实验五

1. 幂法

第五章上机题1: 编程实现幂法，用它求下列矩阵按模最大的特征值 λ_1 及其对应的特征向量 $\mathbf{x_1}$，使 $|(\lambda_1)_{k+1} - (\lambda_1)_k| < 10^{-5}$.

(1) $\mathbf{A} = \begin{bmatrix} 5 & -4 & 1 \\ -4 & 6 & -4 \\ 1 & -4 & 7 \end{bmatrix}$.

(2) $\mathbf{B} = \begin{bmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{bmatrix}$

幂法的数学原理

幂法（Power Iteration）是一种用于计算矩阵**主特征值**（即绝对值最大的特征值）及其对应**特征向量**的迭代算法。设矩阵 $A \in \mathbb{R}^{n\times n}$ 有特征值 $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$，对应的特征向量为 $v_1, v_2, \ldots, v_n$。幂法的核心思想是通过迭代计算 $A^k x$，使得 x 逐渐收敛到主特征向量 v_1。

数学推导：

1. 初始向量 x_0 可以表示为特征向量的线性组合：

$$x_0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

2. 迭代 k 次后：

$$A^k x_0 = c_1 \lambda_1^k v_1 + c_2 \lambda_2^k v_2 + \cdots + c_n \lambda_n^k v_n$$

3. 由于 $|\lambda_1| > |\lambda_i|$ （$i \geq 2$），当 $k \to \infty$ 时，$(\lambda_i/\lambda_1)^k \to 0$，因此：

$$A^k x_0 \approx c_1 \lambda_1^k v_1$$

4. 归一化后，$x_k = \dfrac{A^k x_0}{\|A^k x_0\|} \to \pm v_1$。

5. 估计主特征值 $\lambda = v_i$，其中 $i = \arg\max_j |v_j|$。（进阶版也可以用 Rayleigh 商 $\lambda_1 \approx \dfrac{(Ax_k)^T x_k}{x_k^T x_k}$）。

算法流程

1. **初始化**：随机选择一个初始向量 v （通常设为全 1 向量）。

2. **归一化**：计算 $u = \frac{v}{\|v\|}$。
3. **迭代**：
   - 计算 $v = Au$。
   - 估计主特征值 $\lambda_{\text{new}} = v_i$，其中 $i = \arg\max_j |v_j|$。
   - 归一化 $u = \frac{v}{\|v\|}$。
   - 检查收敛条件：$|\lambda_{\text{new}} - \lambda_{\text{old}}| < \epsilon$。
4. **输出**：主特征向量 u 和主特征值 λ_{new}。

结果

```
1   x_A = [ 0.4497837  -0.66731639  0.59361895],
2   lambda_A = -8.17750578617666
3
4   x_B = [-0.50156506  0.83044375 -0.2085536   0.12369746],
5   lambda_B = 81.8167283711789
```

---

2. QR 算法

第五章上机题3: 编程实现基本的QR算法（其中QR分解可以调用现成的函数），用它计算

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix}$$

的所有特征值, 观察迭代过程中矩阵序列收敛的情况, 然后解释观察到的现象.

QR 算法是一种用于计算矩阵所有特征值的经典算法。其基本思想是通过不断进行 QR 分解和矩阵乘法，使矩阵逐渐收敛为一个上三角矩阵（或分块上三角矩阵），从而可以直接读取特征值。

数学原理

给定一个方阵 A，QR 算法的迭代步骤如下：

1. **QR 分解**：将 A_k 分解为正交矩阵 Q_k 和上三角矩阵 R_k：

$$A_k = Q_k R_k$$

2. **矩阵乘法**：用 R_k 和 Q_k 构造新的矩阵 A_{k+1}：

$$A_{k+1} = R_k Q_k$$

3. **收敛判断**：重复上述步骤直到 A_k 收敛为一个上三角矩阵（或对角阶至多为 2 的分块上三角矩阵），此时可从对角线以及 2 阶对角块求得特征值。

结果

```
代码块
1   QR 算法共迭代 999 次，得到特征值: [0.5 0.5 0.5 0.5]
```

可见 QR 算法完全不收敛，这是因为：

- 基本 QR 算法的收敛速度取决于特征值之间的比值 $\frac{\lambda_{i+1}}{\lambda_i}$，因此如果矩阵的特征值绝对值相近（例如对称矩阵或特征值分布密集），收敛会非常缓慢。
- 此外，特征值重数较高（这里的 1 是重三的特征值），无位移 QR 算法无法区分多个 1 的特征空间，导致缓慢收敛。
- 针对本题的 A，其实际特征值是 $\lambda = [1, 1, 1, -1]$（3 个 1 和 1 个 -1），它们的绝对值完全相等。此时基本 QR 算法的收敛速度会变得极慢，甚至无法收敛。

---

3. 带位移的 QR 算法

第五章上机题4: 编程实现带原点位移的 QR 算法，计算 $\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix}$ 的所有特征值, 观察迭代过程的收敛情况, 与上机题 3 的实验结果做比较.

带位移的 QR 算法是 QR 算法的改进版本，通过引入位移（shift）加速收敛，尤其是对于接近对角化的矩阵。

数学原理

1. **位移选择**：通常选择位移 s 为矩阵的右下角元素 $A(n,n)$，因为其特征值可能接近 $A(n,n)$。
2. **QR 分解**：对位移后的矩阵 $A_k - sI$ 进行 QR 分解：

$$A_k - sI = Q_k R_k$$

3. **矩阵乘法**：用 R_k 和 Q_k 构造新的矩阵 A_{k+1}：

$$A_{k+1} = R_k Q_k + sI$$

4. **收敛判断**：重复直到 A_k 收敛。

结果

```
代码块
1   带位移 QR 算法共迭代 3 步，得到特征值: [-1.  1.  1.  1.]
```

可见，带位移的 QR 算法不仅收敛而且很快，这是因为带位移的 QR 算法每次都减去一个近似特征值 s ，使得：

$$A - sI = QR\,,\ A' = RQ + sI$$

这个策略通过选取合适的 s ，可以加速收敛：

- Shift 会将离 s 最近的特征值**快速放到对角线上**。
- 即使存在多个相同特征值，它也能“锁定”一个，逐步剥离。

对于本题的矩阵，选 $s = 0.5$ 就能很快得到特征值 1，从而让算法聚焦于剩余的部分。