Numerical Analysis (089180)
prof. Simona Perotto
simona.perotto@polimi.it

A.Y. 2022-2023
Luca Liverotti
luca.liverotti@polimi.it

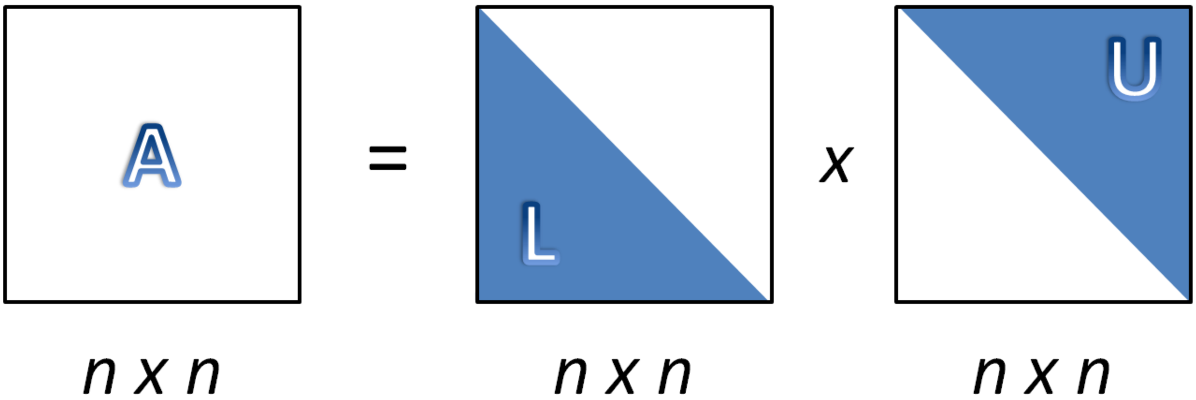
Lab 4 – Solutions

October 14, 2022

This lab deals with the numerical resolution of a linear system

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b}, \quad \text{for } \mathbb{A} \in \mathbb{R}^{n\times n}, \boldsymbol{b} \in \mathbb{R}^n.$$

1 Backward and forward substitution methods



Let $\mathbb{A} \in \mathbb{R}^{n\times n}$. We assume to have LU factorization of $\mathbb{A}$, $\mathbb{A} = \mathbb{L}\mathbb{U}$, with $\mathbb{L}$ **lower triangular** and $\mathbb{U}$ **upper triangular** matrix.

How to use $\mathbb{L}$ and $\mathbb{U}$ to solve thelinear system $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

$$\left.\begin{array}{l} \mathbb{A}\boldsymbol{x} = \boldsymbol{b} \\ \mathbb{A} = \mathbb{L}\mathbb{U} \end{array}\right\} \longrightarrow \mathbb{L}\mathbb{U}\boldsymbol{x} = \boldsymbol{b} \longrightarrow \left\{\begin{array}{l} \mathbb{L}\boldsymbol{y} = \boldsymbol{b} \\ \mathbb{U}\boldsymbol{x} = \boldsymbol{y} \end{array}\right.$$

Forward substitution

$$\begin{array}{cccrc} L_{1,1}y_1 & & & = & b_1 \\ L_{2,1}y_1+ & L_{2,2}y_2 & & = & b_2 \\ \vdots & \vdots\ddots & & & \vdots \\ L_{m,1}y_1+ & L_{m,2}y_2 + \cdots + & L_{m,m}y_m & = & b_m \end{array}$$

$$
\begin{aligned}
y_1 &= \frac{b_1}{L_{1,1}}, \\
y_2 &= \frac{b_2 - L_{2,1} y_1}{L_{2,2}}, \\
&\vdots \\
y_m &= \frac{b_m - \sum_{i=1}^{m-1} L_{m,i} y_i}{L_{m,m}}.
\end{aligned}
$$

Backward substitution

$$
\begin{array}{cccc}
U_{1,1}x_1 + \cdots + & U_{1,m-1}x_{m-1} + & U_{1,m}x_m = & y_1 \\
& \vdots & \vdots & \vdots \\
& U_{m-1,m-1}x_{m-1} + & U_{m-1,m}x_m = & y_{m-1} \\
& & U_{m,m}x_m = & y_m
\end{array}
$$

$$
\begin{aligned}
x_m &= \frac{y_m}{U_{m,m}} \\
x_{m-1} &= \frac{y_{m-1} - U_{m-1,m} x_m}{U_{m-1,m-1}}, \\
&\vdots \\
x_1 &= \frac{y_1 - \sum_{j=2}^{m} U_{1,j} x_j}{U_{1,1}}.
\end{aligned}
$$

Exercise 4.1. a. Write a function which implements the backward substitution method for solving a generic upper triangular system. Apply such a function to solve the linear system $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$ with

$$
\mathbb{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.
$$

b. Write a function to implement the forward substitution method to solve lower triangular systems. Apply this function to the system $\mathbb{A}^T\boldsymbol{x} = \boldsymbol{b}$. with $\mathbb{A}$ and $\boldsymbol{b}$ defined as above.

c. Use the backward and the forward algorithms to solve the system $\mathbb{A}^T\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$.

Solution Exercise 4.1.

backward_substitution.m

```
function x = backward_substitution(A,b)
  %BACKWARD_SUBSTITUTION Solve an upper triangular system using backward
  %substitution method.
  %  x = backward_substitution(A,b)
  %
  %  Inputs : A = system coefficient matrix
  %          b = right-hand side vector
  %  Outputs : x = solution vector

  n = length(b);
  x = 0*b;
  x(n) = b(n) / A(n,n);
```

```
  for (k = n-1:-1:1)
    x(k) = (b(k)-A(k,k+1:n)*x(k+1:n)) / A(k,k);
  end

end
```

forward_substitution.m

```
function x = forward_substitution(A,b)
  %FORWARD_SUBSTITUTION Solve an lower triangular system using forward
  %substitution method.
  %  x = forward_substitution(A,b)
  %
  %  Inputs : A = system coefficient matrix
  %           b = right-hand side vector
  %  Outputs : x = solution vector

  n = length(b);
  x = 0*b;
  x(1) = b(1) / A(1,1);

  for i=2:n
    x(i) = (b(i) - A(i, 1:i-1) * x(1:i-1)) / A(i,i);
  end

end
```

ex_4_1.m

```
%edit backward_substitution

A = [1 2 3 4;
     0 1 2 3;
     0 0 1 2;
     0 0 0 1];
b = [1 1 1 1]'

x = backward_substitution(A, b)

% Check with builtin \ operator of MATLAB
A \ b

%edit forward_substitution

x = forward_substitution(A', b)

% Check with builtin \ operator of MATLAB
A' \ b

y = forward_substitution(A', b);
x = backward_substitution(A, y)

% Check with builtin \ operator of MATLAB
(A'*A) \ b
```

2 LU decomposition

Let $\mathbb{A} \in \mathbb{R}^{n\times n}$: the LU decomposition of $\mathbb{A}$ allows to write

$$\mathbb{A} = \mathbb{L}\mathbb{U}$$

with $\mathbb{L}$ **lower triangular** matrix s.t. $l_{ii} = 1,\ \forall i = 1 \ldots n$, and $\mathbb{U}$ **upper triangular** matrix.

The existence and the uniqueness of such decomposition is related to the principal submatrices of order i of the matrix $\mathbb{A}$, with $i = 1, \cdots, n-1$, which are demanded to be non singular (necessary and sufficient condition). As an alternative, the sufficient condition holds:

- $\mathbb{A}$ is a strictly **diagonally dominant** by row OR by column matrix
- $\mathbb{A}$ is a **symmetric positive definite** matrix.

LU decomposition in MATLAB/Octave

```
>> help lu
lu   lu factorization
[L,U,P] = lu(A)
returns unit lower triangular matrix L, upper triangular matrix U, and permutation
matrix P so that P*A = L*U.

[L,U] = lu(A)
stores an upper triangular matrix in U and a "psychologically lower triangular matrix
" (i.e. the product of a lower triangular and a permutation matrix) in L, so that A =
 L*U, A can be rectangular.
```

LU decomposition with no pivoting

$$\left.\begin{array}{l}\mathbb{A}\boldsymbol{x}=\boldsymbol{b}\\ \mathbb{A}=\mathbb{L}\mathbb{U}\end{array}\right\}\longrightarrow \mathbb{L}\mathbb{U}\boldsymbol{x}=\boldsymbol{b}\longrightarrow\left\{\begin{array}{l}\mathbb{L}\boldsymbol{y}=\boldsymbol{b}\\ \mathbb{U}\boldsymbol{x}=\boldsymbol{y}\end{array}\right.$$

This approach is ideal to solve multiple systems $\mathbb{A}\boldsymbol{x}=\boldsymbol{b}_i$ with different right hand-sides $\boldsymbol{b}_i$, and all sharing the same coefficient matrix. Actually it sufficies to compute the decomposition $\mathbb{L}\mathbb{U}$ just **once**. Computational cost: $\mathcal{O}(n^3)$.

Exercise 4.2. Consider the system $\mathbb{A}\boldsymbol{x}=\boldsymbol{b}$ with

$$\mathbb{A}=\begin{bmatrix}2&10&4&0\\1&0&2&2\\1&4&0&2\\1&2&1&1\end{bmatrix}\qquad\text{and}\qquad\boldsymbol{b}=\begin{bmatrix}10\\1\\3\\3\end{bmatrix}.$$

a. Compute the solution of the system $\mathbb{A}\boldsymbol{x}=\boldsymbol{b}$ using the $\mathbb{L}\mathbb{U}$ decomposition of matrix $\mathbb{A}$.

b. Find the determinant of $\mathbb{A}$ without using the command `det`.

Solution Exercise 4.2.

ex_4_2.m

```
clc
clear all
close all

A = [2 10 4 0;
    1 0 2 2;
    1 4 0 2;
    1 2 1 1];
b = [10 1 3 3]';

[L, U] = lu(A);
% In this case it is not essential to include the matrix P among the output
% variables in order to have L in the form of a lower triangular matrix
% with L_ii = 1 for each i.

y = L \ b; %forward
x = U \ y %backward

% Alternatively, you could have used the functions for the forward and
% backward substitutions that you defined in Lab4.

% Check
A \ b
```

```
% Thanks to the Binet-Cauchy formula, we have, for generic square matrices,
% det(A*B) = det(A)*det(B).
% In our case, since A = L*U, det(A) = det(L*U) = det(L)*det(U).
% However L_ii = 1 for each i, so det(L) = 1
% => det(A) = det(U).

detA = prod(diag(U))

% Check with the command det
det(A)
```

Exercise 4.3. Let us consider the matrix $\mathbb{A} = \begin{bmatrix} 50 & 1 & 3 \\ 1 & 6 & 0 \\ 3 & 0 & 1 \end{bmatrix}$.

1. Apply the Matlab® command `lu` and compute the LU factorization of the matrix $\mathbb{A}$.
2. Solve the system $\mathbb{A}\mathbf{x} = \mathbf{b}$. Choose the vector $\mathbf{b}$, such that the solution of the system is $\mathbf{x}_{ex} = [1,\, 1,\, 1]^T$.
3. Compute the solution of the system $\mathbb{A}\mathbf{x} = \mathbf{b}$, by using the backward and forward substitution the functions previously implemented.

Solution Exercise 4.3.

ex_4_3.m

```
clc
clear all
close all

A = [50 1 3;
   1 6 0;
   3 0 1];

[L, U] = lu(A);

% Let us check if the matrices L and U are as we expect them to be.

L
U

xex = [1; 1; 1];
b = A*xex;

y = forward_substitution(L, b);
x = backward_substitution(U, y);

% Finally, let's check if x is equal to xex:

x
```

Inverse Matrix
We can define the inverse of a squared matrix $\mathbb{A} \in \mathbb{R}^{n\times n}$ as the matrix $\mathbb{X} = \mathbb{A}^{-1} \in \mathbb{R}^{n\times n}$ such that $\mathbb{A}\mathbb{X} = \mathbb{X}\mathbb{A} = \mathbb{I}$. It is possible to determine $\mathbb{A}^{-1}$ by solving the following n linear systems:

$$\mathbb{A}\mathbf{v}_i = \mathbf{e}_i \;, \quad i = 1, \cdots, n \;,$$

where $\mathbf{e}_i$ denote the consecutive columns of the matrix $\mathbb{I}$ (i.e the vectors of the standard basis of $\mathbb{R}^n$). Thus it turns out that

$$\mathbb{A}^{-1} = [\mathbf{v}_1|\mathbf{v}_2|...|\mathbf{v}_n]$$

Exercise 4.4.

1. Write a function `[InvA]=MyInv(A)` that computes the inverse `InvA` of a generic square matrix `A`.

2. Use the function `MyInv` to compute the inverse of the matrix $\mathbb{A}$ in the previous exercise. Compare the result with the output provided by Matlab with command `inv`.

Solution Exercise 4.4.

ex_4_4.m

```
clc
clear all
close all

A = [50 1 3;
   1 6 0;
   3 0 1];

inverseA = MyInv(A)

% Check

inv(A)
```

MyInv.m

```
function [InvA] = MyInv(A)

   [L,U] = lu(A);
   %s = size(A);
   %n = s(1);
   n = length(A);

   for k = 1:n

      e = zeros(n,1);
      e(k) = 1;

      y = forward(L,e);
      InvA(:,k) = backward(U,y);

   end

end
```