Week 5 Hand-in

NumIntro 2019
Department of Computer Science
University of Copenhagen

Casper Lisager Frandsen <fsn483@alumni.ku.dk>

Version 1
Due: October 9th, 08:00

Contents

1. Exercise

(a)

This algorithm has been implemented in the "Q0038.py" file.

```
def crout(A: np.ndarray):
    """
    Computes LU factorization of A.

    :param A:      The matrix.

    :return:       The pair of matrices L & U.
    """
    n = np.shape(A)[0]
    U = np.identity(n)
    L = np.zeros((n, n))
    for i in range(n):
        L[i,i] = (A[i,i] - np.sum(L[i,:] * U[:,i]))/U[i,i]
        for j in range(i,n):
            U[i,j] = (A[i,j] - np.sum(L[i,range(i)]* U[range(i),j]))/L[i,i]
            L[j,i] = (A[j,i] - np.sum(L[j,range(i)] * U[range(i),i]))/U[i,i]

    return L, U
```

Figure 1: crout function

(b)

This algorithm has been implemented in the "Q0038.py" file.

```
def solve_system(A: np.ndarray, b: np.array) -> np.array:
    """
    Solve linear system A x = b, assume A to be a square non-singular matrix.

    :param A:    A square non-singular matrix.
    :param b:    A right hand side vector.
    :return:    Upon return this argument holds the solution for A x = b
    """
    # TODO add your code here
    L, U = crout(A)

    x = forward_substitution(L, b)
    y = back_substitution(U,x)

    return y
```

Figure 2: solve_system function

(c)

The matrix has been defined in the "Q0038.py", and probably has some typo errors due to having to enter it by hand. It has been defined in the code as thus:

```
exMatrix = np.zeros((13,13))

exMatrix[0,0] = -0.7071

exMatrix[0,1] = 0.7071
exMatrix[0,4] = 0.7071
exMatrix[10,6] = 0.7071
exMatrix[10,10] = 0.7071
exMatrix[10,12] = 0.7071

exMatrix[4,0] = 0.6585
exMatrix[7,3] = 0.6585

exMatrix[4,9] = -0.6585

exMatrix[4,4] = 0.7526
exMatrix[7,7] = 0.7526
exMatrix[4,8] = 0.7526
exMatrix[7,11] = 0.7526

exMatrix[3,0]=exMatrix[1,1]=exMatrix[2,2]=exMatrix[6,3]=exMatrix[2,4]=exMatrix[5,5]=exMatr
ix[8,7]=exMatrix[8,8]=exMatrix[9,9]=exMatrix[11,10]=exMatrix[11,11]=exMatrix[12,12]= 1.

exMatrix[3,3]=exMatrix[1,5]=exMatrix[6,6]=exMatrix[5,9]= -1.

exVector = np.array([0., 0., 200., 0., -1000., 0., 0., -500., 4000., 0., -500., 2000.,
0.,])

print(solve_system(exMatrix, exVector))


att.start()
```

Figure 3: The linear system of equations being entered into a matrix

Passing this matrix into the *solve_system* function returns the following vector as the solution:

$$\begin{pmatrix} -2320.09555208 \\ 0. \\ 2520.09555208 \\ -2320.09555208 \\ -2320.09555208 \\ 0. \\ -2320.09555208 \\ 978.62498865 \\ 3021.37501135 \\ 0. \\ 1612.98198965 \\ 387.01801035 \\ 0. \end{pmatrix}$$

2. Exercise

(a)

This algorithm has been implemented in the "Q0016.py" file.

```
def neumann(B: np.ndarray, N: int) -> np.ndarray:
    """
    Computes the Neumann series approximation to the inverse of B.

    :param B:       The matrix.
    :param N:       The number of terms to include in the series

    :return:        An approximation to the inverse matrix if series
                    converge otherwise a zero matrix.
    """
    I = np.identity(np.shape(B)[0])
    A = I - B
    if np.linalg.norm(A) < 1:
        if N > 1:
            return I + np.matmul(A,neumann(B, N-1))
        else:
            return I + A
    else:
        return np.zeros(np.shape(B))
```

Figure 4: Neumann function

(b)

To find the interval J such that the Neumann series applied to a matrix B converges to A^{-1} when $a \in J$, we will first write down A:

$$A = \begin{bmatrix} 0.9 & -0.2 & -0.2 \\ 0.1 & 0.7 & -e \\ 0.1 & -0.3 & 0.5 \end{bmatrix}$$

And we will let $B = I - A$:

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.9 & -0.2 & -0.2 \\ 0.1 & 0.7 & -e \\ 0.1 & -0.3 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.2 & 0.2 \\ -0.1 & 0.3 & e \\ -0.1 & 0.3 & 0.5 \end{bmatrix}$$

We know from Theorem 1 on pg. 198 of the textbook that if $||B||_\infty < 1$ then the Neumann series $\sum_{k=0}^{\infty} B^k$ will converge to A^{-1}. Thus, we have to find an interval for e where this is true.

$$||B||_\infty = max(0.7, |0.2 + e|) < 1 \iff e \in J = (-1.2, 0.8)$$

That is, for any value of $-1.2 < e < 0.8$, the Neumann series applied to B will converge to A^{-1}.

(c)

To compute the condition number of A, we use the following formula:

$$_K(A) = ||A|| \cdot ||A^{-1}||$$

With this, we get the condition number of A to equal 6.407. We use the theorem on pg. 193 of the textbook:

$$\frac{1}{_K(A)} \frac{||r||}{||b||} \leq \frac{||e||}{||x||} \leq_K (A) \frac{||r||}{||b||}$$

Since $||r|| = \mathcal{E}||b||$, we can rewrite to the following:

$$\frac{\mathcal{E}}{_K(A)} \leq \frac{||e||}{||x||} \leq_K (A)\mathcal{E}$$

Since $\frac{||e||}{||x||}$ is the relative error, we can bound it thus:

$$\frac{\mathcal{E}}{6.407} \leq \frac{||e||}{||x||} \leq 6.407\mathcal{E}$$