Assignment 2

1) Given $A \in \mathbb{C}^{m \times n}$ $(m \geq n)$ show that A^*A is non singular if and ~~if~~ only if A has full rank.

if A has full rank $= n \Rightarrow$ columns are linearly independent and form a basis for its column space.

Therefore A^*A also has full rank $\Rightarrow A^*A$ is non singular.

if A^*A is singular, then

$$(A^*A)x = 0.$$

$$\therefore \quad A^* x = 0.$$

Since A has full rank $\Rightarrow x = 0$.

$\therefore$ A^*A is non singular.

A^*A is non singular, then A has full rank

null $(A^*A) = \phi$.

if x is a non zero vector

$$Ax = 0.$$

$$A^* A x = A^* 0$$

$$(A^*A) x = 0$$

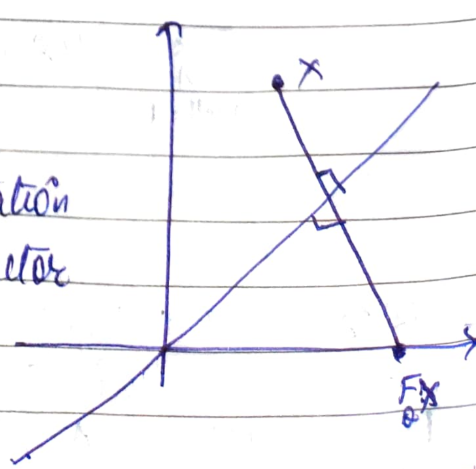
(~~contract~~ ~~cond~~ Contradicts our assumption of A^*A is non singular)

Therefore A must be full rank

3)a $F_\theta = \begin{bmatrix} -c & s \\ s & c \end{bmatrix}$ $\quad J_\theta = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ $\quad \det F_\theta = -1 \quad \det J_\theta = 1$

F_θ is the reflector across the line

Indicate that F flips the orientation of the space means it is a reflector or mirror reflect of a point across a line in 2D space.



Page No.
Date

suppose $v = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$

$F_\theta v = \begin{bmatrix} -c & s \\ s & c \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -cx + sy \\ sx + cy \end{bmatrix}$ (reflects the line defined by the normal vector $\begin{bmatrix} c \\ s \end{bmatrix}$.

The line of reflection is along the eigen vector corresponding to the eigen value $= -1$.

$J_\theta = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ Preserve the Orientation of the space
Perform the rotation by θ.

suppose $v = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$

$J_\theta v = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} cx + sy \\ -sx + cy \end{bmatrix}$

$\theta > 0$ rotation counterclockwise
$\theta < 0$ rotation clockwise.

$A = QR.$ $\quad A \in \mathbb{C}^{m \times n}$

3b) initialize $Q = I$, $R = A$.
$j = 1, \ldots, n$ column.
$i =$ rows from the bottom up.
matrix entry a_{ij} & $a_{i+1j} = 0$

$G_{i,j} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$. $\quad c = \dfrac{a_{ij}}{\sqrt{a_{ij}^2 + a_{i+1j}^2}}$

$s = \dfrac{a_{i+1j}}{\sqrt{a_{ij}^2 + a_{i+1j}^2}}$

update $Q = Q \cdot G_{ij}^*$
$R = G_{ij} \cdot R$

2) $A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$ reduced $A = \hat{Q}\hat{R}$

$a_1 = (1,0,1) = u_1$ $\qquad a_2 = (2,1,0)$

$q_1 = \dfrac{u_1}{\|u_1\|} = \left(\dfrac{1}{\sqrt{2}}, 0, \dfrac{1}{\sqrt{2}}\right)$

$r_{11} = \|u_1\| = \sqrt{2}$.

$v_2 = a_2 - q_1 q_1^* a_2$.

$= \cancel{(2,1,0)} \quad (1,1,-1)$

$q_2 = \dfrac{v_2}{\|v_2\|} = \left(\dfrac{1}{\sqrt{3}}, \dfrac{1}{\sqrt{3}}, \dfrac{-1}{\sqrt{3}}\right)$

$\hat{Q} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{3} \\ 0 & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{3} \end{bmatrix} \in \mathbb{R}^{3\times 2}$.

$\hat{R} = \begin{bmatrix} q_1^* a_1 & q_1^* a_2 \\ 0 & q_2^* a_2 \end{bmatrix} = \begin{bmatrix} \sqrt{2} & \sqrt{2} \\ 0 & \sqrt{3} \end{bmatrix} \in \mathbb{R}^{2\times 2}$

full $A = QR$.

$q_3 \perp q_2 \,\&\, q_1$

$q_1^* q_3 = 1/\sqrt{2}\, x_1 + 0\, x_2 + 1/\sqrt{2}\, x_3$

$q_2^* q_3 = 1/\sqrt{3}\, x_1 + 1/\sqrt{3}\, x_2 - 1/\sqrt{3}\, x_3$

$q_3 = \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix}$ $\qquad Q = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{3} & 1 \\ 0 & 1/\sqrt{3} & -2 \\ 1/\sqrt{2} & -1/\sqrt{3} & -1 \end{bmatrix} \in \mathbb{R}^{2\times 2}$

$R = \begin{bmatrix} \sqrt{2} & \sqrt{2} \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 2}$

Page No.
Date

4) $A \in \mathbb{C}^{m\times n}$ $\text{rank}(A) = n$ $b \in \mathbb{C}^m$

$$\begin{bmatrix} I & A \\ A^* & 0 \end{bmatrix}\begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

$$I\cdot r + A\cdot x = b$$

$A^*\cdot r = 0$ $\rightarrow$ r null space (A^*)

r is orthogonal to the column of A.

$I\cdot r = b - A\cdot x$ $\Longrightarrow$ $(r = b - Ax)$

$$A^*(I\cdot r) = A^*(b - A\cdot x)$$

$$A^* r = A^* b - A^* A\cdot x$$

$$A^* A x = A^* b$$

A = full rank. $\Rightarrow$ A^*A = positive definite

$$x = \underbrace{(A^*A)^{-1}A^*}_{A^+}b$$

$= A^+ b$. which is a unique solution

5) Function "magic" performs computation to reconstruct a matrix from its singular value decomposition (SVD).

→ Given Input matrix A

$$A = U\cdot S\cdot V^*$$

$U \in \mathbb{C}^{m\times m}$ unitary matrix containing left singular value

$S \in \mathbb{C}^{m\times n}$ ~~with~~ diagonal of ≥ 0 $(\sigma_1 \geq, \sigma_2 \geq, \ldots)$

$V \in \mathbb{C}^{n\times n}$ unitary matrix containing right singular value.

Calculation of numerical tolerance:

define a machine epsilon "eps"

Compute "tol" as the product of the maximum matrix dimension the 1st singular value and eps.

determine the rank (r) of the matrix. count the number of singular value in S greater than the computed "tol".

Page No.

Date

Create a degonal matrix S' by taking the reciprocal of the 1st r singular value and truncate the rest.

Use the truncated matrix (U_{tr} S' V_{tr}) to form matrix X

$$X = ~~V_{tr}~~ V_{tr} \cdot S' \cdot U_{tr}^T$$

U_{tr} represent the 1st r column of matrix U.

V_{tr} represent the 1st r column of matrix V.

The result X will be a reconstruction of matrix (A), Preserving only the information Corresponding to the Significant singular value, effectively performing a rank reduction while maintaining the most Critical information.

```python
import copy
import numpy as np

def implicit_qr(A):
    m, n = np.shape(A)
    W = np.zeros((m, n), dtype=complex)
    R = A.copy().astype(complex)

    for k in range(n):
        x = R[k:m, k][:, np.newaxis]
        # Determine the complex sign
        complex_sign_x1 = x[0,0]/np.abs(x[0,0]) if np.abs(x[0,0]) != 0 else 1

        e1 = np.zeros((m-k, 1), dtype=complex)
        e1[0] = 1
        v_k = x + complex_sign_x1 * np.linalg.norm(x) * e1
        v_k = v_k / np.linalg.norm(v_k)

        W[k:m, k] = v_k[:, 0]
        # Apply transformation to R
        R[k:m, k:n] = R[k:m, k:n] - 2 * np.dot(v_k, np.dot(v_k.conj().T, R[k:m, k:n]))
    return W, R
```

```python
def form_q(W):
    m, n = np.shape(W)
    Q = np.eye(m, dtype=complex)

    for k in range(n-1, -1, -1):
        v_k = W[k:m, k][:, np.newaxis]
        Q[k:m, k:m] -= 2 * np.dot(v_k, np.dot(v_k.conj().T, Q[k:m, k:m]))

    return Q


if __name__ == "__main__":
# Test

 A = np.array([[2 + 1j, 4 - 2j, 1 + 0j],
              [1 - 1j, 3 + 3j, 2 - 2j],
              [5 + 0j, 1 - 1j, 3 + 3j],
              [1 + 1j, 2 + 2j, 1 - 1j]], dtype=complex)

 W, R = implicit_qr(A)
 Q = form_q(W)

 print("Q:\n", np.round(Q))
 print("R:\n", np.round(R))
 print("Q @ R (should be close to original A):\n", np.round(Q @ R))
 print("Difference btw original A and QR:\n", np.round(np.abs(Q @ R - A)))
```