即求解 $f(h)$ 的零点

又因为

$$f(h)' = 3h(2r - h) \qquad (0 < \mathrm{h} < \mathrm{r})$$

所以

$$f(h)' \geq 0$$

又因为

$$f(0) < 0$$

$$f(r) > 0$$

所以一定有唯一零点位于 $[0, r]$

注：电子文档命名要求：学号+姓名+实验序号