第四次作业

赖显松 2021214726

1 书面作业题目

P90 4

解: $A=\begin{bmatrix} a & 1 \\ 1 & a \end{bmatrix} = D-L-U \qquad D=\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \quad L=\begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix} \quad U=\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$

则 $B_J = D^{-1}(L+U)$

$$= \begin{bmatrix} \frac{1}{a} & 0 \\ 0 & \frac{1}{a} \end{bmatrix}\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{a} \\ -\frac{1}{a} & 0 \end{bmatrix}$$

$\det[\lambda I - B_J] = \det\begin{bmatrix} \lambda & \frac{1}{a} \\ \frac{1}{a} & \lambda \end{bmatrix} = \lambda^2 - \frac{1}{a^2} \quad \therefore \rho(B_J) = \left|\frac{1}{a}\right|$

J法收敛的条件为 $a \in (-\infty,-1)\cup(1,+\infty)$

$$B_G = (D-L)^{-1}U = \begin{bmatrix} a & 0 \\ 1 & a \end{bmatrix}^{-1}\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{a} & 0 \\ -\frac{1}{a^2} & \frac{1}{a} \end{bmatrix}\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -\frac{1}{a} \\ 0 & \frac{1}{a^2} \end{bmatrix}$$

$\det[\lambda I - B_G] = \begin{vmatrix} \lambda & \frac{1}{a} \\ 0 & \lambda-\frac{1}{a^2} \end{vmatrix} = \lambda\left(\lambda - \frac{1}{a^2}\right)$ 则 $\rho(B_G) = \frac{1}{a^2}$

GS法收敛的条件为 $a \in (-\infty,-1)\cup(1,+\infty)$

(2)

J法与GS法的收敛速度之比

$$R(B_J)/R(B_G) = \frac{-\ln\rho(B_J)}{-\ln\rho(B_{GS})} = \frac{-\ln\left|\frac{1}{a}\right|}{-\ln\frac{1}{a^2}}$$

$= \frac{1}{2}$ 即J法收敛速率为GS法的$\frac{1}{2}$

P90 8 (2)

解：根据线性方程组

$$A=\begin{bmatrix}10 & -1 & 0\\ -1 & 10 & -2\\ 0 & -2 & 10\end{bmatrix}$$

$\Delta_1=10>0$

$\Delta_2=10\times10-(-1)\times(-1)=99>0$

$\Delta_3=10\times10\times10-(10+40)=950>0$

∴ 方程组系数矩阵A为对称正定的.

$$B_J=D^{-1}(L+U)=\begin{bmatrix}\frac{1}{10} & 0 & 0\\ 0 & \frac{1}{10} & 0\\ 0 & 0 & \frac{1}{10}\end{bmatrix}\begin{bmatrix}0 & -1 & 0\\ -1 & 0 & -2\\ 0 & -2 & 0\end{bmatrix}=\begin{bmatrix}0 & -\frac{1}{10} & 0\\ -\frac{1}{10} & 0 & -\frac{1}{5}\\ 0 & -\frac{1}{5} & 0\end{bmatrix}$$

$$\det[\lambda I-B_J]=\begin{vmatrix}\lambda & \frac{1}{10} & 0\\ \frac{1}{10} & \lambda & \frac{1}{5}\\ 0 & \frac{1}{5} & \lambda\end{vmatrix}=\lambda^3-\left(\frac{1}{100}+\frac{1}{25}\right)\lambda \quad 则\ \rho(B_J)=\sqrt{\frac{1}{20}}$$

$$=\lambda\left(\lambda^2-\frac{1}{20}\right)$$

最优松弛因子

$$\omega_b=\frac{2}{1+\sqrt{1-\frac{1}{20}}}=1.0128$$

当 $\omega=\omega_b$ 时 $\rho(\mathcal{L}_{\omega_b})=\omega_b-1=0.0128$

渐近收敛率：

$$R(\mathcal{L}_{\omega_b})=-\ln\rho(\mathcal{L}_{\omega_b})=4.357$$

2 编程题目

2.1 五对角矩阵

编写生成五对角矩阵的函数（输入：对角元素列表 nums，阶次 n；返回：n 阶的多对角矩阵矩阵）。代码如下：

```python
def geneDlgMat(nums: list, n: int):
```

```python
    ap = len(nums)
    if ap % 2 == 1:
        A = np.zeros((n, n + ap - 1))
        for i in range(n):
            for j in range(ap):
                A[i,i + j] = nums[j]
        A = A[:,int((ap-1)/2):n+int((ap-1)/2)]
        return A
    else:
        print('请输入一个奇数')
        return None
```

生成一个 10 阶五对角矩阵，对角元素为（1，-8，20，-8，1），如图 2.1 所示：

	0	1	2	3	4	5	6	7	8	9
0	20.00000	-8.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1	-8.00000	20.00000	-8.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	1.00000	-8.00000	20.00000	-8.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	1.00000	-8.00000	20.00000	-8.00000	1.00000	0.00000	0.00000	0.00000	0.00000
4	0.00000	0.00000	1.00000	-8.00000	20.00000	-8.00000	1.00000	0.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	1.00000	-8.00000	20.00000	-8.00000	1.00000	0.00000	0.00000
6	0.00000	0.00000	0.00000	0.00000	1.00000	-8.00000	20.00000	-8.00000	1.00000	0.00000
7	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	-8.00000	20.00000	-8.00000	1.00000
8	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	-8.00000	20.00000	-8.00000
9	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	-8.00000	20.00000

图 2.1 10 阶五对角矩阵

2.2 迭代法求解线性方程组

迭代法解线性方程组的思想为[1]：通过构造迭代函数 $\vec{x}^{k+1}=B\vec{x}^{k}+\vec{f}$，使得通过这种方式产生的序列$\{\vec{x}^{(k)}\}$若能满足(2 - 1)：

$$\lim_{k\to\infty}\vec{x}^{(k)}=\vec{x}^{*},\forall\vec{x}^{(0)}\in\mathbb{R}^{n} \qquad (2-1)$$

则晨构造函数迭代法为收敛的，且经过一定次数的迭代，随机的序列可以近似等于至线性方程组解的真实值。

数值方法这门课程主要涉及的迭代方法有 Jacobi 迭代法、Gauss-Seidel 迭代法和超松弛迭代法。

2.3 Jacobi 迭代法及 python 程序实现

迭代法需要将系数矩阵 A 先拆分为 3 个不同的矩阵部分作为构造迭代函数用到的元素：对角元矩阵 D，零对角上三角矩阵 U 和零对角下三角矩阵 L，并且满足 $A=D-L-U$。

获取矩阵迭代函数元素的 python 代码如下：

```python
def Mat2LUD(A):
    n = A.shape[0]
    D = np.zeros((n, n))
    L = np.zeros((n, n))
    U = np.zeros((n, n))
    for i in range(n):
        D[i, i] = A[i, i]
        idx1 = np.arange(0,i)
        idx2 = np.arange(i+1,n)
        L[i, idx1] = -A[i, idx1]
        U[i, idx2] = -A[i, idx2]
    return L, U, D
```

Jacobi 迭代法的迭代函数为(2 - 2)：

$$\begin{aligned}&\vec{x}^{k+1} = B\vec{x}^{k} + \vec{f}\\&B_J = D^{-1}(L+U) = I - D^{-1}A\\&f_J = D^{-1}\vec{b}\end{aligned} \tag{2 - 2}$$

Jacobi 迭代法迭代函数获取的 Python 代码如下：

```python
def BJandfJ(Mat_A, b):
    n = Mat_A.shape[0]
    E = np.eye(n)
    Mat_L, Mat_U, Mat_D = Mat2LUD(Mat_A)
    BJ = E-(np.linalg.inv(Mat_D))@(Mat_A)
    fJ = np.linalg.inv(Mat_D)@b
    return BJ, fJ
```

题目通过 Jacobi 迭代法求矩线性方程组的流程为：

1 生成系数矩阵、和 $\vec{b}$

2 取 $\vec{x}^{(0)} = (1,1,\ldots,1)^{\mathrm{T}}$

3 设置迭代误差范数计算方法、迭代停止精度、初始化迭代误差

4 循环：在误差未满足精度前，通过迭代函数获取新的序列，比较新旧序列差值范数是否小于精度：若是，退出循环；若否将新序列作为旧序列继续迭代。

具体 python 代码实现如下（精度设置为 10^{-6}），其中设置一个循环计算 $n = 10, 20, 40$ 时五对角矩阵线性方程组的结果：

```python
for n in [10, 20, 40]:
    A = SM.geneDlgMat([1, -8, 20, -8, 1], n)
    b = np.zeros(n)
    xOld = np.ones(n)
```

```python
    xTrue = li.solve(A, b)
    s = -6  # 选择精度
    method = 'inf'
    itNumJ = 0
    eJ = IS.iterNorm(np.zeros(n), xOld, method)  # 初始值

    B, f = IS.BJandfJ(A, b)  # J 法收敛次数
    while eJ > 10 ** s:
        xNewJ = B @ xOld + f
        eJ = IS.iterNorm(xOld, xNewJ, method)
        xOld = xNewJ
        itNumJ += 1
    print('J 法>>', n, '阶 5 对角的迭代次数: ', itNumJ)
```

结果如图 2. 2 所示：

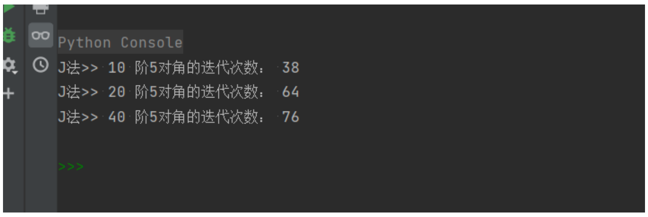

图 2. 2 10，20，40 阶五对角矩阵线性方程组的迭代次数

10、20、40 阶五对角矩阵线性方程组的收敛次数约为 38、64 和 76。可见，Jacobi 迭代法的是收敛的，且随着五对角矩阵阶次的增加，收敛的次数增多。

2.4 SOR 迭代法及 python 程序实现

SOR 迭代法的迭代函数为(2 - 3)：

$$\begin{aligned} \vec{x}^{k+1} &= L_{\omega}\vec{x}^{k} + \omega(D-\omega L)^{-1}\vec{b} \\ L_{\omega} &= (D-\omega L)^{-1}[(1-\omega)D+\omega U] \end{aligned} \quad (2 - 3)$$

其中 ω 为松弛因子，是影响 SOR 迭代法收敛速度的一个重要参数。

SOR 迭代法迭代函数获取的 Python 代码如下：

```python
def SOR(A, b, omiga):  # 迭代因子 omiga
    n = A.shape[0]
    E = np.eye(n)
    A_L, A_U, A_D = Mat2LUD(A)
    Lw_B = (np.linalg.inv(A_D-omiga*A_L)) @ (omiga*A_U+(1-omiga)*A_D)
    Lw_f = omiga*np.linalg.inv(A_D-omiga*A_L) @ b
    return Lw_B, Lw_f
```

题目通过 SOR 迭代法求矩线性方程组的流程为：

1 生成系数矩阵、和$\vec{b}$

2 取 $\vec{x}^{(0)}=(1,1,...,1)^{\mathrm{T}}$

3 设置迭代误差范数计算方法、迭代停止精度、初始化迭代误差

4 循环：在误差未满足精度前，通过迭代函数获取新的序列，比较新旧序列差值范数是否小于精度：若是，退出循环；若否将新序列作为旧序列继续迭代。

具体 python 代码实现如下（精度设置为 10^{-6}），其中设置一个循环计算 $n=10,20,40$，ω 从 $(0.5,2)$ 中间隔 0.05 取值，并且时五对角矩阵线性方程组的结果，计算迭代的渐进收敛速度，最后绘制收敛速度和松弛因子取值的关系曲线：

```python
w = np.arange(0.5, 2, 0.05)
    k = np.zeros(len(w))
    RB_Lw_inf = np.zeros(len(w))
    for i in range(len(w)):
        xOld = np.ones(n)
        itNumJ = 0
        eJ = IS.iterNorm(np.zeros(n), xOld, method)  # 初始值
        B, f = IS.SOR(A, b, w[i])
        RB_Lw_inf[i] = -np.log(np.linalg.norm(B, np.inf))
        while eJ > 10 ** s:
            xNewJ = B @ xOld + f
            eJ = IS.iterNorm(xOld, xNewJ, method)
            xOld = xNewJ
            itNumJ += 1
        # eJFinal = IS.iterNorm(xTrue, xNewJ, method)  # 最终误差
        print('SOR 法>>w==', w[i], n, '阶 5 对角的迭代次数: ', itNumJ)
        k[i] = itNumJ

    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    plt.subplot(1, 2, 1)
    plt.plot(w, k)
    plt.xlabel('松弛因子')
    plt.ylabel('迭代次数 k')
    plt.subplot(1, 2, 2)
    plt.plot(w, RB_Lw_inf)
    plt.xlabel('松弛因子')
    plt.ylabel('渐进收敛速度 R(B)')
    plt.show()
```

结果如图 2. 3 所示：

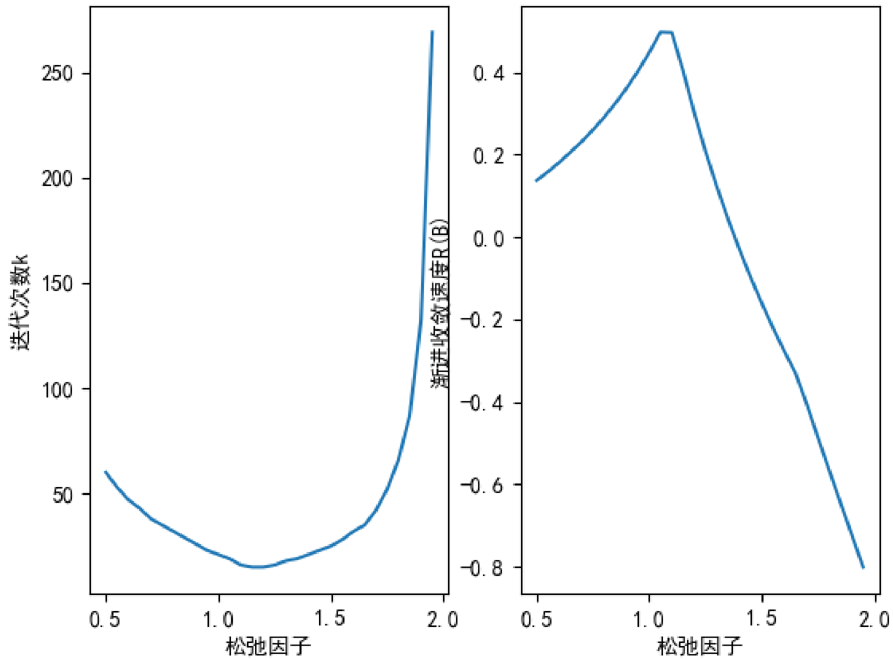

图 2. 3 10 阶五对角矩阵线性方程组的迭代次数随松弛因子变化

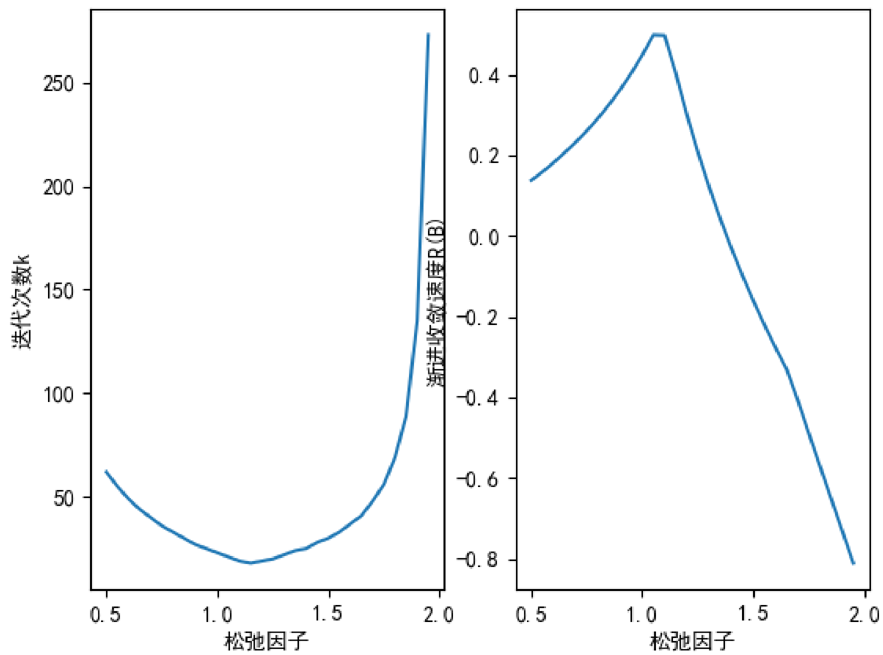

图 2. 4 20 阶五对角矩阵线性方程组的迭代次数随松弛因子变化

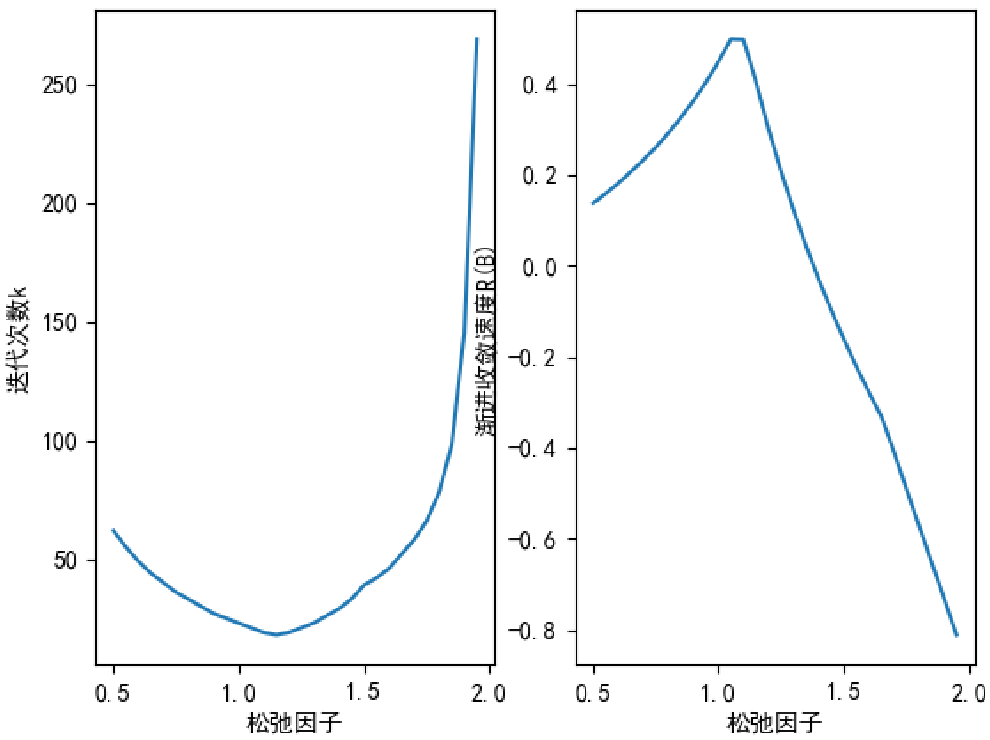

图 2. 5 40 阶五对角矩阵线性方程组的迭代次数随松弛因子变化

可见，SOR 迭代法的是收敛的，且随着五对角矩阵阶次的增加，收敛的次数并没有什么的变化。10、20、40 阶五对角矩阵线性方程组的最低收敛次数约为 15、18 和 18。而且 SOR 迭代法的收敛速度（渐进收敛率）随松弛因子在 $(0,2)$ 内变化呈现先下降后上升的趋势，在松弛因子 ω 约为 1.2 左右时速度最快。

参考文献

[1] 关冶．数值方法[M]．北京：清华大学出版社，2006：66-92．

附录

见附件 python 源代码。