Numerical Calculus

Pătrulescu Flavius

Technical University of Cluj-Napoca

2024

References

1. R.L. Burden, D.J. Faires, A.M. Burden, *Numerical Analysis*, 10th Ed., Cengage Learning, Boston, 2022.
2. J.W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
3. A. Quarteroni, R. Sacco, F. Saleri, *Numerical Mathematics*, Springer-Verlag, New-York, 2000.
4. L.F. Shampine, R.C. Allen, S. Pruess, *Fundamental of Numerical Computing*, John Wiley & Sons Inc., New York, 1997.
5. E. Süli, D. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, Cambridge, 2003.

Linear Systems (Süli and Mayers, 2003)

To compute the solution of a system of n simultaneous linear equations according to Cramer's rule, we need to evaluate $n+1$ determinants, each of size $n \times n$, so the total number of operations required is about $e(n+1)!$ as $n \to \infty$. For $n = 100$, this means approximately $101!e \approx 2.56 \times 10^{160}$ arithmetic operations. Today's fastest parallel computers are capable of teraflop speeds, *i.e.*, 10^{12} floating point operations per second; therefore, the computing time for our solution would be around 8.11×10^{140} years. According to the prevailing theoretical position, the Universe began in a violent explosion, the Big Bang, about $12.5(\pm 3) \times 10^9$ years ago.

Linear Systems (Quarteroni *et al.*, 2000)

For this reason, numerical methods that are alternatives to Cramer's rule have been developed. They are called *direct methods* if they yield the solution of the system in a finite number of steps, and *iterative* if they require (theoretically) an infinite number of steps. We notice from now on that the choice between a direct and an iterative method does not depend only on the theoretical efficiency of the scheme, but also on the particular type of matrix, on memory storage requirements, and finally, on the architecture of the computer.

Solution of Triangular Systems (Quarteroni *et al.*, 2000)

Lower triangular system

$$\begin{cases} l_{11}x_1 = b_1 \\ l_{21}x_1 + l_{22}x_2 = b_2 \\ \dots \quad \dots \quad \dots \\ l_{n1}x_1 + l_{n2}x_2 + \dots + l_{nn}x_n = b_n \end{cases}$$

$$\underbrace{\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix}}_{L:=} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\boldsymbol{x}:=} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{\boldsymbol{b}:=}$$

Forward substitution (Quarteroni *et al.*, 2000)

$$\begin{cases} x_1 = \frac{b_1}{l_{11}} \\ x_i = \frac{1}{l_{ii}}\left(b_i - \sum_{j=1}^{i-1} l_{ij}x_j\right),\ i = 2, \ldots, n \end{cases}$$

Number of multiplications and divisions is $\frac{n(n+1)}{2}$.

Number of sums and subtractions is $\frac{n(n-1)}{2}$.

The global operations count is n^2. (Complexity: $O(n^2)$)

Solution of Triangular Systems (Quarteroni *et al.*, 2000)

Upper triangular system

$$\begin{cases} u_{11}x_1 + u_{12}x_2 + \ldots + u_{1n}x_n = b_1 \\ \qquad\quad u_{22}x_2 + \ldots + u_{2n}x_n = b_2 \\ \qquad\qquad \ldots \quad \ldots \quad \ldots \\ \qquad\qquad\qquad\qquad u_{nn}x_n = b_n \end{cases}$$

$$\underbrace{\begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1n} \\ 0 & u_{22} & \ldots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & u_{nn} \end{bmatrix}}_{U:=} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\boldsymbol{x}:=} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{\boldsymbol{b}:=}$$

Backward substitution (Quarteroni *et al.*, 2000)

$$\begin{cases} x_n = \frac{b_n}{u_{nn}} \\ x_i = \frac{1}{u_{ii}}\left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j\right),\ i = n-1,\ldots,1 \end{cases}$$

Number of multiplications and divisions is $\frac{n(n+1)}{2}$.

Number of sums and subtractions is $\frac{n(n-1)}{2}$.

The global operations count is n^2. (Complexity: $O(n^2)$)

Gaussian Elimination (Burden *et al.*, 2022)

$$\begin{cases} (E_1) : a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ (E_2) : a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \qquad\qquad \ldots \quad \ldots \quad \ldots \\ (E_n) : a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}}_{A:=} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\boldsymbol{x}:=} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{\boldsymbol{b}:=}$$

Gussian elimination (Burden *et al.*, 2022)

We use three operations to simplify the linear system

- Equation E_i can be multiplied by any nonzero constant λ with the resulting equation used in place of E_i. This operation is denoted $(\lambda E_i) \rightarrow (E_i)$.
- Equation E_j can be multiplied by any constant λ and added to equation E_i, the resulting equation being used in place of E_i. This operation is denoted $(E_i + \lambda E_j) \rightarrow (E_i)$
- Equations E_i and E_j can be transposed. This operation is denoted $(E_i) \leftrightarrow (E_j)$

By a sequence of these operations, a linear system will be systematically transformed into to a new linear system that is more easily solved and has the same solutions.

Gaussian Elimination (Burden *et al.*, 2022)

Augmented matrix

$$A^{(1)} := [A, \boldsymbol{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right] \underbrace{=}_{a^{(1)}_{in+1} := b_i}$$

$$= \underbrace{\left[\begin{array}{cccc|c} a^{(1)}_{11} & a^{(1)}_{12} & \dots & a^{(1)}_{1n} & a^{(1)}_{1n+1} \\ a^{(1)}_{21} & a^{(1)}_{22} & \dots & a^{(1)}_{2n} & a^{(1)}_{2n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a^{(1)}_{n1} & a^{(1)}_{n2} & \dots & a^{(1)}_{nn} & a^{(1)}_{nn+1} \end{array}\right]}_{A^{(1)} :=}$$

Gaussian Elimination (Burden *et al.*, 2022)

Provided $a_{11} \neq 0$, we perform the operations corresponding to

$$E_j - (a_{j1}/a_{11})E_1 \to E_j, \quad j = 2, 3, \ldots, n$$

to eliminate the coefficient of x_1 in each of these rows.

$$\underbrace{\left[\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n-1}^{(1)} & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n-1}^{(2)} & a_{2n}^{(2)} & a_{2n+1}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn-1}^{(2)} & a_{nn}^{(2)} & a_{nn+1}^{(2)} \end{array}\right]}_{A^{(2)} :=}$$

Gaussian Elimination (Burden *et al.*, 2022)

We follow a sequential procedure for $i = 2, 3, \ldots, n-1$ and perform the operation

$$E_j - (a_{ji}/a_{ii})E_i \rightarrow E_j, \quad j = i+1, \ldots, n$$

provided $a_{ii} \neq 0$. This eliminates x_i in each row below the ith for all values of $i = 1, 2, ..., n-1$. The augmented matrix $A^{(k)}$ has entries $a_{ij}^{(k)}$ defined by

$$a_{ij}^{(k)} = \begin{cases} a_{ij}^{(k-1)}, & i = \overline{1, k-1},\ j = \overline{1, n+1} \\ 0, & i = \overline{k, n},\ j = \overline{1, k-1} \\ a_{ij}^{(k-1)} - \frac{a_{ik-1}^{(k-1)}}{a_{k-1k-1}^{(k-1)}} a_{k-1j}^{(k-1)}, & i = \overline{k, n},\ j = \overline{k, n+1} \end{cases}$$

Gaussian Elimination (Burden *et al.*, 2022)

Thus, the augmented matrix $A^{(k)}$ is

$$\underbrace{\left[\begin{array}{ccccccc|c}
a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k-1}^{(1)} & a_{1k}^{(1)} & \cdots & a_{1n-1}^{(1)} & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\
0 & a_{22}^{(2)} & \cdots & a_{2k-1}^{(2)} & a_{2k}^{(2)} & \cdots & a_{2n-1}^{(2)} & a_{2n}^{(2)} & a_{2n+1}^{(2)} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & a_{k-1k-1}^{(k-1)} & a_{k-1k}^{(k-1)} & \cdots & a_{k-1n-1}^{(k-1)} & a_{k-1n}^{(k-1)} & a_{k-1n+1}^{(k-1)} \\
0 & 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn-1}^{(k)} & a_{kn}^{(k)} & a_{kn+1}^{(k)} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn-1}^{(k)} & a_{nn}^{(k)} & a_{nn+1}^{(k)}
\end{array}\right]}_{A^{(k)} :=}$$

The final matrix has the form

$$\underbrace{\left[\begin{array}{ccccc|c}
a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n-1}^{(1)} & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\
0 & a_{22}^{(2)} & \cdots & a_{2n-1}^{(2)} & a_{2n}^{(2)} & a_{2n+1}^{(2)} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & a_{nn}^{(n)} & a_{nn+1}^{(n)}
\end{array}\right]}_{A^{(n)} :=}$$

Gaussian Elimination (Burden *et al.*, 2022)

The new linear system is triangular and backward substitution can be performed.

$$\underbrace{\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn}^{(n)} \end{bmatrix}}_{U:=} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \underbrace{\begin{bmatrix} a_{1n+1}^{(1)} \\ a_{2n+1}^{(2)} \\ \vdots \\ a_{nn+1}^{(n)} \end{bmatrix}}_{\bar{\boldsymbol{b}}:=}$$

$$U\boldsymbol{x} = \bar{\boldsymbol{b}}$$

Gaussian Elimination (Burden *et al.*, 2022)

The procedure will fail if one of the elements $a_{11}^{(1)}, a_{22}^{(2)}, \ldots, a_{n-1n-1}^{(n-1)}$ is zero because either the step

$$\left(E_i - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} E_k\right) \rightarrow E_i$$

cannot be performed or the backward substitution cannot be accomplished (in the case $a_{nn}^{(n)} = 0$). The system may still have a solution, but the technique for finding the solution must be altered.

Gaussian Elimination (Burden *et al.*, 2022)

Case $a_{kk}^{(k)} = 0$ for some $k = 1, 2, \ldots, n-1$:

if $a_{pk}^{(k)} \neq 0$ for some $k+1 \leqslant p \leqslant n$ then the operation $(E_k) \leftrightarrow (E_p)$ is performed.

if $a_{pk}^{(k)} = 0$ for each $k+1 \leqslant p \leqslant n$ the linear system does not have a unique solution and the procedure stops.

Finally, if $a_{nn}^{(n)} = 0$ the linear system does not have a unique solution and again the procedure stops.

Gaussian Elimination (Burden *et al.*, 2022)

To reduce round-off error, it is often necessary to perform row interchanges even when the pivot elements are not zero ($a_{kk}^{(k)} \neq 0$). If $a_{kk}^{(k)}$ is small in magnitude compared to $a_{ik}^{(k)}$, $i = \overline{k+1, n}$ then the magnitude of the multiplier

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

will be much larger than 1. The round-off error introduced in the computation of one of the terms $a_{kl}^{(k)}$, $l = \overline{k+1, n}$ is multiplied by m_{ik} when computing $a_{il}^{(k+1)}$ $(i, l = \overline{k+1, n})$, which compounded the original error.

Partial Pivoting (Burden *et al.*, 2022)

Also, when performing the backward substitution for

$$x_k = \frac{a_{kn+1}^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j}{a_{kk}^{(k)}}$$

with a small value of $a_{kk}^{(k)}$, any error in the numerator can be dramatically increased. The simplest strategy, called *partial pivoting*, is to select an element in the same column that is below the diagonal and has the largest absolute value; specifically, we determine the smallest $p > k$ such that

$$|a_{pk}^{(k)}| = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|$$

and perform $(E_k) \leftrightarrow (E_p)$. Each multiplier in the partial pivoting algorithm has magnitude less than or equal to 1.

Gaussian Elimination (Burden *et al., 2022*)

Multiplications/divisions

$$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$$

Additions/subtractions

$$\frac{n^3}{3} - \frac{n}{3}$$

The global operations count is

$$\frac{2n^3}{3} + \frac{n^2}{2} - \frac{7n}{6}$$

Complexity: $O(\frac{2n^3}{3})$.

Gauss-Jordan Method (Burden *et al., 2022*)

This method is described in the following. Use the ith equation to eliminate x_i not only from equations $E_{i+1}, E_{i+2}, \ldots, E_n$ as was done in the Gaussian elimination method, but also from $E_1, E_2, \ldots, E_{i-1}$. On reducing $[A, b]$ to

$$\left[\begin{array}{ccccc|c} a_{11}^{(1)} & 0 & \ldots & 0 & 0 & a_{1n+1}^{(1)} \\ 0 & a_{22}^{(2)} & \ldots & 0 & 0 & a_{2n+1}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & a_{nn}^{(n)} & a_{nn+1}^{(n)} \end{array}\right]$$

the solution is obtained by setting

$$x_i = \frac{a_{in+1}^{(i)}}{a_{ii}^{(i)}}, \quad i = 1, 2, \ldots, n.$$

This procedure circumvents the backward substitution in the Gaussian elimination.

Gauss-Jordan Method (Burden *et al.*, *2022*)

Additions/Subtractions

$$\frac{n^3}{2} - \frac{n}{2}$$

Multiplications/Divisions

$$\frac{n^3}{2} + n^2 - \frac{n}{2}$$

Complexity: $O(n^3)$.

Special Types of Matrices (Burden *et al.*, 2022)

A strictly diagonally dominant matrix A is non-singular. In addition, in this case, Gaussian elimination can be performed on any linear system of the form $A\boldsymbol{x} = \boldsymbol{b}$ to obtain its unique solution without row or column interchanges, and the calculations will be stable with respect to the growth of round-off errors.

The symmetric matrix A is positive definite if and only if Gaussian elimination without row interchanges can be performed on the linear system $A\boldsymbol{x} = \boldsymbol{b}$ with all pivot elements positive. Moreover, in this case, the computations are stable with respect to the growth of round-off errors.

Complete Pivoting (Burden *et al.*, 2022)

Pivoting can incorporate the interchange of both rows and columns. *Complete (or maximal) pivoting* at the kth step searches all the entries a_{ij}, for $i = k, k+1, \ldots, n$ and $j = k, k+1, \ldots, n$, to find the entry with the largest magnitude. Both row and column interchanges are performed to bring this entry to the pivot position. The total additional time required to incorporate complete pivoting into Gaussian elimination is

$$\sum_{k=2}^{n} (k^2 - 1) = \frac{n(n-1)(2n+5)}{6}$$

comparisons. Complete pivoting is, consequently, the strategy recommended only for systems where accuracy is essential and the amount of execution time needed for this method can be justified.

Matrix Factorization (Burden *et al.*, 2022)

The steps used to solve a system $A\boldsymbol{x} = \boldsymbol{b}$ can be used to factor a matrix. Suppose that A has been factored into the triangular form $A = LU$, where L is lower triangular and U is upper triangular. Then we can solve the system more easily using a two-step process.

$$A\boldsymbol{x} = \boldsymbol{b} \Leftrightarrow L\underbrace{U\boldsymbol{x}}_{\boldsymbol{y}:=} = \boldsymbol{b} \Leftrightarrow \begin{cases} L\boldsymbol{y} = \boldsymbol{b} \\ U\boldsymbol{x} = \boldsymbol{y} \end{cases}$$

Solve the lower-triangular system $L\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y}$.

Solve the upper-triangular system $U\boldsymbol{x} = \boldsymbol{y}$.

Once factorization is determined, systems involving the matrix A can be solved in this simplified manner for any number of $\boldsymbol{b}$.

Matrix Factorization (Burden *et al.*, 2022)

Solving a linear system $A\boldsymbol{x} = \boldsymbol{b}$ in factored form means that the number of operations needed to solve the system $A\boldsymbol{x} = \boldsymbol{b}$ is reduced from $O(\frac{2}{3}n^3)$ to $O(2n^2)$.

Suppose that Gaussian elimination can be performed on the system without row interchanges. This is equivalent to having nonzero pivot elements $a_{ii}^{(i)} \neq 0$ for each $i = \overline{1, n}$.

Matrix Factorization (Burden *et al.*, 2022)

The first step in the Gaussian elimination process consists of performing, for each $j = \overline{2, n}$, the operations

$$(E_j - m_{j1}E_1) \to E_j, \quad m_{j1} = \frac{a_{j1}^{(1)}}{a_{11}^{(1)}}$$

The previous system of operations is simultaneously accomplished by multiplying the original matrix $A^{(1)}$ on the left by *the first Gaussian transformation matrix:*

$$M^{(1)} := \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ -m_{21} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -m_{n1} & 0 & \dots & 0 & 1 \end{bmatrix}$$

$$A^{(2)} := M^{(1)}A^{(1)}$$

Matrix Factorization (Burden *et al.*, 2022)

The second step in the Gaussian elimination process consists of performing, for each $j = \overline{3, n}$, the operations

$$(E_j - m_{j2}E_2) \to E_j, \quad m_{j2} = \frac{a_{j2}^{(2)}}{a_{22}^{(2)}}$$

The previous system of operations is simultaneously accomplished by multiplying the original matrix $A^{(2)}$ on the left by *the second Gaussian transformation matrix:*

$$M^{(2)} := \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & -m_{32} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & -m_{n2} & 0 & \dots & 0 & 1 \end{bmatrix}$$

$$A^{(3)} := M^{(2)}A^{(2)} = M^{(2)}M^{(1)}A^{(1)}$$

Matrix Factorization (Burden *et al.*, 2022)

The kth step in the Gaussian elimination process consists of performing, for each $j = \overline{k+1, n}$, the operations

$$(E_j - m_{jk}E_k) \to E_j, \quad m_{jk} = \frac{a_{jk}^{(k)}}{a_{kk}^{(k)}}$$

The previous system of operations is simultaneously accomplished by multiplying the original matrix $A^{(k)}$ on the left by *the kth Gaussian transformation matrix*

$$M^{(k)} := \begin{bmatrix} 1 & 0 & \dots & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & -m_{k+1k} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -m_{nk} & \dots & 0 & 1 \end{bmatrix}$$

$$A^{(k+1)} := M^{(k)}A^{(k)} = M^{(k)}M^{(k-1)} \dots M^{(1)}A^{(1)}$$

Matrix Factorization (Burden *et al.*, 2022)

We obtain $A^{(n)}$ given by

$$A^{(n)} = M^{(n-1)} \dots M^{(1)} A^{(1)}$$

This process forms the upper triangular matrix

$$U := A^{(n)}(1:n, 1:n)$$

portion of the matrix factorization $A = LU$.
The lower-triangular matrix L in the factorization of A is the product of the matrices

$$L^{(k)} = (M^{(k)})^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & m_{k+1k} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & m_{nk} & \dots & 0 & 1 \end{bmatrix}$$

Matrix Factorization (Burden *et al.*, 2022)

$$
\begin{aligned}
L &= L^{(1)}L^{(2)}\dots L^{(n-1)} = \\
&= \begin{bmatrix}
1 & 0 & \dots & 0 & \dots & 0 & 0 \\
m_{21} & 1 & \dots & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
m_{k1} & m_{k2} & \dots & 1 & \dots & 0 & 0 \\
m_{k+11} & m_{k+12} & \dots & m_{k+1k} & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
m_{n-11} & m_{n-12} & \dots & m_{n-1k} & \dots & 1 & 0 \\
m_{n1} & m_{n2} & \dots & m_{nk} & \dots & m_{nn-1} & 1
\end{bmatrix}
\end{aligned}
$$

Matrix Factorization (Burden *et al.*, 2022)

If Gaussian elimination can be performed on the linear system $A\boldsymbol{x} = \boldsymbol{b}$ without row interchanges, then the matrix A can be factored into the product of a lower-triangular matrix L and an upper-triangular matrix U, that is, $A = LU$, where $m_{ij} = \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}}$,

$$U = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn}^{(n)} \end{bmatrix}, \; L = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ m_{21} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{n1} & m_{n2} & \dots & m_{nn-1} & 1 \end{bmatrix}$$

Compact Form of Factorization (Quarteroni *et al.*, 2000)

Remarkable variants of factorization LU are the *Crout factorization* and *Doolittle factorization*, and are also known as *compact forms* of the Gauss elimination method. This name is due to the fact that these approaches require less intermediate results than the standard G.E.M. to generate the factorization of A.
Computing the LU factorization of A is formally equivalent to solving the following nonlinear system of n^2 equations

$$a_{ij} = \sum_{r=1}^{\min(i,j)} l_{ir} u_{rj},$$

the unknowns being the $n^2 + n$ coefficients of the triangular matrices L and U.

Compact Form of Factorization (Quarteroni *et al.*, 2000)

In fact, supposing that the first $k-1$ columns of L and U are available, the following equations are obtained

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + l_{kk} u_{kj}, \; j = \overline{k, n}$$

$$a_{ik} = \sum_{r=1}^{k-1} l_{ir} u_{rk} + l_{ik} u_{kk}, \; i = \overline{k+1, n}$$

If we arbitrarily set n coefficients to 1, for example the diagonal entries of L or U,we end up with the *Doolittle and Crout methods*, respectively.

Doolittle Method (Quarteroni *et al.*, 2000)

Setting $l_{kk} = 1$ the following equations are obtained

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + u_{kj}, \; j = \overline{k, n}$$

$$a_{ik} = \sum_{r=1}^{k-1} l_{ir} u_{rk} + l_{ik} u_{kk}, \; i = \overline{k+1, n}$$

Note that these equations can be solved in a sequential way with respect to the variables u_{kj} and l_{ik}. We thus obtain first the kth row of U and then the kth column of L, as follows: for $k = 1, \ldots, n$

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}, \; j = \overline{k, n}$$

$$l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk} \right), \; i = \overline{k+1, n}$$

Crout Method (Quarteroni *et al.*, 2000)

Setting $u_{kk} = 1$ the following equations are obtained

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + l_{kk} u_{kj}, \; j = \overline{k, n}$$

$$a_{ik} = \sum_{r=1}^{k-1} l_{ir} u_{rk} + l_{ik}, \; i = \overline{k+1, n}$$

Note that these equations can be solved in a sequential way with respect to the variables u_{kj} and l_{ik}. We thus obtain first the kth column of L and then the kth row of U, as follows: for $k = 1, \ldots, n$

$$l_{ik} = a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk}, \; i = \overline{k, n}$$

$$u_{kj} = \frac{1}{l_{kk}} \left(a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj} \right), \; j = \overline{k+1, n}$$

Real symmetric positive definite matrices (Demmel, 1997)

Let $A \in \mathcal{M}_n(\mathbb{R})$ a symmetric positive definite (s.p.d.) matrix ($A = A^t$ and $\boldsymbol{x}^t A \boldsymbol{x} > 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \neq \mathbf{0}_{\mathbb{R}^n}$)

A is s.p.d. $\Leftrightarrow X^t A X$ is s.p.d. $\forall\, X$, $\det X \neq 0$

A is s.p.d. and H is any principal submatrix of $A \Rightarrow H$ is s.p.d.

A is s.p.d. $\Leftrightarrow A = A^t$ and all its eigenvalues are positive

A is s.p.d. $\Rightarrow$ all $a_{ii} > 0$ and

$$\max_{i,j=\overline{1,n}} |a_{ij}| = \max_{i=\overline{1,n}} a_{ii} > 0$$

Cholesky factorization (Quarteroni *et al.*, 2000)

A is s.p.d. $\Leftrightarrow$ there is a unique non-singular lower triangular matrix L, with positive diagonal entries, such that $A = LL^t$
$A = LL^t$ is called *Cholesky factorization* and L is called *Cholesky factor* of A
Cholesky algorithm

$$l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\right)^{1/2}, \forall\, j = \overline{1, n}$$

$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\right) / l_{jj}, \forall\, i = \overline{j+1, n}$$

Cholesky factorization (Burden *et al., 2022*)

The LL^t Cholesky factorization of a positive definite matrix requires

- $\frac{n^3}{6} - \frac{n}{6}$ additions/subtractions
- $\frac{n^3}{6} + \frac{n^2}{2} - \frac{2n}{3}$ multiplications/divisions
- n square roots

The global operations count: $\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$

Complexity $O(\frac{n^3}{3})$

This computational advantage of Cholesky's factorization is misleading because it requires extracting n square roots. However, the number of operations required for computing the n square roots is a linear factor of n and will decrease in significance as n increases.

Cholesky factorization (Demmel,1997; Quarteroni *et al.*, 2000)

If A is not positive definite, then (in exact arithmetic) this algorithm will fail by attempting to compute the square root of a negative number or by dividing by zero; this is the cheapest way to test if a symmetric matrix is positive definite.
Also, for Cholesky factorization, it is possible to overwrite the matrix L in the lower triangular portion of A, without any additional memory storage. By doing so, both A and factorization are preserved, noting that A is stored in the upper triangular section since it is symmetric and that its diagonal entries can be computed as

$$a_{11} = l_{11}^2,\ a_{ii} = l_{ii}^2 + \sum_{k=1}^{i-1} l_{ik}^2$$

Thomas Algorithm (Shampine *et al.*, 1997)

The *Thomas algorithm* is a simplified form of Gaussian elimination that can be used to solve *tridiagonal system of equations*

$$\begin{bmatrix} a_1 & c_1 & \dots & \dots & \dots & 0 \\ d_2 & a_2 & c_2 & \dots & \dots & 0 \\ 0 & d_3 & a_3 & c_3 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & 0 \\ 0 & 0 & \dots & d_{n-1} & a_{n-1} & c_{n-1} \\ 0 & 0 & \dots & 0 & d_n & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Equivalent form

$$\begin{cases} a_1 x_1 + c_1 x_2 = b_1 \\ d_i x_{i-1} + a_i x_i + c_i x_{i+1} = b_i, \quad i = 2, \dots, n-1 \\ d_n x_{n-1} + a_n x_n = b_n \end{cases}$$

Thomas Algorithm (Shampine *et al.*, 1997)

Thomas algorithm is done in two distinct steps. The first consists of removing the coefficients d_k through a forward sweep.

$$f_1 = a_1,\ f_{k+1} = a_{k+1} - \frac{d_{k+1}}{f_k} c_k$$

$$e_1 = d_1,\ e_{k+1} = b_{k+1} - \frac{d_{k+1}}{f_k} e_k$$

The new system with the coefficient matrix having two diagonals (*bidiagonal system*) looks like

$$\begin{bmatrix} f_1 & c_1 & \dots & \dots & \dots & 0 \\ 0 & f_2 & c_2 & \dots & \dots & 0 \\ 0 & 0 & f_3 & c_3 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & 0 \\ 0 & 0 & \dots & 0 & f_{n-1} & c_{n-1} \\ 0 & 0 & \dots & 0 & 0 & f_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix}$$

Thomas Algorithm (Shampine *et al.*, 1997)

Backward substitution is

$$\begin{cases} x_n = \frac{e_n}{f_n} \\ x_k = \frac{1}{f_k}(e_k - c_k x_{k+1}), \quad k = \overline{n-1, 1} \end{cases}$$

Storage can be managed extremely efficiently. A general matrix requires storage for n^2 entries, but a tridiagonal matrix requires storage for only $3n - 2$ entries. A natural scheme is to store the three diagonal bands with a_k, d_k, and c_k as three vectors of maximum length n.

Condition number (Quarteroni *et al.*, 2000)

The condition number of a matrix $A \in \mathcal{M}_n(\mathbb{R})$ is defined as

$$k(A) = \left|\left|\left| A^{-1} \right|\right|\right| \left|\left|\left| A \right|\right|\right|$$

where $|||\cdot|||$ is an induced matrix norm. In general, $k(A)$ depends on the choice of the norm.

$$1 = \left|\left|\left| A^{-1}A \right|\right|\right| \leqslant \left|\left|\left| A^{-1} \right|\right|\right| \left|\left|\left| A \right|\right|\right| = k(A)$$

$$k(A^{-1}) = k(A)$$

$$k(\alpha A) = k(A), \, \forall \, \alpha \in \mathbb{R}^*$$

$$A\text{-orthogonal} \Rightarrow \; k_2(A) = 1$$

The condition number of a singular matrix is set equal to infinity,

$$k(A) = \infty \Leftrightarrow \det(A) = 0$$

Relative distance (Quarteroni *et al.*, 2000)

Relative distance of $A \in \mathcal{M}_n(\mathbb{R})$ from the set of singular matrices with respect to the p-norm is defined by

$$dist_p(A) = \min \left\{ \frac{|||\delta A|||_p}{|||A|||_p} : A + \delta A \text{ is singular} \right\}$$

where $\delta A \in \mathcal{M}_n(\mathbb{R})$. We have

$$dist_p(A) = \frac{1}{k_p(A)}$$

A matrix A with a higher condition number can behave like a singular matrix of the form $A + \delta A$.

Forward *a priori* analysis (Quarteroni *et al.*, 2000)

Forward *a priori* analysis is a measure of the sensitivity of the system

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

to changes in the data. Due to rounding errors, a numerical method for solving the system does not provide the exact solution but only of approximate one, which satisfy a *perturbed system*

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

Let $A \in \mathcal{M}_n(\mathbb{R})$ be a non-singular matrix, $\delta A \in \mathcal{M}_n(\mathbb{R})$ be s.t.

$$|||\delta A||| \, |||A||| < 1.$$

If $\boldsymbol{x} \in \mathbb{R}^n$ is the solution of the system and $\boldsymbol{b}$, $\delta \boldsymbol{x}$, $\delta \boldsymbol{b} \in \mathbb{R}^n$ then

$$\frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{k(A)}{1 - k(A)\frac{|||\delta A|||}{|||A|||}} \left(\frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|} + \frac{|||\delta A|||}{|||A|||} \right), \ \boldsymbol{b} \neq \boldsymbol{0}_{\mathbb{R}^n}$$

Forward *a priori* analysis(Quarteroni *et al.*, 2000)

Let $\delta A = O_n$. Then

$$\frac{1}{k(A)} \frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leqslant \frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant k(A) \frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|}$$

In order to employ the previous inequalities in the analysis of propagation of rounding errors in the case of direct methods, $|||\delta A|||$ and $\|\delta \boldsymbol{b}\|$ should be bounded in terms of the dimension of the system and of the characteristics of the floating-point arithmetic that is being used.
Assume that $|||\delta A||| \leqslant \gamma |||A|||$, $\|\delta \boldsymbol{b}\| \leqslant \gamma \|\boldsymbol{b}\|$. If $\gamma k(A) < 1$ then

$$\frac{\|\boldsymbol{x} + \delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{1 + \gamma k(A)}{1 - \gamma k(A)}, \ \gamma \in \mathbb{R}_+$$

$$\frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{2\gamma}{1 - \gamma k(A)} k(A), \ \gamma \in \mathbb{R}_+$$