Computional Homework16 Report

Student: 纪浩正, jihz2023@mail.sustech.edu.cn

1 Find all Eigenvalues of Matrix A

Given the symmetric matrix

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$$

We compute all eigenvalues using the **QR Iteration Algorithm** with **Householder QR decomposition**. The method repeatedly applies

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

which is a similarity transformation, therefore preserving eigenvalues. As $k \to \infty$, A_k converges to an upper-triangular matrix

$$A_k \to \begin{pmatrix} \lambda_1 & * & * & * \\ 0 & \lambda_2 & * & * \\ 0 & 0 & \lambda_3 & * \\ 0 & 0 & 0 & \lambda_4 \end{pmatrix}$$

so the diagonal values are the eigenvalues of A.

1.1 Householder Reflection to Construct Q

To transform a vector α to align with e_1, we construct a Householder reflection

$$H = I - 2\omega\omega^T, \qquad \omega = \frac{\alpha - \|\alpha\| e_1}{\|\alpha - \|\alpha\| e_1\|}$$

```
import numpy as np

def calH(alpha):
    alpha[0,0]-=np.linalg.norm(alpha)
    omega=alpha/np.linalg.norm(alpha)
    oo=omega@omega.T
```

```
    H=np.eye(oo.shape[0])-2*oo
    return H
```

Mathematically, applying H performs:

$$H_k A_{k-1} = \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$$

and repeating for $n-1$ steps yields

$$A = QR, \qquad Q = H_1 H_2 \cdots H_{n-1}.$$

1.2 QR Decomposition from Iterative Householder Transformations

```
def calQR(A):
    n=A.shape[0]
    subA=A.copy()
    Q=np.eye(n)
    for ii in range(n-1):
        alpha=subA[:,0].reshape(-1,1)
        h=calH(alpha)
        H=np.eye(n)
        H[ii:,ii:]=h
        Q=Q@H
        A=H@A
        subA=A[ii+1:,ii+1:].copy()
    return Q,A
```

1.3 QR Iteration to Extract Eigenvalues

We repeat QR decomposition until A_k becomes upper triangular:

$$\text{iterate:} \quad A_k = Q_k R_k, \qquad A_{k+1} = R_k Q_k.$$

Since A is symmetric, QR iteration converges to diagonal form:

$$A_k \to \Lambda = \text{diag}(\lambda_1, \lambda_2, \lambda_3, \lambda_4).$$

```
def calE(A,iter=200,tol=1e-10):
    Ak=A.copy().astype(float)
```

```
    for ii in range(iter):
        Q,R=calQR(Ak)
        Ak=R@Q
        if np.linalg.norm(Ak-np.triu(Ak)) < tol:
            break
    return np.diag(Ak)
```

1.4 Result

```
A=np.array([[10,7,8,7],
            [7,5,6,5],
            [8,6,10,9],
            [7,5,9,10]])

print(calE(A))
```

The eigenvalues obtained numerically are:

$$\lambda \approx [30.2887,\ 3.8581,\ 0.8431,\ 0.01015]$$

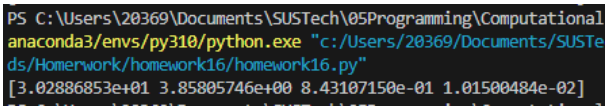

Figure 1 Diagonal entries after QR iterations (eigenvalues of A)