Linear systems

Laura Grigori

EPFL and PSI

slides based on lecture notes/slides from L. Dede/S. Deparis

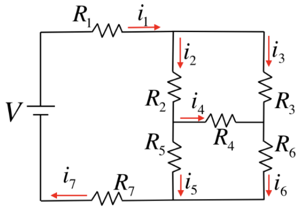
November 6/13, 2024

Plan

Plan

Examples and motivation

Find the currents i_j for $j = 1, \ldots, n$ through the circuit, for $n = 7$, given the tension V and the resistances R_j of the circuit.



The problem is defined by the following equations:

- Balance of the tensions:

$$\begin{aligned} V &= V_1 + V_2 + V_5 + V_7, \\ V_3 &= V_2 + V_4, \\ V_5 &= V_4 + V_6. \end{aligned}$$

- Kirchhoff's laws:

$$\begin{aligned} i_1 &= i_2 + i_3, \\ i_2 &= i_4 + i_5, \\ i_6 &= i_3 + i_4, \\ i_7 &= i_5 + i_6. \end{aligned}$$

Obtained linear system

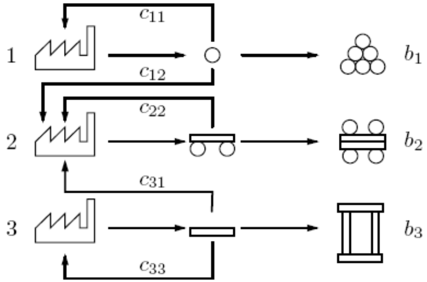
Considering the constitutive equations:

$$V_j = R_j i_j \quad \text{for all } j = 1, \dots, n.$$

we obtain the linear system:

$$\begin{bmatrix} R_1 & R_2 & 0 & 0 & R_5 & 0 & R_7 \\ 0 & R_2 & -R_3 & R_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_4 & -R_5 & R_6 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ i_7 \end{bmatrix} = \begin{bmatrix} V \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example[Economy/Logistic] We want to determine the situation of equilibrium between demand and offer of certain goods. Let us consider that $m >= n$ factories produce n different products. They have to adapt their productions to the internal demand (i.e. the goods needed as input by the other factories) as well as to the external demand, from the consumers.
$x_i,\ i = 1, \ldots, n$ is the total number of goods made by the factory i,
$b_i,\ i = 1, \ldots, n$ is the corresponding demand from the market and
c_{ij} the amount produced by the factory i needed for the factory j to make one unit of product.



Interaction scheme between 3 factories and the market

Example If we suppose that the relation between the different products is linear, the equlibrium is reached when the vector $\mathbf{x} = [x_1, \dots, x_n]^T$ satisfies

$$\mathbf{x} = C\mathbf{x} + \mathbf{b},$$

where $C = (c_{ij})$ and $\mathbf{b} = [b_1, \dots, b_n]^T$. Consequently, the total production $\mathbf{x}$ is solution of the linear system :

$$A\mathbf{x} = \mathbf{b}, \quad \text{where } A = I - C.$$

Plan

Formulation of the problem

We call linear system of order n (n positive integer), an expression of the form

$$A\mathbf{x} = \mathbf{b},$$

where $A = (a_{ij})$ is a given matrix of size $n \times n$, $\mathbf{b} = (b_j)$ is a given vector and $\mathbf{x} = (x_j)$ is the unknown vector of the system. The previous relation is equivalent to the n equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \ i = 1, \ldots, n.$$

The matrix A is called *non-singular* if $\det(A) \neq 0$; the solution $\mathbf{x}$ will be unique (for any given vector $\mathbf{b}$) if and only if the matrix associated to the linear system is non-singular.

In theory, if A is non-singular, the solution is given by the *Cramer's rule*:

$$x_i = \frac{\det(B_i)}{\det(A)},\ i = 1, \dots, n,$$

where B_i is the matrix obtained by substituting the i-th column of A by the vector $\mathbf{b}$:

$$B_i = \left[\begin{array}{ccccc} a_{11} & \dots & b_1 & \dots & a_{1n} \\ a_{21} & \dots & b_2 & \dots & a_{2n} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & \dots & b_n & \dots & a_{nn} \end{array}\right]$$
$$\begin{array}{c} \uparrow \\ i \end{array}$$

Unfortunately, the application of this rule is unacceptable for the practical solution of systems because the computational cost is of the order of $(n+1)!$ *floating point operations* (*flops*). In fact, every determinant requires $n!$ *flops*.

For example, the following table gives the time required by different computers to solve a linear system using the Cramer rule (o.r. stands for "out of reach"):

	Number of flops of the computer				
n	10^9 (Giga)	10^{10}	10^{11}	10^{12} (Tera)	10^{15} (Peta)
10	10^{-1} sec	10^{-2} sec	10^{-3} sec	10^{-4} sec	negligible
15	17 hours	1.74 hours	10.46 min	1 min	$6\ 10^{-2}$ sec
20	4860 years	486 years	48.6 years	4.86 years	1.7 days
25	o.r.	o.r.	o.r.	o.r.	38365 years

Alternative algorithms have to be developed. In the following sections, several methods will be analysed.

Triangular systems

A matrix $U = (u_{ij})$ is *upper triangular* if

$$u_{ij} = 0 \quad \forall i, j \ : \ 1 \leq j < i \leq n$$

and a matrix $L = (l_{ij})$ is *lower triangular* if

$$l_{ij} = 0 \quad \forall i, j \ : \ 1 \leq i < j \leq n.$$

Respectively, the system to be solved is called *upper or lower triangular system*.

Remark: If a matrix A is non singular and triangular, knowing that

$$\det(A) = \prod_{i=1}^{n} \lambda_i(A) = \prod_{i=1}^{n} a_{ii}$$

($\lambda_i(A)$ being the i-th eigenvalue of A), we can deduce that $a_{ii} \neq 0$, for all $i = 1, \ldots, n$.

If L is lower triangular and non-singular, the linear system $L\mathbf{y} = \mathbf{b}$ corresponds to

$$\left\{ \begin{array}{lcl} l_{11}y_1 & = & b_1 \\ l_{21}y_1 + l_{22}y_2 & = & b_2 \\ \vdots & & \\ l_{n1}y_1 + l_{n2}y_2 + \ldots + l_{nn}y_n & = & b_n \end{array} \right.$$

Thus:

$$\boxed{y_1 = b_1/l_{11}}, \qquad [\ 1 \text{ operation}]$$

and for $i = 2, 3, \ldots, n$

$$\boxed{y_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right).} \qquad [\ 1 + 2(i-1) \text{ operations}]$$

This algorithm is called *forward substitutions algorithm.*

The forward substitutions algorithm requires n^2 operations, where n is the size of the system:

$$\begin{aligned} 1 + \sum_{i=2}^{n} (1 + 2(i-1)) &= 1 + (n-1) + 2\sum_{i=2}^{n}(i-1) \\ &= 1 + (n-1) + 2\frac{n(n-1)}{2} \\ &= n^2. \end{aligned}$$

If U is upper triangular and non-singular, the system $U\mathbf{x} = \mathbf{y}$ is:

$$\begin{cases} u_{11}x_1 + \ldots + u_{1,n-1}x_{n-1} + u_{1n}x_n & = & y_1 \\ & \vdots & \\ u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n & = & y_{n-1} \\ u_{nn}x_n & = & y_n \end{cases}$$

Thus:

$$\boxed{x_n = y_n/u_{nn}},$$

and for $i = n-1, n-2, \ldots, 2, 1$

$$\boxed{x_i = \frac{1}{u_{ii}}\left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right)}.$$

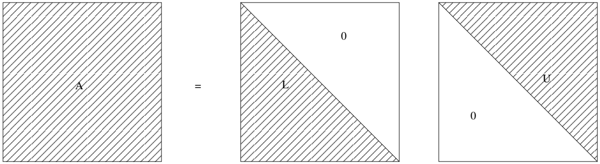
This algorithm is called *backward substitutions algorithm*. The cost is, once again, n^2 operations.

The LU factorization method

(Sec. 6.2.2 of the book)

Let $A = (a_{ij})$ be a non-singular $n \times n$ matrix. Assume that there exist a matrix $U = (u_{ij})$, upper triangular and a matrix $L = (l_{ij})$, lower triangular such that

$$A = LU. \tag{1}$$



We call (1) an *LU factorization of A*.

If we know the LU factorization of A, solving the system $A\mathbf{x} = \mathbf{b}$ is equivalent to solving two systems defined by *triangular* matrices. Indeed,

$$\boxed{A\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad LU\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad \begin{cases} L\mathbf{y} = \mathbf{b}, \\ U\mathbf{x} = \mathbf{y}. \end{cases}}$$

We can easily calculate the solutions of both systems:

- first, we use the forward substitutions algorithm to solve $L\mathbf{y} = \mathbf{b}$ (order n^2 *flops*);
- then, we use the backward substitutions algorithm to solve $U\mathbf{x} = \mathbf{y}$ (order n^2 *flops*).

It is required to find first (if possible) the matrices L and U (what requires a number of operations of the order $\frac{2n^3}{3}$ *flops*).

Example

Let's try to find an LU factorization in the case case where the size of the matrix A is $n = 2$. We can write the equation (1) as

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix},$$

Or equivalently:

$$\begin{array}{llll} (a) & l_{11}u_{11} = a_{11}, & (b) & l_{11}u_{12} = a_{12}, \\ (c) & l_{21}u_{11} = a_{21}, & (d) & l_{21}u_{12} + l_{22}u_{22} = a_{22}. \end{array}$$

We have then a system (non-linear) with 4 equations and 6 unknowns; in order to have the same number of equations and unknowns, we fix the diagonal of L by taking $l_{11} = l_{22} = 1$. Consequently, from (a) and (b) we have $u_{11} = a_{11}$ and $u_{12} = a_{12}$; finally, if we assume $a_{11} \neq 0$, we obtain $l_{21} = a_{12}/a_{11}$ and $u_{22} = a_{22} - l_{21}u_{12} = a_{22} - a_{21}a_{12}/a_{11}$ using the equations (c) and (d).

To determine an LU factorization of the matrix A of any size n, we apply the following method.

1. The elements of L and U satisfy the non-linear system

$$\sum_{r=1}^{\min(i,j)} l_{ir} u_{rj} = a_{ij}, \quad i, j = 1, \ldots, n; \tag{2}$$

2. The system (2) has n^2 equations and $n^2 + n$ unknowns. We can wipe out n unknowns if we set the n diagonal elements of L equal to 1:

$$l_{ii} = 1, \quad i = 1, \ldots, n.$$

   We will see that in this case there exist an algorithm (*Gauss elimination*), allowing us to efficiently compute the factors L and U.

Plan

The Gauss elimination method

The Gauss elimination method transforms the system

$$A\mathbf{x} = \mathbf{b}$$

in an equivalent system (i.e. with the same solution) of the form:

$$U\mathbf{x} = \widehat{\mathbf{b}},$$

where U is an upper triangular matrix and $\widehat{\mathbf{b}}$ is a properly modified second member.

This system can be solved by a backward substitutions method.

In the transformation, we essentially use the proprierty that says that we don't change the solution of the system if we add to a given equation a linear combination of other equations.

Let us consider an invertible matrix $A \in \mathbb{R}^{n\times n}$ in which the diagonal element a_{11} is assumed to be non-zero. we set $A^{(1)} = A$ and $\mathbf{b}^{(1)} = \mathbf{b}$. We introduce the *multiplier*

$$l_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad i = 2, 3, \ldots, n, \qquad A^{(1)} = \begin{bmatrix} a_{11}^{(1)} & \ldots & a_{1j}^{(1)} & \ldots & a_{1n}^{(1)} \\ \vdots & & \vdots & & \vdots \\ a_{i1}^{(1)} & \ldots & a_{ij}^{(1)} & \ldots & a_{in}^{(1)} \\ \vdots & & \vdots & & \vdots \\ a_{n1}^{(1)} & \ldots & a_{nj}^{(1)} & \ldots & a_{nn}^{(1)} \end{bmatrix}$$

where the $a_{ij}^{(1)}$ represent the elements of $A^{(1)}$. Example:

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 4 & 8 & 10 \\ 7 & 8 & 9 \end{bmatrix} \quad \Longrightarrow \quad l_{21} = \frac{4}{2}, l_{31} = \frac{7}{2}.$$

The unknown x_1 can be removed from the rows $i = 2, \ldots, n$ by substracting l_{i1} times the first row and doing the same at the right-hand side.

Let us define

$$a_{ij}^{(2)} = a_{ij}^{(1)} - l_{i1} a_{1j}^{(1)}, \quad i, j = 2, \dots, n,$$

$$b_i^{(2)} = b_i^{(1)} - l_{i1} b_1^{(1)}, \quad i = 2, \dots, n,$$

where the $b_i^{(1)}$ are the components of $\mathbf{b}^{(1)}$ and we get a new system of the form

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix},$$

which will be written as $A^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$ and that is equivalent to the system we had at the beginning.

Once again we can transform this system by removing the unknown x_2 from the rows $3, \ldots, n$. By repeating this step we obtain a finite series of systems

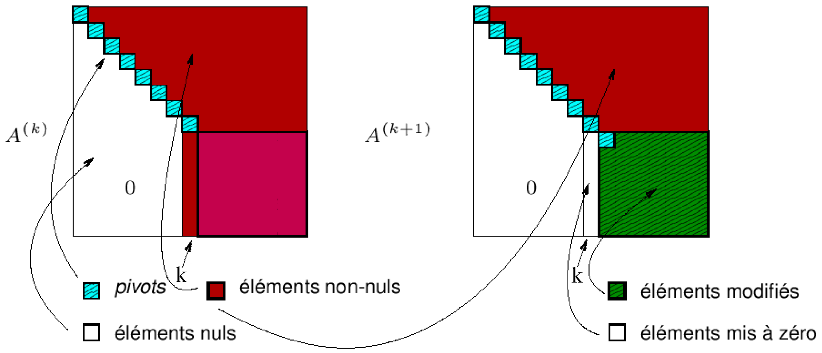
$$A^{(k)}\mathbf{x} = \mathbf{b}^{(k)}, \quad 1 \leq k \leq n, \tag{3}$$

where, for $k \geq 2$, the matrix $A^{(k)}$ is of the form

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & \ldots & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & & \vdots \\ 0 & \ldots & 0 & a_{kk}^{(k)} & \ldots & a_{kn}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \ldots & 0 & a_{nk}^{(k)} & \ldots & a_{nn}^{(k)} \end{bmatrix},$$

where we assume $a_{ii}^{(i)} \neq 0$ for $i = 1, \ldots, k-1$.

Gauss elimination method: diagram showing how the matrix $A^{(k+1)}$ is obtained from the matrix $A^{(k)}$.



It is clear that for $k = n$ we obtain the following upper triangular system $A^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & \vdots \\ 0 & & & \ddots & \vdots \\ 0 & & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ \vdots \\ b_n^{(n)} \end{bmatrix}.$$

To be consistent with the previous notation, we write as U upper triangular matrix $A^{(n)}$. The elements $a_{kk}^{(k)}$ are called *pivots* and have to be non-zero for $k = 1, \dots, n-1$.

In order to make explicit the formulae to get from the k-th systm to the $k+1$-th, for $k = 1, \dots, n-1$, we asssume that $a_{kk}^{(k)} \neq 0$ and we define the multiplier

$$\boxed{l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1, \dots, n,} \qquad [(n-k) \text{ operations}] \qquad (4)$$

we set then

$$\boxed{a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, \quad i, j = k+1, \dots, n,} \qquad [2(n-k)^2 \text{ operations}]$$
$$\boxed{b_i^{(k+1)} = b_i^{(k)} - l_{ik} b_k^{(k)}, \quad i = k+1, \dots, n.} \qquad [2(n-k) \text{ operations}] \qquad (5)$$

Remark

To perform the Gauss elimination,

$$
\begin{aligned}
2\sum_{k=1}^{n-1}(n-k)^2 + 3\sum_{k=1}^{n-1}(n-k) &= \\
2\sum_{p=1}^{n-1}p^2 + 3\sum_{p=1}^{n-1}p &= 2\frac{(n-1)n(2n-1)}{6} + 3\frac{n(n-1)}{2}
\end{aligned}
$$

operations are required, plus n^2 operations for the resolution with the backward substitutions method of the triangular system $U\,\mathbf{x} = \mathbf{b}^{(n)}$. By keeping only the dominant elements (of order n^3), we can say that the Gauss elimination method has a cost of around

$$\frac{2}{3}n^3 \text{ operations.}$$

The following table shows the estimated computation time to solve a system using the LU factorization in different computers:

	Number of flops of the computer		
n	10^9 (Giga)	10^{12} (Tera)	10^{15} (Peta)
10^2	$7\ 10^{-4}$ sec	negligible	negligible
10^4	11 min	0.7 sec	$7\ 10^{-4}$ sec
10^6	21 years	7.7 months	11 min
10^8	o.r.	o.r.	21 years

Algebra of Gaussian elimination

Example

Given the matrix

$$A = \begin{pmatrix} 3 & 1 & 3 \\ 6 & 7 & 3 \\ 9 & 12 & 3 \end{pmatrix}$$

Let

$$M_1 = \begin{pmatrix} 1 & & \\ -2 & 1 & \\ -3 & & 1 \end{pmatrix}, \quad M_1 A = \begin{pmatrix} 3 & 1 & 3 \\ 0 & 5 & -3 \\ 0 & 9 & -6 \end{pmatrix}$$

Algebra of Gaussian elimination

- In general, with $\mathbf{I}_{k-1} \in \mathbb{R}^{(k-1)\times(k-1)}$, $\mathbf{I} \in \mathbb{R}^{n\times n}$ being the identity matrices,

$$
\begin{aligned}
A^{(k+1)} &= M_k A^{(k)} := \begin{pmatrix} \mathbf{I}_{k-1} & & & & \\ & 1 & & & \\ & -l_{k+1,k} & 1 & & \\ & \dots & & \ddots & \\ & -l_{n,k} & & & 1 \end{pmatrix} A^{(k)}, \text{where} \\
M_k &= \mathbf{I} - m_k e_k^T, \quad M_k^{-1} = \mathbf{I} + m_k e_k^T
\end{aligned}
$$

  where e_k is the k-th unit vector, $m_k = (0, \dots, 0, l_{k+1,k}, \dots, l_{n,k})^T$, $e_i^T m_k = 0, \forall i \leq k$

- The factorization can be written as

$$M_{n-1} \dots M_1 A = A^{(n)} = U$$

Algebra of Gaussian elimination

- We obtain

$$\begin{aligned} A &= M_1^{-1} \dots M_{n-1}^{-1} U \\ &= (\mathbf{I} + m_1 e_1^T) \dots (\mathbf{I} + m_{n-1} e_{n-1}^T) U \\ &= \left(\mathbf{I} + \sum_{i=1}^{n-1} m_i e_i^T \right) U \\ &= \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & 1 \end{pmatrix} U = LU \end{aligned}$$

Gauss $\sim LU$

We have shown that the Gauss method is equivalent to the factorization $A = LU$ of the matrix A, with $L =$ multiplier matrix and $U = A^{(n)}$.

More exactly:

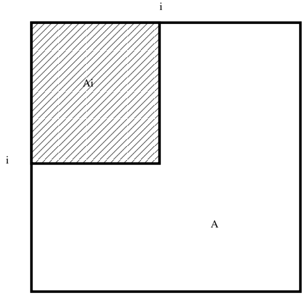
$$\boxed{A = \underbrace{\begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ l_{21} & 1 & & & 0 \\ \vdots & l_{32} & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ l_{n1} & & & l_{n,n-1} & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & \vdots \\ 0 & & & \ddots & \vdots \\ 0 & & & & a_{nn}^{(n)} \end{bmatrix}}_{U}.}$$

The Gauss method is only properly defined if the pivots $a_{kk}^{(k)}$ are non zero for $k = 1, \ldots, n-1$. Unfortunately, knowing that the diagonal elements of A are not zero is not enough to avoid null pivots during the elimination phase. For example, the matrix A in (6) is invertible and its diagonal elements are non zero

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}, \text{but we find } A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & \boxed{0} & -1 \\ 0 & -6 & -12 \end{bmatrix}. \qquad (6)$$

Nevertheless, we have to stop the Gauss method at he second step, because $a_{22}^{(2)} = 0$.

Let A_i be the i-th main submatrix of A $(i = 1, \ldots, n-1)$, i.e. the submatrix made of the i first rows and columns of A:



and let d_i be the principal minor of A defined as $d_i = det(A_i)$. We have the following result.

Proposition

(Proposition 6.2 in the book) For a given matrix $A \in \mathbb{R}^{n\times n}$, its Gauss factorization exists and is unique iff the principal submatrices A_i $(i = 1, \dots, n-1)$ are non singular (i.e. the principal minors d_i are non zero: $d_i \neq 0$).

Remark: If $d_i \neq 0$ $(i = 1, \dots, n-1)$, then the pivots $a_{ii}^{(i)}$ are also non zero.

The matrix of the previous example doesn't satisfy this condition because $d_1 = 1$ but $d_2 = 0$.

There are some categories of matrices for which the hypothesis of the proposition (1) are fulfilled. In particular, we mention:

1. *Strictly diagonal dominant by row* matrices. A matrix A is said diagonal dominant by row if

$$|a_{ii}| \geq \sum_{j=1,\dots,n; j\neq i} |a_{ij}|, \quad i = 1, \dots, n.$$

2. *Strictly diagonal dominant by column* matrices. A matrix A is said diagonal dominant by column if

$$|a_{jj}| \geq \sum_{i=1,\dots,n; i\neq j} |a_{ij}|, \quad j = 1, \dots, n.$$

Examples: $\begin{bmatrix} -4 & 1 & 2 \\ 2 & 5 & 0 \\ -2 & 1 & 7 \end{bmatrix}$ is diagonal dominant by row and by column, whereas
$\begin{bmatrix} -3 & 1 & 2 \\ 2 & 5 & 0 \\ -2 & 1 & 7 \end{bmatrix}$ is only diagonal dominant by row.

3. *Symmetric definite positive* matrices. A matrix A is symmetric if $A = A^T$; it is definite positive if and only of $z^T Az > 0$ for all $z \in \mathbb{R}^n$ with $z \neq 0$. All its eigenvalues are positive:

$$\lambda_i(A) > 0, \quad i = 1, \dots, n.$$

The matrices L and U only depend on A (and not on $\mathbf{b}$), the same factorization can be resused for solving several linear systems that share the same matrix A but different vectors $\mathbf{b}$.

The number of operations is then considerably reduced, since most of the computational weight, around $\frac{2}{3}n^3$ *flops*, is due to the Gaussian elimination process. Indeed, let us consider the M linear systems:

$$A\mathbf{x}_m = \mathbf{b}_m \qquad m = 1, \ldots, M$$

- the cost of the factorization $A = LU$ is $\frac{2}{3}n^3$ *flops*;
- the cost of the resolution of both triangular systems, $L\mathbf{y}_m = \mathbf{b}_m$ and $U\mathbf{x}_m = \mathbf{y}_m$ $(m = 1, \ldots, M)$ is $2Mn^2$ *flops*,

for a total of $\frac{2}{3}n^3 + 2Mn^2$ *flops* which is much smaller than $\frac{2}{3}Mn^3$ *flops* required to solve all the systems with Gaussian elimination.

Plan

The pivoting technique

It has been already noted that the Gauss method fails if a pivot becomes zero.

In that case, we can use a technique called pivoting that consists in exchanging the rows (or the columns) of the system in such a way that no pivot is zero.

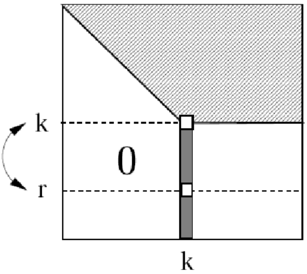
Example

Let us go back to the matrix (6) for which the Gauss method gives a null pivot at the second step. By just exchanging the second and the third rows, we get a non zero pivot and can execute one step further. Indeed,

$$A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & \boxed{0} & -1 \\ 0 & -6 & -12 \end{bmatrix} \quad \Longrightarrow \quad P_2 A^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & \boxed{-6} & -12 \\ 0 & 0 & -1 \end{bmatrix},$$

where $P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ is called permutation matrix.

The pivoting strategy used for the example 2 can be generalized by finding, at every step k of the elimination, a non zero pivot among the elements of the subcolumn $A^{(k)}(k:n,k)$. This is called a *partial pivot change* (by row).



From (4) we know that a big value of l_{ik} (coming for instance from a small $a_{kk}^{(k)}$) can amplify the rounding errors affecting the elements $a_{kj}^{(k)}$.

Consequently, in order to ensure a better stability, we choose as pivot the biggest element in module of the column $A^{(k)}(k : n, k)$, and the partial pivoting is performed at every step, even if it is not strictly necesary (i.e. even if the pivot is non zero).

In general, if at the step k we have to exchange the rows k and r, we will have to multiply $A^{(k)}$ by the following permutation matrix P_k before continuing:

$$\begin{array}{cc} & \\ & \\ k \rightarrow & \\ & \\ r \rightarrow & \\ & \\ & \\ & \end{array} \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & \dots & 1 & & \\ & & & \vdots & & & \\ & & 1 & \dots & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \\ & & \uparrow & & \uparrow & & \\ & & r & & k & & \end{pmatrix} = P_k$$

This means we will consider $P_k A^{(k)}$ insted of $A^{(k)}$.

We can prove that the result obtained is of the form:

$$\boxed{PA = LU,} \tag{7}$$

being $P = P_{n-1}P_{n-2}\dots P_2P_1$ (global permutation matrix). L is the multiplier matrix (the new ones!) and $U = A^{(n)}$.

Once the matrices L, U and P have been calculated, the resolution of the initial system is transformed into the resolution of the triangular systems

$$A\mathbf{x} = \mathbf{b} \implies PA\mathbf{x} = P\mathbf{b} \implies \begin{cases} L\mathbf{y} = P\mathbf{b}, \\ U\mathbf{x} = \mathbf{y}. \end{cases}$$

Remark that the coefficients of the matrix L have the same values as the multipliers calculated by an LU factorization of the matrix PA without pivoting.

Remark

In Matlab, we can get the factorization of a matrix A with the command

```
>> [L,U,P] = lu(A);
```

The matrix P is a permutation matrix. In the case where the matrix P is the identity, the matrices L and U are the matrices we are looking for (such that $LU = A$). Otherwise, we have $LU = PA$.

Example In Matlab, we first need to define the matrix `A` and the vector `b` of the linear system:

```
>> A = [-0.37, 0.05, 0.05, 0.07; 0.05, -0.116, 0, 0.05;...
          0.05, 0, -0.116, 0.05; 0.07, 0.05, 0.05, -0.202];
>> b = [-2; 0; 0; 0];
```

Then, we can use the command $\backslash$ as follows:

```
>> x = A\b
x =
    8.1172
    5.9893
    5.9893
    5.7779
```

This command computes the solution of the system with *ad hoc* algorithms (it tests the matrix to choose an optimal algorithm).

Example

If we wanted to use the LU factorization:

```
>> [L,U,P] = lu(A);
>> y = L\(P*b);
>> x = U\y
x =
    8.1172
    5.9893
    5.9893
    5.7779
```

The solution is the same. We can verify that, in this case, no permutation takes place (P is the identity matrix):

```
>> P
P =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

Remark

Using the LU factorization, obtained by fixing the value 1 *for the n diagonal elements of L, we can calculate the determinant of a square matrix with* $O(n^3)$ *operations, because*

$$\boxed{\det(A) = \det(L)\det(U) = \det(U) = \prod_{k=1}^{n} u_{kk};}$$

indeed, the determinant of a triangular matrix is the product of the diagonal elements. The Matlab command det(A) makes use of this.

Plan

The Cholesky factorization

In the case where the $n \times n$ matrix A is symmetric and positive definite, there exists a unique upper triangular matrix R with positive diagonal elements such that

$$\boxed{A = R^T R.}$$

This factorization is called *Cholesky factorization*. In Matlab, the command

```
>> R = chol(A)
```

can be used to compute R.

The elements r_{ij} of R can be calculated using the expressions $r_{11} = \sqrt{a_{11}}$ and for $i = 2, \ldots, n$:

$$r_{ji} = \frac{1}{r_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} r_{ki} r_{kj} \right), \quad j = 1, \ldots, i-1, \tag{8}$$

$$r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2} \tag{9}$$

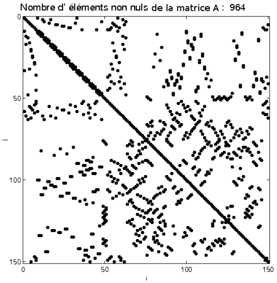
The Cholesky factorization needs around $\frac{n^3}{3}$ operations (half the operations for a LU factorization).

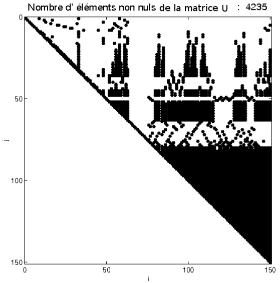
Memory space limitations

Example

Let us consider the problem of calculating the deformations in a structure subject to a given set of forces. The discretization using the finite elements method generates a matrix A of size 150×150. (The same matrix would have been produced by the approximation of an electric potential field.) This matrix is symmetric definite positive. The number of non-null entries of A is 964, and thus much smaller than $(150)^2 = 22500$. It is a *sparse* matrix.

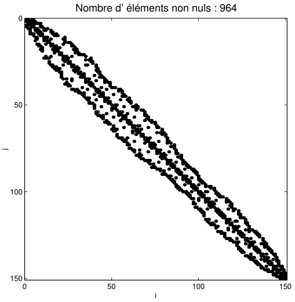
Example The figure on the left shows the disposition of the non-null entries of A, whereas the one on the right shows the non-null entries of the matrix R.

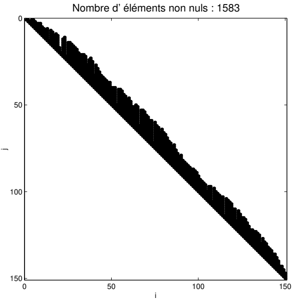




Example We notice that the number of non-null entries of R is much bigger than those of A (*fill-in phenomenon*). This leads to a bigger memory usage. To reduce the fill-in phenomenon, we can reorder rows and columns of A in a particular fashion; this is called *reordering* of the matrix. There are several algorithms that allow us to do this.

Example For example, the following figure shows, on the left, one possiblility of reordering A, while the one on the right shows the disposition of the non-null entries of the Cholesky factorization of the reordered matrix A.





Plan

Accuracy of the numerical solution for direct methods

(Sect. 6.2.5 of the book)

The methods we have seen until now allow us to find the solution of a linear system in a finite number of operations. That is why they are called *direct methods*. However, there are cases where these methods are not satisfactory.

Accuracy of the numerical approximation

Due to round-off errors, solving numerically the linear system $Ax = b$ is equivalent to solving, in exact arithmetic, the following perturbed linear system:

$$(A + \delta A)\,\hat{x} = b + \delta b,$$

where $\hat{x} \in \mathbb{R}^n$ is the numerical solution, $\delta A \in \mathbb{R}^{n \times n}$ the perturbation matrix of A, and $\delta b \in \mathbb{R}^n$ the perturbation vector of b.

Question: Is $\hat{x}$ close to x ?

Precision limitations

Example: the Hilbert matrix of size $n \times n$ is a symmetric matrix defined by:

$$\boxed{A_{ij} = \frac{1}{i+j-1}, \qquad i,j = 1,\ldots,n}$$

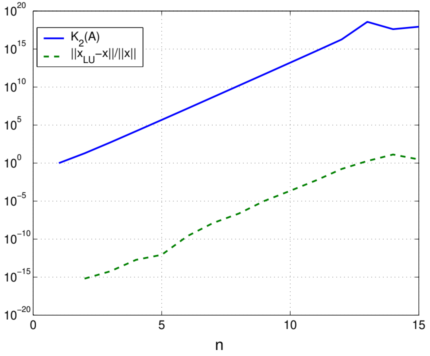
In Matlab, we can build a Hilbert matrix of any size n with the command `A = hilb(n)`. For example, for $n = 4$, we get:

$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

We consider the linear systems $A_n\mathbf{x}_n = \mathbf{b}_n$ where A_n is the Hilbert matrix of size n with $n = 4, 6, 8, 10, 12, 14, \ldots$, whereas $\mathbf{b}_n$ is chosen such that the exact solution is $\mathbf{x}_n = (1, 1, \cdots, 1)^T$.

Example

For every n, we calculate the conditioning of the matrix, we solve the linear system by LU factorization and we get $\mathbf{x}_n^{LU}$ as the found solution. The obtained conditioning as well as the error $\|\mathbf{x}_n - \mathbf{x}_n^{LU}\|/\|\mathbf{x}_n\|$ (where $\|\cdot\|$ is the euclidian norm of a vector, $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \cdot \mathbf{x}}$) are shown in the figure below.



Some definitions

The p-norm of a vector $\mathbf{v} \in \mathbb{R}^n$ is defined as:

$$\|\mathbf{v}\|_p := \left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p} \quad \text{for } 1 \leq p \leq +\infty.$$

We obtain:

$$\|\mathbf{v}\|_2 = \sqrt{\mathbf{v}\cdot\mathbf{v}} = \sqrt{\sum_{i=1}^{n} |v_i|^2}, \quad \|\mathbf{v}\|_1 = \sum_{i=1}^{n} |v_i|, \quad \text{and} \quad \|\mathbf{v}\|_\infty = \max_{i=1,\dots,n} |v_i|.$$

Often, the norm 2 of the vector $\mathbf{v}$ indicated as $\|\mathbf{v}\| \equiv \|\mathbf{v}\|_2$.

Some definitions

The p-norm of a vector $\mathbf{v} \in \mathbb{R}^n$ is defined as:

$$\|\mathbf{v}\|_p := \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} \quad \text{for } 1 \leq p \leq +\infty.$$

We obtain:

$$\|\mathbf{v}\|_2 = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{\sum_{i=1}^{n} |v_i|^2}, \quad \|\mathbf{v}\|_1 = \sum_{i=1}^{n} |v_i|, \quad \text{and} \quad \|\mathbf{v}\|_\infty = \max_{i=1,\dots,n} |v_i|.$$

Often, the norm 2 of the vector $\mathbf{v}$ indicated as $\|\mathbf{v}\| \equiv \|\mathbf{v}\|_2$.

Eigenvalues and spectral radius: definitions

Given $A \in \mathbb{C}^{n\times n}$, its eigenvalues $\{\lambda_i(A)\}_{i=1}^n \in \mathbb{C}$ and the corresponding eigenvectors $\{\mathbf{v}_i\}_{i=1}^n \in \mathbb{C}^n$ are such that

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad \text{for all } i = 1, \dots, n.$$

The eigenvalues $\{\lambda_i(A)\}_{i=1}^n$ correspond to the zeros of the characteristic polynomial of the matrix A, say

$$p_A(\lambda) := \det(A - \lambda I).$$

The spectral radius of $A \in \mathbb{C}^{n\times n}$, with eigenvalues $\{\lambda_i(A)\}_{i=1}^n \subset \mathbb{C}$, is defined as:

$$\rho(A) := \max_{i=1,\dots,n} |\lambda_i(A)|.$$

Remark: If A is nonsingular, $\lambda_i(A^{-1}) = 1/\lambda_{n+1-i}(A)$ for $i = 1, \dots, n$

Eigenvalues and spectral radius: definitions

Given $A \in \mathbb{C}^{n\times n}$, its eigenvalues $\{\lambda_i(A)\}_{i=1}^n \in \mathbb{C}$ and the corresponding eigenvectors $\{\mathbf{v}_i\}_{i=1}^n \in \mathbb{C}^n$ are such that

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad \text{for all } i = 1, \ldots, n.$$

The eigenvalues $\{\lambda_i(A)\}_{i=1}^n$ correspond to the zeros of the characteristic polynomial of the matrix A, say

$$p_A(\lambda) := \det(A - \lambda I).$$

The spectral radius of $A \in \mathbb{C}^{n\times n}$, with eigenvalues $\{\lambda_i(A)\}_{i=1}^n \subset \mathbb{C}$, is defined as:

$$\rho(A) := \max_{i=1,\ldots,n} |\lambda_i(A)|.$$

Remark: If A is nonsingular, $\lambda_i(A^{-1}) = 1/\lambda_{n+1-i}(A)$ for $i = 1, \ldots, n$

Eigenvalues and spectral radius: definitions

Given $A \in \mathbb{C}^{n\times n}$, its eigenvalues $\{\lambda_i(A)\}_{i=1}^n \in \mathbb{C}$ and the corresponding eigenvectors $\{\mathbf{v}_i\}_{i=1}^n \in \mathbb{C}^n$ are such that

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad \text{for all } i = 1, \dots, n.$$

The eigenvalues $\{\lambda_i(A)\}_{i=1}^n$ correspond to the zeros of the characteristic polynomial of the matrix A, say

$$p_A(\lambda) := \det(A - \lambda I).$$

The spectral radius of $A \in \mathbb{C}^{n\times n}$, with eigenvalues $\{\lambda_i(A)\}_{i=1}^n \subset \mathbb{C}$, is defined as:

$$\rho(A) := \max_{i=1,\dots,n} |\lambda_i(A)|.$$

Remark: If A is nonsingular, $\lambda_i(A^{-1}) = 1/\lambda_{n+1-i}(A)$ for $i = 1, \dots, n$

Eigenvalues of a symmetric matrix

Proposition

If the matrix $A \in \mathbb{R}^{n \times n}$ is symmetric, then its eigenvalues are real, i.e., $\lambda_i(A) \in \mathbb{R}$ for all $i = 1, \ldots, n$.

If $A \in \mathbb{R}^{n \times n}$ is symmetric, then it is also positive definite (SPD) if and only if all its eigenvalues are strictly positive, i.e., $\lambda_i(A) > 0$ for all $i = 1, \ldots, n$.

Norm of a matrix: definitions

For the matrix $A \in \mathbb{R}^{n \times n}$, its p-norm is defined as:

$$\|A\|_p := \sup_{\mathbf{v} \in \mathbb{R}^n,\, \mathbf{v} \neq 0} \frac{\|A\mathbf{v}\|_p}{\|\mathbf{v}\|_p}.$$

For $A \in \mathbb{R}^{n \times n}$, we have:

- $\|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}|$;
- $\|A\|_\infty = \max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{ij}|$;
- $\|A\|_2 = \sup_{\mathbf{v} \in \mathbb{R}^n,\, \mathbf{v} \neq 0} \frac{\|A\mathbf{v}\|_2}{\|\mathbf{v}\|_2} = \sqrt{\lambda_{\max}(A^T A)}$, with $\lambda_{\max}(A^T A)$ the maximum eigenvalue of $A^T A$;
  $\rightarrow \|Av\|_2 \leq \|A\|_2 \|v\|_2$
- if A is symmetric and positive definite, then $\|A\|_2 = \lambda_{\max}(A)$ since $\lambda_{\max}(A^T A) = (\lambda_{\max}(A))^2$.

Conditioning number: definitions

The condition number in the p-norm of a nonsingular matrix $A \in \mathbb{R}^{n \times n}$ is:

$$K_p(A) := \|A\|_p \, \|A^{-1}\|_p \quad \text{for some } 1 \leq p \leq +\infty.$$

The spectral condition number of a nonsingular matrix $A \in \mathbb{R}^{n \times n}$ is:

$$K(A) := \frac{\rho(A)}{\rho(A^{-1})} = \frac{\max_{i=1,\dots,n} |\lambda_i(A)|}{\min_{i=1,\dots,n} |\lambda_i(A)|},$$

where $\rho(A)$ and $\rho(A^{-1})$ are the spectral radii of the matrices A and A^{-1}, respectively.

Conditioning number

For the nonsingular matrix $A \in \mathbb{R}^{n\times n}$, we have:

$$K_p(A) \geq 1 \quad \text{for all } 1 \leq p \leq +\infty;$$

$$K_2(A) = \|A\|_2 \, \|A^{-1}\|_2 = \frac{\sqrt{\lambda_{\max}(A^T A)}}{\sqrt{\lambda_{\min}(A^T A)}};$$

If the eigenvalues of A are real and strictly positive, then $K(A) = \dfrac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$,

where $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the maximum and minimum eigenvalues of A, respectively.

Definition

We call **conditioning** of a matrix A, symmetric definite positive, the ratio between the maximum and minimum of its eigenvalues, i.e.

$$\boxed{K_2(A) \equiv K(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}}$$

Conditioning number

For the nonsingular matrix $A \in \mathbb{R}^{n\times n}$, we have:

$$K_p(A) \geq 1 \quad \text{for all } 1 \leq p \leq +\infty;$$

$$K_2(A) = \|A\|_2 \, \|A^{-1}\|_2 = \frac{\sqrt{\lambda_{\max}(A^T A)}}{\sqrt{\lambda_{\min}(A^T A)}};$$

If the eigenvalues of A are real and strictly positive, then $K(A) = \dfrac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$, where $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the maximum and minimum eigenvalues of A, respectively.

Definition

We call **conditioning** of a matrix A, symmetric definite positive, the ratio between the maximum and minimum of its eigenvalues, i.e.

$$\boxed{K_2(A) \equiv K(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}}$$

Error and residual for linear systems

Definition:
For the linear system $A\mathbf{x} = \mathbf{b}$, we define:

- the (absolute) error $\mathbf{e} := \mathbf{x} - \hat{\mathbf{x}}$, with $\mathbf{e} \in \mathbb{R}^n$;
- the relative error $e_{\text{rel}} := \frac{\|\mathbf{x}-\hat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2}$, for $\mathbf{x} \neq 0$, with $e_{\text{rel}} \in \mathbb{R}$;
- the residual $\mathbf{r} := \mathbf{b} - A\hat{\mathbf{x}}$, with $\mathbf{r} \in \mathbb{R}^n$;
- the relative residual $r_{\text{rel}} := \frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2}$, for $\mathbf{b} \neq 0$, with $r_{\text{rel}} \in \mathbb{R}$.

Accuracy of the numerical approximation

It can be shown that, the bigger the conditioning of a matrix, the worse the solution obtained by a direct method.
Solve:

$$A\mathbf{x} = \mathbf{b}$$

Assume A is symmetric positive definite, $\delta A = 0$ and consider the perturbed system $A\hat{\mathbf{x}} = \mathbf{b} + \delta\mathbf{b}$
The following relation can be shown :

$$\boxed{e_{\mathrm{rel}} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq K(A)\frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2}} \qquad (10)$$

where $\mathbf{r}$ is the residual $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$.
Remark that, if the conditioning of A is big, the distance $\|\mathbf{x} - \hat{\mathbf{x}}\|$ between the exact solution and the numerically found solution can be very big even if the residual is very small.

Proof for (10) : We have:

$$\mathbf{x} - \hat{\mathbf{x}} = A^{-1}\mathbf{b} - A^{-1}\mathbf{b} - A^{-1}\delta\mathbf{b} = -A^{-1}\delta\mathbf{b}$$

We obtain:

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_2 \leq \|A^{-1}\|_2\|\delta\mathbf{b}\|_2 = \frac{1}{\lambda_{min}(A)}\|\delta\mathbf{b}\|_2$$

We also have:

$$Ax = b \implies \|\mathbf{b}\|_2 \leq \|A\|_2\|x\|_2 = \lambda_{max}(A)\|x\|_2$$

We obtain:

$$e_{\text{rel}} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq K(A)\frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2}$$

because $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$