In [ ]:

```python
import numpy as np
import scipy
import scipy.linalg
```

Hw 3

3.30

取 α, st $\alpha = -\text{sign}(a_1)\|a\|_2$.

3.33

(a) 可以用A来储存 LU. 由于 L 为 ~~上~~下三角, U为上三角且 L的对角线元为1.

$\Rightarrow$ 无需额外空间.

(b) 可将上三角矩阵储存在原来位置, 如 $A \in \mathbb{R}^{m\times n}$. A的第一列可以用来存 Householder 变换向量的非零部分. 不过须要加一行

$\Rightarrow$ 需 $(m+1)\times n$ 的额外空间.

3.34.

i) LU分解: $r(A) = r(U)$

ii) QR分解: $r(A) = r(R)$

3.1.

(a) $\begin{pmatrix} 1 & 10 \\ 1 & 15 \\ 1 & 20 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 11.6 \\ 11.85 \\ 12.25 \end{pmatrix}$.

由 -① +② $\Rightarrow$ $5x_2 = 0.25$.

由 -② +③ $\Rightarrow$ $5x_2 = 0.4$, 矛盾

(b) (i) $\begin{pmatrix} 1 & 10 \\ 1 & 15 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 11.6 \\ 11.85 \end{pmatrix}$

$\Rightarrow x_1 = 11.1,\ x_2 = 0.05$

(ii) $\begin{pmatrix} 1 & 10 \\ 1 & 20 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 11.6 \\ 12.25 \end{pmatrix} \Rightarrow x_1 = 10.95,\ x_2 = 0.065$

(iii) $\begin{pmatrix} 1 & 15 \\ 1 & 20 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 11.85 \\ 12.25 \end{pmatrix}$

$\Rightarrow x_1 = 10.65,\ x_2 = 0.08.$

(c) 求解 $A^TAx = A^Tb$.

$\Rightarrow x = \begin{pmatrix} 10.925 \\ 0.065 \end{pmatrix}$. $\Rightarrow$ (ii) 比较准确.

3.3. $\begin{pmatrix} 1 & e \\ 2 & e^2 \\ 3 & e^3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$.

$\Rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (A^TA)^{-1}A^Tb \Leftrightarrow \begin{pmatrix} 14 & e+2e^2+3e^3 \\ e+2e^2+3e^3 & e^2+e^4+e^6 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 23 \\ 2e+3e^2+5e^3 \end{pmatrix}$

3.34

$H^T = I - 2\left(\frac{vv^T}{v^Tv}\right)^T = I - 2\frac{v^Tv^T}{v^Tv} = H.$

$H^TH = H^2 = \left(I - 2\frac{vv^T}{v^Tv}\right)^2 = I + 4\frac{v(v^Tv)v^T}{(v^Tv)^2} - 4\frac{vv^T}{v^Tv}$

$= I + 4\frac{vv^T}{v^Tv} - 4\frac{vv^T}{v^Tv} = I.$

⇒ H 对称正定

3.31. $Ax \cong b$ 的最小二乘解 $x = A^+ b = V\Sigma^+ U^T b$.

⇒ $x = V\Sigma^+(u_1^T b, u_2^T b, \cdots, u_m^T b)$

$= (v_1, v_2, \cdots, v_m)(\frac{u_1^T b}{\sigma_1}, \cdots, \frac{u_m^T b}{\sigma_m})$, $\sigma_i \neq 0$.

$= \sum_{\sigma_i \neq 0} \frac{u_i^T b}{\sigma_i} v_i$

(由于 σ_i 为0时，对应广义逆为0).

3.32. (a) $AA^+A = A \Leftrightarrow U\Sigma\Sigma^+\Sigma V^T = U\Sigma V^T$.

由于 $\Sigma\Sigma^+\Sigma = \Sigma$. ⇒ 结论成立.

(b). $\Sigma^+\Sigma\Sigma^+ = \Sigma^+$. ⇒ $A^+AA^+ = A^+$.

(c) 由于 $\Sigma\Sigma^+$ 为对角阵. ⇒ $(\Sigma\Sigma^+)^T = (\Sigma\Sigma^+)$

⇒ 结论成立

(d) 同上，结论成立.

Hw4.

4.13. 设 $Ax_0 = \lambda_{max} x_0$, $\|x_0\| = 1$

⇒ $\|A\| = \sup_{\|x\|=1} \|Ax\| \geq \lambda_{max}$. 即 $\rho(A) \leq \|A\|$

4.19 $Ax = \lambda x$. ⇒ $\lambda^2 x = A^2 x = Ax = \lambda x$

⇒ $\lambda = 1$ 或 0.

4.20. (a) $Ax = \lambda x$, $Ay = \mu y$. ⇒ $y^H A x = \mu y^H x$.

$x^H A y = \lambda x^H y$

T1

Computer problem (in C or C++): Using Gaussian elimination to achieve the LU decomposition with and without a column pivoting; Using the two LU decomposition algorithm to solve linear systems in which the coefficient matrix is (1) general nonsingular matrix; (2) positive definite matrix; (3) diagonally dominant matrix. Compare the numerical accuracy for the two algorithms. The size of the matrices should be greater than 1000.

In [ ]:

```
def Q_i(Q_min, i, j, k):
    if i < k or j < k:
        return float(i == j)
    else:
        return Q_min[i-k][j-k]

def QR_householder(A):
    n = A.shape[0]

    # Set R equal to A, and create Q as a zero matrix of the same size
    R = A.copy()
    Q = np.zeros((n,n))

    # The Householder procedure
    for k in range(n-1):
        I = np.eye(n)
        x = R[k:,k]
        e = I[k:,k]
        alpha = np.sign(x[0]) * np.linalg.norm(x)

        # Using anonymous functions, we create u and v
        u = np.array(list(map(lambda p,q: p + alpha * q, x, e)))
        # print(u)
        norm_u = np.linalg.norm(u)
        v = np.array(list(map(lambda p: p/norm_u, u)))

        # Create the Q minor matrix
        Q_min = np.array([ [float(i==j) - 2.0 * v[i] * v[j] for i in range(n-k)] for j

        # "Pad out" the Q minor matrix with elements from the identity
        Q_t = np.array([[ Q_i(Q_min,i,j,k) for i in range(n)] for j in range(n)])

        if k == 0:
            Q = Q_t
            R = Q_t@A
        else:
            Q = Q_t@Q
            R = Q_t@R

    # Since Q is defined as the product of transposes of Q_t,
    # we need to take the transpose upon returning it
    return Q.T, R
```

In [ ]:

```
def QR_Household_column_pivot(A_x):
    A = A_x
    A_star = A_x
    Q = np.eye(A.shape[0])
    R = np.zeros((A.shape[0],A.shape[1]))
    P = np.eye(A.shape[0])
    for k in range(A.shape[0]):
        argmax_k = np.argmax(np.abs(A[k:,k]))
        P_k = np.eye(A.shape[0])
        P_k[[k,argmax_k+k],:] = P_k[[argmax_k+k,k],:]
        A = P_k@A
        a_k = - np.sign(A[k][k])*np.sqrt(np.sum(A[k:,k]**2))
        v_k = np.reshape(np.concatenate((np.zeros((k)),A[k:,k])),(np.shape(A)[0],1)) -
        beta_k = np.transpose(v_k)@v_k
        if beta_k == 0:
```

```
            continue
        for j in range(k,A.shape[1]):
            gamma_j = np.transpose(v_k)@A[:,j]
            A[:,j] = A[:,j] - np.reshape((2*gamma_j/beta_k)*v_k,np.shape(A)[0])
        H = np.eye(A.shape[0]) - 2 / beta_k * v_k@np.transpose(v_k)
        A_star = P_k@A_star
        A_star = H@A_star
        Q = Q@np.transpose(P_k)
        Q = Q@np.transpose(H)

    return Q , A
```

验证

In [ ]:

```
A = np.array([[12, -51, 4], [6, 167, -68], [-4, 24, -41]])
Q, R = QR_householder(A)
```

In [ ]:

```
print("Q,R分别为：",Q,R)
```

```
Q,R分别为： [[-0.85714286  0.39428571  0.33142857]
 [-0.42857143 -0.90285714 -0.03428571]
 [ 0.28571429 -0.17142857  0.94285714]] [[-1.40000000e+01 -2.10000000e+01  1.40000000e+01]
 [-5.57673565e-16 -1.75000000e+02  7.00000000e+01]
 [-5.08994556e-16 -7.64989650e-16 -3.50000000e+01]]
```

In [ ]:

```
q,r = scipy.linalg.qr(A)
print("Q,R与真实的Q,R对比为：",Q-q,R-r)
```

```
Q,R与真实的Q,R对比为： [[ 2.22044605e-16 -1.66533454e-16 -5.55111512e-17]
 [ 5.55111512e-17  2.22044605e-16 -4.85722573e-17]
 [-5.55111512e-17 -2.77555756e-17  0.00000000e+00]] [[ 1.77635684e-15  0.00000000e+00 -3.55271368e-15]
 [-5.57673565e-16  5.68434189e-14 -2.84217094e-14]
 [-5.08994556e-16 -7.64989650e-16  0.00000000e+00]]
```