Abgabe - Übungsblatt [5]

Angewandte Mathematik: Numerik

[Felix Lehmann] [Markus Menke]

8. Dezember 2020

Aufgabe 1

Here comes your text ...

Aufgabe 2

QR Zerlegung:

$$A = \begin{pmatrix} -2 & 2 & 3 \\ 2 & 3 & 1 \\ 1 & 4 & -2 \end{pmatrix}$$

$$q_1' = a_1 = \begin{pmatrix} -2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 2 \\ 1 \end{pmatrix}$$

$$r_{11} = ||q_1'|| = \sqrt{-2^2 + 2^2 + 1^2} = \sqrt{9} = 3$$

$$q_1 = \frac{1}{||q_1'||} * q_1' = \frac{1}{3} * \begin{pmatrix} -2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} \\ \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}$$

$$r_{12} = q_1^T * a_2 = \begin{pmatrix} -\frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{pmatrix} \times \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix} = 2$$

$$q_2' = a_2 - r_{12} * q_1 = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix} - 2 * \begin{pmatrix} -\frac{2}{3} \\ \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}$$

$$r_{22} = ||q_2'|| = \sqrt{\frac{10}{3}^2 + \frac{5}{3}^2 + \frac{10}{3}^2} = \sqrt{25} = 5$$

$$q_2 = \frac{1}{||q_2'||} * q_2' = 1/5 * \begin{pmatrix} \frac{10}{3} \\ \frac{5}{3} \\ \frac{10}{3} \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ \frac{2}{3} \end{pmatrix}$$

$$r_{13} = q_1^T * a_3 = \begin{pmatrix} -\frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{pmatrix} \times \begin{pmatrix} 3 \\ 1 \\ -2 \end{pmatrix} = -2$$

$$r_{23} = q_2^T * a_3 = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \end{pmatrix} \times \begin{pmatrix} 3 \\ 1 \\ -2 \end{pmatrix} = 1$$

$$q_3' = a_3 - r_{13} * q_1 - r_{23} * q_2 = \begin{pmatrix} 3 \\ 1 \\ -2 \end{pmatrix} + 2 * \begin{pmatrix} -\frac{2}{3} \\ \frac{2}{3} \\ \frac{1}{3} \end{pmatrix} - 1 * \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ \frac{2}{3} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ -2 \end{pmatrix}$$

$$r_{33} = ||q_3'|| = \sqrt{1^2 + 2^2 + (-2)^2} = \sqrt{9} = 3$$

$$q_3 = \frac{1}{||q_3'||} * q_3' = \frac{1}{3} * \begin{pmatrix} 1 \\ 2 \\ -2 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ -\frac{2}{3} \end{pmatrix}$$

$$Q = \begin{pmatrix} q_1 & q_2 & q_3 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \end{pmatrix}$$

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix} = \begin{pmatrix} 3 & 2 & -2 \\ 0 & 5 & 1 \\ 0 & 0 & 3 \end{pmatrix}$$

Aufgabe 3

Aufgabe 4

```python
import numpy as np


def QR(A):
    """Given a matrix A with full column rank this
        function uses the classical
       Gram-Schmidt algorithm to compute a QR
           decomposition. It returns a tuple
       (Q,R) of np.matrix objects with Q having shape
           identical to A and Q*R=A."""
    m, n = A.shape
    A = A.copy()
    Q = np.zeros((m, n))
    R = np.zeros((n, n))

    for k in range(n):
        R[k, k] = np.linalg.norm(A[:, k:k+1].reshape(-1),
            2)
        Q[:, k:k+1] = A[:, k:k+1]/R[k, k]
        R[k:k+1, k+1:n+1] = np.dot(Q[:, k:k+1].T, A[:, k
            +1:n+1])
        A[:, k+1:n+1] = A[:, k+1:n+1] - np.dot(Q[:, k:k
            +1], R[k:k+1, k+1:n+1])

    return np.asmatrix(Q), np.asmatrix(R)


def BackSubstitution(R: np.matrix, y: np.ndarray):
    """Given a square upper triangular matrix R and a
        vector y of same size this
```

```
        function solves R*x=y using backward substitution
            and returns x."""
    m, n = R.shape
    x = np.zeros_like(y)

    for i in range(n):
        x[i] = y[i] / R[i, i]
        y[1:i-1] = y[1:i-1] - R[1:i-1, i] * x[i]

    return x


def LeastSquares(A, b):
    """Given a matrix A and a vector b this function
        solves the least squares
        problem of minimizing |A*x-b| and returns the
            optimal x."""
    Q, R = QR(A)

    # Wenn ich das nachfolgende zurueckgeben soll brauche
        ich doch gar keine BackSubstitution?
    x = np.linalg.inv(R) * Q.H * b

    return x


if(__name__ == "__main__"):
    A = np.random.rand(13, 10) * 1000
    # Try the QR decomposition
    A = np.matrix([
        [1.0, 1.0, 1.0],
        [1.0, 2.0, 3.0],
        [1.0, 4.0, 9.0],
        [1.0, 8.0, 27.0]
    ])
    Q, R = QR(A)
    print("If the following numbers are nearly zero, QR
        seems to be working.")
    print(np.linalg.norm(Q*R-A))
    print(np.linalg.norm(Q.H*Q-np.eye(3)))
    # Try solving a least squares system
    b = np.matrix([1.0, 2.0, 3.0, 4.0]).T
    x = LeastSquares(A, b)
    print("If the following number is nearly zero, least
        squares solving seems to be working.")
    print(np.linalg.norm(x-np.linalg.lstsq(A, b)[0]))
```