MA398 Matrix Analysis and Algorithms: Exercise Sheet 1

1. (a) Given the matrix

$$A = \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & -1 \\ 3 & -2 & -9 \end{bmatrix}, \quad b = \begin{bmatrix} 8 \\ 2 \\ 9 \end{bmatrix}$$

Use the Gaussian elimination method described in the lecture notes to solve the system $Ax = b$. Show each step of your work.

Answer: Given system of equations:

$$\begin{aligned} x_1 - x_2 + x_3 &= 8, \\ 2x_1 + 3x_2 - x_3 &= 2, \\ 3x_1 - 2x_2 - 9x_3 &= 9. \end{aligned}$$

Starting with these equations, we can perform the following row operations: $-3 \times R_1 + R_2$ and $-2 \times R_1 + R_3$ to get:

$$\begin{aligned} x_1 - x_2 + x_3 &= 8, \\ 0x_1 + x_2 - 12x_3 &= -15, \\ 0x_1 + 5x_2 - 3x_3 &= -18. \end{aligned}$$

Then, to get rid of x_2 in the third equation, we can subtract 5 times the second row from the third row:

$$\begin{aligned} x_1 - x_2 + x_3 &= 8, \\ 0x_1 + x_2 - 12x_3 &= -15, \\ 0x_1 + 0x_2 + 57x_3 &= 57. \end{aligned}$$

From the third equation we get $x_3 = 1$. Substituting $x_3 = 1$ into the second equation gives $x_2 = -3$. Using these values in the first equation gives $x_1 = 4$.

So the solution to the system of equations is $x = (4, -3, 1)$.

(b) Implement a Python function called **gaussian_elimination(A, b)** that takes as input a numpy array A and a vector b and outputs the solution vector x. Test your function on the matrix and vector given in part (a).

2. (Forward substitution) Formulate the algorithm **FS** (forward substitution) to solve the system $Lx = b$ where $L \in \mathbb{C}^{n \times n}$ is unit lower triangular and $b \in \mathbb{C}^n$, and show that the algorithm computes the correct result, with a computational cost of n^2.

3. (LU decomposition)

(a) Find the LU decomposition of the matrix

$$A = \begin{bmatrix} 2 & -1 & 3 \\ 4 & 2 & 1 \\ -6 & -1 & 2 \end{bmatrix},$$

and use it to solve $Ax = b$ with $b = (7, 8, -3)$.

Answer: The LU decomposition of a matrix is a process where we factorize the original matrix A into a product of a lower triangular matrix L and an upper triangular matrix U.

The LU factorization is given by:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & -1 & 3 \\ 0 & 4 & -5 \\ 0 & 0 & 6 \end{bmatrix}.$$

Now that we have the LU decomposition of A, we can solve $Ax = b$ as follows:
First, we solve $Ly = b$ for y:

$$Ly = b$$

where $b = (7, 8, -3)$.
Then, we solve $Ux = y$ for x:

$$Ux = y$$

We can solve $Ly = b$ as follows:
Here,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & 1 & 1 \end{bmatrix}, \quad b = \begin{pmatrix} 7 \\ 8 \\ -3 \end{pmatrix}.$$

Solving $Ly = b$ yields:

$$\begin{aligned} y_1 &= 7, \\ 2y_1 + y_2 &= 8, \\ -3y_1 + y_2 + y_3 &= -3. \end{aligned}$$

From the above system, we can find $y = (7, -6, -6)^T$.
Next, we use y to solve $Ux = y$:

$$U = \begin{bmatrix} 2 & -1 & 3 \\ 0 & 4 & -5 \\ 0 & 0 & -1 \end{bmatrix}, \quad y = \begin{pmatrix} 7 \\ -6 \\ -6 \end{pmatrix}.$$

Solving $Ux = y$ yields:

$$\begin{aligned} 2x_1 - x_2 + 3x_3 &= 7, \\ 4x_2 - 5x_3 &= -6, \\ -x_3 &= -6. \end{aligned}$$

From the above system, we can find $x = (1, 1, 2)^T$, which is the solution to the original system $Ax = b$.

(b) Implement a Python function called **lu_factorization(A)** that takes as input a numpy array A and outputs the lower triangular matrix L and the upper triangular matrix U. Test your function on the matrix given in part (a).

(c) Show that the multiplication of your L and U matrices gives back the original matrix A.

4. (Operation count) How many operations (divisions and multiplications) are necessary to perform an LU decomposition without pivoting?

5. (Diagonal dominance) A matrix $A = (a_{ij})_{i,j=1}^n \in \mathbb{C}^{n\times n}$ is called strictly diagonal dominant if
$$|a_{ii}| > \sum_{j\neq i} |a_{ij}| \quad \text{for all } i \in \{1,\dots,n\}.$$
Show that such a matrix is invertible and its LU factorisation exists.
For this purpose, show that the remaining matrix $(u_{ij}^{(k)})_{i,j=k+1}^n$ after step k of the Gaussian elimination without pivoting still is strictly diagonal dominant.

Answer: Without loss of generality, we show the assertion for the first step ($k=1$) and for the second row. We will need that
$$\sum_{i\neq 2} |u_{2i}^{(0)}| < |u_{22}^{(0)}| \quad \Rightarrow \quad \sum_{i=3}^n |u_{2i}^{(0)}| < |u_{22}^{(0)}| - |u_{21}^{(0)}|$$
and
$$\sum_{i\neq 1} |u_{1i}^{(0)}| < |u_{11}^{(0)}| \quad \Rightarrow \quad |u_{12}^{(0)}| + \sum_{i=3} |u_{1i}^{(0)}| < |u_{11}^{(0)}| \quad \text{divide by } |u_{11}^{(0)}|$$
$$\Rightarrow \frac{1}{|u_{11}^{(0)}|} \sum_{i=3}^n |u_{1i}^{(0)}| < 1 - \frac{|u_{12}^{(0)}|}{|u_{11}^{(0)}|}.$$
Recall that
$$u_{2i}^{(1)} = u_{2i}^{(0)} - \frac{u_{21}^{(0)}}{u_{11}^{(0)}} u_{1i}^{(0)}, \quad i = 2,\dots,n.$$
Using this, we infer
$$\begin{aligned}
\sum_{i=3}^n |u_{2i}^{(1)}| &= \sum_{i=3}^n \left| u_{2i}^{(0)} - \frac{u_{21}^{(0)}}{u_{11}^{(0)}} u_{1i}^{(0)} \right| \\
&\le \sum_{i=3}^n |u_{2i}^{(0)}| + \sum_{i=3}^n \frac{|u_{21}^{(0)}|}{|u_{11}^{(0)}|} |u_{1i}^{(0)}| \\
&< |u_{22}^{(0)}| - |u_{21}^{(0)}| + |u_{21}^{(0)}| \frac{1}{|u_{11}^{(0)}|} \sum_{i=3}^n |u_{1i}^{(0)}| \\
&< |u_{22}^{(0)}| - |u_{21}^{(0)}| + |u_{21}^{(0)}| \Big(1 - \frac{|u_{12}^{(0)}|}{|u_{11}^{(0)}|} \Big) \\
&= |u_{22}^{(0)}| - \frac{|u_{21}^{(0)}|}{|u_{11}^{(0)}|} |u_{12}^{(0)}| \\
&\le \left| u_{22}^{(0)} - \frac{u_{21}^{(0)}}{u_{11}^{(0)}} u_{12}^{(0)} \right| \\
&= |u_{22}^{(1)}|
\end{aligned}$$
which was to be shown.
Since the Gaussian elimination and the LU factorisation is possible all submatrices of A are regular, in particular $A = A_n$ itself is invertible.

6. (a) Let $A = (a_{ij})_{i,j=1}^n \in \mathbb{C}^{n\times n}$ be a matrix of bandwidth $w \in \{0,\dots,n-1\}$, i.e.,
$$a_{ij} = 0 \quad \text{if } |j-i| > w.$$
Give an example of a 4×4 matrix of bandwidth $w=2$ but not $w=1$ which fulfils the strong row sum criterion (also known as strict diagonal dominance).

Answer: Example:

$$A = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix}$$

(b) Assume that the LU factorisation of a matrix $A \in \mathbb{C}^{n\times n}$ of bandwidth $w = 1$ can be computed with the algorithm LU (without pivoting!). Show that then the computed matrices L and U are of bandwidth $w = 1$, too.

Answer: By induction. Assume that $U^{(k-1)}$ and $L^{(k-1)}$ after step $k-1$ have bandwidth $w = 1$. Then $u_{ik}^{(k-1)} = 0$ if $i > k+1$ which yields that $l_{ik} = 0$ (if $i > k+1$). But this means that $L^{(k)}$ will have bandwidth $w = 1$. Moreover, only the row $i = k+1$ (if $i < n$) of $U^{(k-1)}$ may involve changes when updating to $U^{(k)}$.
From this row $i = k+1$ the multiple l_{ik} of row k is subtracted. The bandwidth assumption on $U^{(k-1)}$ implies that $u_{kj}^{(k-1)} = 0$ if $j > k+1$. Therefore, only the entries $u_{ij}^{(k-1)}$ with $j = k, \dots, \min(k+1, n)$ may involve changes. But since $i = k+1$ we have for these entries that $|i-j| <= w = 1$. As a consequence, if $|i-j| > w = 1$ then $u_{ij}^{(k)} = u_{ij}^{(k-1)} = 0$ so that also $U^{(k)}$ will have bandwidth $w = 1$.

(c) Formulate a specialised version of the algorithm LU for band matrices of bandwidth $w = 1$ where only the necessary operations are carried out. Ensure and check that the number of elementary executable operations is $O(n)$ as $n \to \infty$.

Answer: Cf. algorithm 1. Only the loops for i and j had to be adapted. In every step $k \in \{1, \dots, n-1\}$ we have to perform at most one division to compute the $l_{k+1,k}$, and in order to update the $u_{k+1,j}$ we need at most one multiplication and one subtraction. Hence, the cost for step k is at most three operations. Altogether therefore

$$C_{LUB}(n) \le \sum_{k=1}^{n-1} 3 = 3(n-1) = O(n) \quad \text{as } n \to \infty.$$

Algorithm 1 LU for banded matrices

input: $A \in \mathbb{C}^{n\times n}$ of bandwidth w with $\det(A_j) \neq 0$ for $j = 0, \dots, n$.
output: $L, U \in \mathbb{C}^{n\times n}$ where LU is the LU factorisation of A.

```
L = I, U = A
for k = 1, ..., n-1 do
  l_{k+1,k} = u_{k+1,k} / u_{k,k}
  u_{k+1,k} = 0
  u_{k+1,k+1} = u_{k+1,k+1} - l_{k+1,k} u_{k,k+1}
end for
```