作业五

Noflowerzzk

2025.3.23

本文代码均用 Python 实现

1

LU 分解代码

```python
import numpy as np

def lu_decomposition(A):
    """ 对方阵 A 进行 LU 分解，返回 L 和 U """
    n = len(A)
    L = np.eye(n)
    U = A.astype(float).copy()

    for i in range(n):
        for j in range(i+1, n):
            if U[i, i] == 0:
                raise ValueError("不存在 LU 分解！")
            factor = U[j, i] / U[i, i]
            L[j, i] = factor
            U[j, i:] -= factor * U[i, i:]

    return L, U

# 测试
A = np.array([[2, -1, 3],
          [1, 2, 1],
          [2, 4, -2]], dtype=float)
L, U = lu_decomposition(A)

print("L:")
print(L)
print("U:")
print(U)
```

测试输出结果为

输出结果

```
L:
    [[1.   0.   0. ]
    [0.5 1.   0. ]
    [1.   2.   1. ]]
R:
    [[ 2.   -1.    3. ]
    [ 0.    2.5 -0.5]
    [ 0.    0.   -4. ]]
```

因此 $\det A = \det R = -20$.

2

Cholesky 分解代码

```python
import numpy as np

def cholesky_decomposition(A):
    """ 对称正定矩阵 A 进行 Cholesky 分解, 返回 L """
    n = A.shape[0]
    L = np.zeros_like(A)

    for i in range(n):
        for j in range(i + 1):
            sum_k = sum(L[i, k] * L[j, k] for k in range(j))

            if i == j:
                L[i, j] = np.sqrt(A[i, i] - sum_k)
            else:
                L[i, j] = (A[i, j] - sum_k) / L[j, j]

    return L

# 测试
A = np.array([[5, -2, 0],
              [-2, 3, -1],
              [0, -1, 1]], dtype=float)

L = cholesky_decomposition(A)
print("L:\n", L)
print("L.T:\n", L.T)
```

测试输出结果为

输出结果

```
L:
    [[ 2.23606798  0.          0.        ]
    [-0.89442719  1.4832397   0.        ]
    [ 0.         -0.67419986  0.73854895]]
L.T:
```

```
[[ 2.23606798 -0.89442719  0.        ]
 [ 0.          1.4832397  -0.67419986]
 [ 0.          0.          0.73854895]]
```

3

仅需 A 是正定阵即可，即其各阶顺序主子式大于 0. 计算得结果为 $a \in (-\sqrt{3}, \sqrt{3})$.

4

$$u_1 = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}$$

$$u_2 = \alpha_2 - \frac{\alpha_2 \cdot u_1}{u_1 \cdot u_1} u_1 = \frac{2}{9} \begin{pmatrix} -11 \\ 10 \\ 16 \end{pmatrix}$$

单位化即有 $Q = \begin{pmatrix} \frac{2}{3} & \frac{-11\sqrt{53}}{159} \\ -\frac{1}{3} & \frac{10\sqrt{53}}{159} \\ \frac{2}{3} & \frac{16\sqrt{53}}{159} \end{pmatrix}$

$$R = \begin{pmatrix} 3 & -\frac{3}{7} \\ 0 & \frac{2\sqrt{53}}{3} \end{pmatrix}$$

Householder 变换计算 QR 分解代码

```python
import numpy as np

def householder_qr(A):
    """ 使用 Householder 变换计算矩阵 A 的 QR 分解 """
    m, n = A.shape
    Q = np.eye(m)
    R = A.copy().astype(float)

    for k in range(n):
        # 选取列向量
        x = R[k:, k]

        # 计算法向量 v
        e1 = np.zeros_like(x)
        e1[0] = np.linalg.norm(x) * (1 if x[0] >= 0 else -1)
        v = x + e1
        v = v / np.linalg.norm(v)

        # 构造 Householder 变换矩阵 H_k = E - 2uu.T
        H_k = np.eye(m)
```

```python
            H_k[k:, k:] -= 2.0 * np.outer(v, v)

            # 更新 R 和 Q
            R = H_k @ R
            Q = Q @ H_k.T  # 注意 Q 需要不断右乘 H_k 的转置

    return Q, R

# 测试
A = np.array([[3, 14, 9],
            [6, 43, 3],
            [6, 22, 15]], dtype=float)

Q_A, R_A = householder_qr(A)

print("Q_A:\n", Q_A)
print("R_A:\n", R_A)

B = np.array([[1, 1, 1],
            [2, -1, -1],
            [2, -4, 10]], dtype=float)

Q_B, R_B = householder_qr(B)

print("Q_B\n", Q_B)
print("R_B:\n", R_B)
```

输出结果为

输出结果

```
Q_A:
    [[-0.33333333  0.13333333  0.93333333]
    [-0.66666667 -0.73333333 -0.13333333]
    [-0.66666667  0.66666667 -0.33333333]]
R_A:
    [[-9.00000000e+00 -4.80000000e+01 -1.50000000e+01]
    [ 1.77635684e-16 -1.50000000e+01  9.00000000e+00]
    [-4.21884749e-16 -6.21724894e-16  3.00000000e+00]]
Q_B
    [[-0.33333333 -0.66666667  0.66666667]
    [-0.66666667 -0.33333333 -0.66666667]
    [-0.66666667  0.66666667  0.33333333]]
R_B:
    [[-3.00000000e+00  3.00000000e+00 -6.33333333e+00]
    [-6.66133815e-16 -3.00000000e+00  6.33333333e+00]
    [ 6.66133815e-16  4.44089210e-16  4.66666667e+00]]
```

极小数值视为 0

因此 $B^{-1} = R^{-1}Q^T$, 计算结果为

B 的逆

```
[[ 3.33333333e-01  3.33333333e-01  3.81822733e-18]
[ 5.23809524e-01 -1.90476190e-01 -7.14285714e-02]
[ 1.42857143e-01 -1.42857143e-01  7.14285714e-02]]
```

5

Givens 变换计算 QR 分解代码

```python
import numpy as np

def givens_rotation(A):
    """ 使用 Givens 变换计算矩阵 A 的 QR 分解 """
    m, n = A.shape
    Q = np.eye(m)
    R = A.copy().astype(float)

    for j in range(n):
        for i in range(m - 1, j, -1):  # 从底部向上遍历行
            if R[i, j] != 0:  # 只在非零元素时进行旋转
                a, b = R[j, j], R[i, j]
                r = np.hypot(a, b)  # 计算 sqrt(a^2 + b^2) 避免溢出
                c, s = a / r, -b / r

                # 构造 Givens 旋转矩阵
                G = np.eye(m)
                G[[j, i], [j, i]] = c
                G[j, i], G[i, j] = -s, s

                R = G @ R
                Q = Q @ G.T

    return Q, R

# 测试
A = np.array([[2, 2, 1],
              [0, 2, 2],
              [2, 1, 2]], dtype=float)

Q, R = givens_rotation(A)

print("Q:\n", Q)
print("R:\n", R)
```

输出结果

```
Q:
    [[ 0.70710678  0.23570226 -0.66666667]
    [ 0.          0.94280904  0.33333333]
```

```
    [ 0.70710678 -0.23570226  0.66666667]]
R:
    [[ 2.82842712e+00  2.12132034e+00  2.12132034e+00]
    [ 0.00000000e+00  2.12132034e+00  1.64991582e+00]
    [ 0.00000000e+00 -4.96469267e-17  1.33333333e+00]]
```