Numerical Methods

Resolution of Linear Systems

I. Direct Methods

2022-2023

[DΣIM]

Linear Systems

- We are interested in numerical methods for solving linear systems of the form:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

- Or in matrix form:

$$\mathbf{Ax} = \mathbf{b}$$

- **A** is the system matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

- And the vector **b** is the independent term:

$$\mathbf{b} = (b_1, \ldots, b_n)^T$$

Diagonals

- Let **A** be an $m \times n$ matrix, not necessarily square, and let $k = min\{m, n\}$. The elements $a_{ii}, i = 1,2, \dots, k$ are said to lie on the ***diagonal*** of **A**, and a_{ii} is called the *i*th diagonal element of **A**. The elements $a_{i,i+1}$ are said to lie on the ***superdiagonal*** of **A**, the elements $a_{i,i-1}$ on the ***subdiagonal*** of **A**. If **A** is square, the elements $a_{n-i+1,i}$ $i = 1,2, \dots, k$ are said to lie on the ***secondary diagonal*** of **A**.

Diagonal Matrix

- A square matrix of size $n \times n$ is ***diagonal*** if its only nonzero elements lie on the principal diagonal. We can write:

$$\mathbf{A} = diag(a_{11}, a_{22}, \ldots, a_{nn})$$

$$\mathbf{A} = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ \vdots & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

Trapezoidal Matrices

- A $m \times n$ matrix is ***upper trapezoidal*** if:

$$i > j \Rightarrow a_{ij} = 0$$

- An it is ***lower trapezoidal*** if:

$$i < j \Rightarrow a_{ij} = 0$$

- A square upper (lower) trapezoidal matrix is said to be ***upper (lower) triangular***.

$$\mathbf{A} = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \end{pmatrix}$$

Strictly Triangular

- If $\boldsymbol{T}$ is upper (lower) triangular with ***zero diagonal elements***, then $\boldsymbol{T}$ is said to be ***strictly upper (lower) triangular***. If the ***diagonal elements of T are unity***, T is said to be ***unit upper (lower) triangular.***
- Note that a matrix is diagonal if and only if it is both upper and lower triangular.

Hessenberg Matrices

- A square matrix **A** is ***upper Hessenberg*** if

$$i > j+1 \Rightarrow a_{ij} = 0$$

- And it is ***lower Hessenberg*** if:

$$i > j-1 \Rightarrow a_{ij} = 0$$

- An upper Hessenberg matrix is zero below its subdiagonal. A lower Hessenberg matrix is zero above its superdiagonal.

$$\mathbf{H} = \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix}$$

Tridiagonal Matrices

- A square matrix is ***tridiagonal*** if it is both upper and lower Hessenberg. A tridiagonal matrix has its nonzero elements arranged in a band along its diagonals.

$$\mathbf{T} = \begin{pmatrix} x & x & 0 & 0 & 0 \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}$$

Submatrices

- Let **A** be a given matrix and suppose that certain rows and columns of **A** have been selected. The rectangular array of elements of **A** laying in the intersection of these rows and columns is again a matrix and is called a ***submatrix*** of **A**

$$S = \begin{pmatrix} a_{22} & a_{23} & a_{25} \\ a_{42} & a_{43} & a_{45} \end{pmatrix}$$

Submatrices

- Let $\mathbf{A}$ be an $m \times n$ matrix and let the selected rows $1 \leq i_1 < i_2 < \cdots < i_k \leq m$ and the selected columns $1 \leq j_1 < j_2 < \cdots < j_l \leq n$. The $k \times l$ matrix $\mathbf{S}$ whose (μ, γ) element is :

$$\sigma_{\mu\gamma} = a_{i_\mu i_\gamma}$$

- Is called a ***submatrix of*** $\mathbf{A}$. If $k = l$ and $i_1 = j_1$, while $i_2 = j_2, \ldots, i_k = j_k$, then $\mathbf{S}$ is called a ***principal submatrix*** of $\mathbf{A}$. If $i_1 = 1, i_2 = 2, \ldots, i_k = k$ and we have $j_1 = 1, j_2 = 2, \ldots, j_k = l$, then S is called a ***leading submatrix*** of $\mathbf{A}$.

Partitioned Matrices

- An $m \times n$ matrix $\mathbf{A}$ is said to be partitioned into submatrices when it is written in the form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1l} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{k1} & \mathbf{A}_{k2} & \cdots & \mathbf{A}_{kl} \end{pmatrix}$$

- Where each $\mathbf{A}_{ij}$ is an $m_i \times n_j$ submatrix of $\mathbf{A}$.

Partitioned Matrices

- A partitioned matrix is a matrix of matrices.
- The partition in the example has m_i rows, and it must be true that $m_1 + m_2 + \cdots + m_k = m$, the number of rows in A. Similarly, the submatrices of the jth column of the partition have n_j columns and $n_1 + n_2 + \cdots + n_l = n$. The matrix $\mathbf{A}_{11}$ is a leading submatrix of **A**.

Linear Systems

- We can use two kind of methods to solve linear systems:
  - ***Direct Methods:*** We obtain the solution after a determined number of operations. There is no possibility for controlling the final precision.
  - ***Iterative Methods:*** We obtain the solution after an iterative process. The precision of the solution will depend on the number of iterations and can be controlled beforehand.

Cramer's Algorithm

- Given the linear system

$$\mathbf{Ax} = \mathbf{b}$$

- Where **A** is a regular matrix with det(**A**)≠0, we can solve the system using Cramer's algorithm.

$$x_i = \frac{D_i}{D_0} \quad i = 1, \ldots, N$$

Cramer's Algorithm

- Where D_0 is the determinant of matrix **A**

$$D_0 = \det(\mathbf{A})$$

- The D_i, are the determinants obtained by substituting the *i-column* in D_0 by the vector ***b.***
- However, in order to compute a determinant of size $n \times n$ we need to compute the $n!$ permutations of n objects. Each permutation needs $n - 1$ multiplications, and we have a total of $n + 1$ determinants

Cramer's Algorithm

- We will need a total number of operations of

$$(n+1)n!(n-1) \approx n(n+1)!$$

- For great values of n this number is enormous. In the common case of matrices of size 1000×1000 we can get computing times greater than the age of the universe!
- Another important problem would be the rounding errors appearing in such a big number of operations.

Triangular Systems

- This is a case of linear systems which is easy to solve. In these systems the matrix $\mathbf{A}$ is always an ***upper or lower triangular matrix***.
- The upper systems are solved with the ***backwards algorithm*** and the lower systems with the ***forward algorithm***, both of order n^2

Triangular Systems

- Consider the upper triangular system:

$$
\begin{aligned}
u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n &= b_1 \\
u_{22}x_2 + \cdots + u_{2n}x_n &= b_2 \\
&\vdots \\
u_{nn}x_n &= b_n
\end{aligned}
$$

- For a nonsingular system, it must be true that:

$$\det(U) = u_{11}u_{22}\cdots u_{nn} \neq 0$$

Triangular Systems

- We can use the following recursive algorithm, known as ***backwards substitution***:

$$x_n = \frac{b_n}{u_{nn}}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij} x_j}{u_{ii}}, \quad i = n-1, \ldots, 1$$

Triangular Systems

- As for upper triangular systems, in the case of lower triangular systems:

$$
\begin{aligned}
&l_{11}x_1 = b_1 \\
&l_{21}x_1 + l_{22}x_2 = b_2 \\
&\vdots \\
&l_{n1}x_1 + l_{n2}x_2 + \cdots + l_{nn}x_n = b_n
\end{aligned}
$$

- We can use also an algorithm of the order n^2

Triangular Systems

- This recursive algorithm is known as ***forward substitution:***

$$x_1 = \frac{b_1}{l_{11}}$$

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij} x_j}{l_{ii}}, \quad i = 2, \ldots, n$$

Gauss Method

- Gaussian methods first ***transform the system in order to get a triangular system***. Then we use the forward or backwards substitution to solve the system.
- Let's write out our linear system as:

$$
\begin{aligned}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\
a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\
&\vdots \\
a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + \cdots + a_{nn}^{(1)}x_n &= b_n^{(1)}
\end{aligned}
$$

Gauss Method

- We will use also the notation:

$$b_i^{(1)} = a_{in+1}^{(1)}$$

- All the information on the system is stored in its constants. We can rewrite the system with the ***augmented matrix***:

$$\mathbf{A}^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & a_{2n+1}^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & a_{nn+1}^{(1)} \end{pmatrix}$$

Gauss Method

- The classical gauss method transforms the former matrix using $n-1$ steps to an upper triangular matrix. This is a process of order $O(n^3)$

$$A^{(n)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2n+1}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} & a_{nn+1}^{(n)} \end{pmatrix}$$

Gauss Method

- In each of the $n-1$ steps we transform to zero all the matrix elements under the principal diagonal, column by column.
- For the first column, we must subtract to rows i=2,3,...,n the first row multiplied by the factor:

$$\pi_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$$

Gauss Method

- The new coefficients of the matrix are

$$
\begin{aligned}
a_{1j}^{(2)} &= a_{1j}^{(1)} && j = 1,\ldots,n+1 \\
a_{i1}^{(2)} &= 0 && i = 2,\ldots,n \\
a_{ij}^{(2)} &= a_{ij}^{(1)} - \pi_{i1} a_{1j}^{(1)} && i = 2,\ldots,n
\end{aligned}
$$

- And the matrix becomes:

$$
\mathbf{A}^{(2)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2n+1}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_{nn+1}^{(2)} \end{pmatrix}
$$

Gauss Method

- And after p steps the matrix of the system has the form:

$$
\mathbf{A}^{(p+1)} = \begin{pmatrix}
a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1p}^{(1)} & a_{1p+1}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\
0 & a_{22}^{(2)} & \cdots & a_{2p}^{(2)} & a_{2p+1}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2n+1}^{(2)} \\
\vdots & \ddots & \ddots & \vdots & \vdots & & \vdots & \vdots \\
\vdots & & \ddots & a_{pp}^{(p)} & a_{pp+1}^{(p)} & & a_{pn}^{(p)} & a_{pn+1}^{(p)} \\
\vdots & & & 0 & a_{p+1p+1}^{(p+1)} & & a_{p+1n}^{(p+1)} & a_{p+1n+1}^{(p+1)} \\
\vdots & & & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & a_{np+1}^{(p+1)} & \cdots & a_{nn}^{(p+1)} & a_{nn+1}^{(p+1)}
\end{pmatrix}
$$

Gauss Method

- The new coefficients will be:

$$a_{pj}^{(p+1)} = a_{pj}^{(p)} \qquad j = p,\ldots,n+1$$

$$a_{ip}^{(p+1)} = 0 \qquad i = p+1,\ldots,n$$

$$a_{ij}^{(p+1)} = a_{ij}^{(p)} - \pi_{ip} a_{pj}^{(p)} \qquad i = p+1,\ldots,n, \;\; j = p+1,\ldots,n$$

- Note that after p steps we do not modify anymore the rows i=1,…,p.

Gauss Method

- After n-1 steps we get the augmented matrix corresponding to a triangular system:

$$\mathbf{A}^{(n-1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1p}^{(1)} & a_{1p+1}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1n+1}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2p}^{(2)} & a_{2p+1}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2n+1}^{(2)} \\ \vdots & \ddots & \ddots & \vdots & \vdots & & \vdots & \vdots \\ \vdots & & \ddots & a_{pp}^{(p)} & a_{pp+1}^{(p)} & & a_{pn}^{(p)} & a_{pn+1}^{(p)} \\ \vdots & & & 0 & a_{p+1p+1}^{(p+1)} & & a_{p+1n}^{(p+1)} & a_{p+1n+1}^{(p+1)} \\ \vdots & & & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 & \cdots & a_{nn}^{(n-1)} & a_{nn+1}^{(n-1)} \end{pmatrix} \vdots$$

Pivoting

- We will need an additional correction if we want that our algorithm works always for any nonsingular matrix.
- Consider for instance the system matrix:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 2 \end{pmatrix}$$

Pivoting

- After the first step, we get the matrix:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

- And as:

$$a_{22}^{(2)} = 0$$

- We will be dividing by zero.

Pivoting

- The algorithm described will not work, as the division by zero will generate an overflow.
- To avoid this problem, we just need to reorder the rows:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

- This process is called ***pivoting***

Pivoting

- Pivoting is not just necessary in the case of zero coefficients. When the multiplying factors are greater than the unit, if there is a great number of operations, all the ***rounding errors*** will be ***largely amplified***. These will propagate through our computations and the results could be completely wrong.

Pivoting

- Consider the following example:

$$\begin{pmatrix} \delta & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

- Where δ is a very small value. After the first step in gauss method, we have:

$$\begin{pmatrix} \delta & 1 & 1 \\ 0 & 1-\frac{1}{\delta} & 2-\frac{1}{\delta} \end{pmatrix}$$

Pivoting

- And the backwards substitution, after machine rounding will give the solutions:

$$y = \frac{2\delta - 1}{\delta - 1} \approx 1$$

$$x = \frac{1 - y}{\delta} \approx 0$$

- Which are completely wrong.

Pivoting

- But, if we just change the order of the rows before applying the Gauss method, we have the system:

$$\begin{pmatrix} 1 & 1 & 2 \\ 0 & 1-\delta & 1-2\delta \end{pmatrix}$$

- And we would obtain the correct solution:

$$x \approx 1 \quad y \approx 1$$

Pivoting

- There are different strategies to change rows or columns in order to improve the final numerical results:
  - ***Partial pivoting:*** In step p we look for the value:

$$a_{kp}^{(p)} = \max_{p \leq i \leq n} a_{ip}^{(p)}$$

  - And we interchange rows p and k. The search time will be proportional to $n - p - 1$

Pivoting

- ***Total pivoting:*** In this case, in step p, we look for the value:

$$a_{kr}^{(p)} = \max_{\substack{p \le i \le n \\ p \le j \le n}} a_{ij}^{(p)}$$

- Now we interchange row p with row k and column p with column r. The search time is far greater now, of the order of $O(n-p-1)^2$ and we need to reorder also the solution vector, as changing the order of the columns we alter also the order of the components of the solution x_i

Number of Operations

- We can give an estimation of the total number of operations needed to obtain a triangular system using Gaussian elimination.
- If we are working with a system matrix of size $n \times n$ and we are in the i step, we will need $n - i$ divisions to compute the multiplicative factors and $(n - i)(n - i + 2)$ products and subtractions to modify the elements under row i, apart from the elements in column i.

Number of Operations

- Products and divisions are harder to compute than additions or subtractions, so we should compute these numbers separately. Thus, in step i we will need a total of

$$(n-i)+(n-i)(n-i+1)=(n-i)(n-i+2)$$

- Multiplications/divisions and a total of

$$(n-i)(n-i+1)$$

- Additions/subtractions.

Number of Operations

- Then adding all the steps, we obtain a grand total of products or divisions of:

$$\sum_{i=1}^{n-1}(n-i)(n-i+2) = (n^2+2n)\sum_{i=1}^{n}1 - 2(n+1)\sum_{i=1}^{n-1}i + \sum_{i=1}^{n-1}i^2$$

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

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

Number of Operations

- And a grand total of additions/subtractions of:

$$\sum_{i=1}^{n-1}(n-i)(n-i+1) = (n^2+2n)\sum_{i=1}^{n-1}1-(2n+1)\sum_{i=1}^{n-1}i+\sum_{i=1}^{n-1}i^2$$

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

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

Number of Operations

- When the system is in triangular form, we still need to solve the triangular system. For each term we need $(n-i)$ products and $(n-i-1)$ additions plus a subtraction and a division. The total number of products/divisions in this process is then:

$$1+\sum_{1}^{n-1}\left((n-i)+1\right)=\frac{n^2+n}{2}$$

Number of Operations

- While the number of additions/subtractions is:

$$\sum_{i=1}^{n-1}\left((n-i-1)+1\right)=\frac{n^2-n}{2}$$

- The grand total of products/divisions is:

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

- And the number of additions/subtractions is:

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

Number of Operations

- We see that all these numbers go as:

$$\boxed{\frac{1}{3}n^3}$$

- This number is large when n increases, but this number is to be compared with the total number of operations in Cramer's method. If $n = 10$ we have about 700 operations in Gauss elimination against 400.000.000 operations in Cramer's method.

Gauss-Jordan method

- This algorithm is a variant of the classic Gauss method. Instead of transforming the system matrix to an upper triangular matrix, we obtain a diagonal matrix, giving a linear system that can be solved straightforward
- We just need in step p, ***subtract to all rows except to the p row***, the p row multiplied by the factor:

$$\pi_{ip} = \frac{a_{ip}^{(p)}}{a_{pp}^{(p)}}, \quad i = 1, \dots, p-1, p+1, \dots, n$$

Gauss-Jordan method

- Then after n steps we obtain the diagonal augmented matrix:

$$\mathbf{A}^{(n-1)} = \begin{pmatrix} a_{11}^{(1)} & & & & b_1^{(n)} \\ & a_{22}^{(2)} & & & b_2^{(n)} \\ & & \ddots & & \vdots \\ & & & a_{nn}^{(n)} & b_n^{(n)} \end{pmatrix}$$

- Giving a system with an easily computed solution:

$$x_i = \frac{b_i^{(n)}}{a_{ii}^{(i)}}, \quad i = 1, \ldots, n$$

Gauss-Jordan Method

- Each of the components of the independent vector **b** will receive n modifications
- For each step we modify $n-1$ rows, and we use $n-p$ additions and products plus the $n-1$ on the independent term. The total of additions and products is:

$$(n-1)\sum_{p=1}^{n}(n-p)+\sum_{p=1}^{n}(n-1)=\frac{n^3-n}{2}$$

- plus

$$n(n-1)=n^2-n$$

- divisions. The total is again of the order n^3

Thomas Algorithm

- In the case of tridiagonal dominant matrices, the gauss method is known also as the Thomas algorithm. Consider the system:

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

Thomas Algorithm

- If the following condition is fulfilled:

$$\left|b_j\right| > \left|a_j\right| + \left|c_j\right|$$

- We will say that the we have a ***diagonal dominant matrix***. The absolute value of the diagonal term must be greater than the sum of the absolute values of the elements of the row. This guarantees the existence of solution.

Thomas Algorithm

- This algorithm results from the application of the Gauss elimination algorithm to a tridiagonal system of the form:

$$
\begin{aligned}
b_1 x_1 + c_1 x_2 &= d_1 \\
a_i x_{i-1} + b_i x_i + c_i x_{i+1} &= d_i, \qquad i = 2,\ldots,n-1 \\
a_n x_n + b_n x_n &= d_n
\end{aligned}
$$

Thomas Algorithm

- To modify the second equation, we use the first equation as:

$$\left(\text{equation } 2\right)\cdot b_1 - \left(\text{equation } 1\right)\cdot a_2$$

- Which gives:

$$\left(b_2 b_1 - c_1 a_2\right) x_2 + c_2 b_1 x_3 = d_2 b_1 - d_1 a_2$$

- After this step x_1 has been eliminated from the second equation.

Thomas Algorithm

- This equation can be used to eliminate similarly x_2 from the third equation obtaining:

$$\left(b_3\left(b_2 b_1 - c_1 a_2\right) - c_2 b_1 a_3\right) x_3 + c_3 (b_2 b_1 - c_1 a_2) x_4$$
$$= d_3 (b_2 b_1 - c_1 a_2) - (d_2 b_1 - d_1 a_2) a_3$$

- This procedure can be repeated until the nth row, when the modified equation will contain only one unknown, x_n. Now the system has a triangular matrix.

Thomas Algorithm

- The coefficients of the modified equation get more and more complicated if stated explicitly. By examining the procedure, however, the modified coefficients may be defined recursively as:

$$\tilde{a}_i = 0$$

$$\tilde{b}_1 = b_1 \qquad \tilde{c}_1 = c_1 \qquad \tilde{d}_i = d_1$$

$$\tilde{b}_i = b_i \tilde{b}_{i-1} - \tilde{c}_{i-1} a_i \qquad \tilde{c}_i = c_i \tilde{b}_{i-1} \qquad \tilde{d}_i = d_i \tilde{b}_{i-1} - \tilde{d}_{i-1} a$$

Thomas Algorithm

- This process can be hastened if we know that no risk of division by zero, which is a common situation in dominant tridiagonal matrices.

$$a'_i = 0 \qquad c'_1 = \frac{c_1}{b_1} \qquad d'_1 = \frac{d_1}{b_1}$$

$$b'_i = 1 \qquad c'_i = \frac{c_i}{b_i - c'_{i-1}\, a_i} \qquad d'_i = \frac{d_i - d'_{i-1}\, a_i}{b_i - c'_{i-1}\, a_i}.$$

Thomas Algorithm

- This gives the following system with the same unknowns in triangular form:

$$x_i + c'_i x_{i+1} = d'_i \qquad i = 1, \ldots, n-1$$
$$x_n = d'_n$$

- Then, we can use the backwards substitution to solve the system:

$$x_n = d'_n$$
$$x_i = d'_i - c'_i x_{i+1}, \quad i = n-1, \ldots, 1$$

Matrix Form

- All the elementary row operations used to transform a matrix in Gaussian elimination can be expressed in matrix form. This gives a compact formulation of the direct method.
- The basic row operations are:
  - Multiply the ith row of $\mathbf{A}$ by a constant α
  - Interchange rows i and j.
  - Add α times row i to row j.

Matrix Form

- These elementary operations can be performed pre-multiplying the matrix **A** by a suitable matrix.
- Multiply the ith row of **A** by λ:

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} & & & i & & & \end{array} \\
\begin{array}{c} \\ \\ \\ i \\ \\ \\ \\ \end{array} &
\begin{pmatrix}
1 & & & & & & \\
 & \ddots & & \vdots & & & \\
 & & 1 & & & & \\
 & \cdots & & \lambda & & & \\
 & & & & 1 & & \\
 & & & & & \ddots & \\
 & & & & & & 1
\end{pmatrix}
\end{array}
$$

Matrix Form

- To interchange rows i and j we use ***permutation matrices:***

$$
\begin{array}{cc}
 & \begin{array}{ccccccccc} & & & i & & j & & & \end{array} \\
\begin{array}{c} \\ \\ \\ i \\ \\ j \\ \\ \\ \\ \end{array} &
\begin{pmatrix}
1 & & & & & & & & \\
 & \ddots & & \vdots & & \vdots & & & \\
 & & 1 & & & & & & \\
 & \cdots & & 0 & \cdots & 1 & & & \\
 & & & \vdots & & \vdots & & & \\
 & \cdots & & 1 & \cdots & 0 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & \ddots & \\
 & & & & & & & & 1
\end{pmatrix}
\end{array}
$$

Matrix Form

- If we want to add λ times row i to row j we use:

$$
\begin{array}{c c}
 & \begin{array}{ccccccc} & & i & & j & & \end{array} \\
\begin{array}{c} \\ \\ i \\ \\ j \\ \\ \\ \end{array} &
\begin{pmatrix}
1 & & & & & & \\
 & \ddots & \vdots & & \vdots & & \\
 & \cdots & 1 & \cdots & 0 & & \\
 & & \vdots & & \vdots & & \\
 & \cdots & \lambda & \cdots & 1 & & \\
 & & & & & \ddots & \\
 & & & & & & 1
\end{pmatrix}
\end{array}
$$

Matrix Form

- In the process of Gaussian elimination, in the kth step we modify all the rows under the kth diagonal element. All the modifications consist in subtract kth row to the rows below $k+1$,..,n using the suitable factors $\mu_{k+1}, \mu_{k+2}, \dots, \mu_n$.
- This modification can be expressed by an elementary lower triangular matrix of order n and index k. These are called ***Frobenius matrices of order k***

$$\mathbf{M}_k = \mathbf{I}_n - \mathbf{m}\mathbf{e}_k^T$$

Matrix Form

- Here $\mathbf{e}_k$ is the kth element of the canonical basis and $\mathbf{m} = (0,0,\dots,\mu_{k+1},\mu_{k+2},\dots,\mu_n)$.

$$\mathbf{M_k} = \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & \cdots & 0 \\ \vdots & & \ddots & \vdots & & & \\ 0 & 0 & \cdots & 1 & & & 0 \\ 0 & 0 & \cdots & -\mu_{k+1} & 1 & & 0 \\ \vdots & & & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & -\mu_n & \cdots & \cdots & 1 \end{pmatrix}$$

Matrix Form

- For instance, consider the system of linear equations:

$$\begin{aligned} 2x_1 + 4x_2 - 2x_3 &= 6 \\ x_1 - x_2 + 5x_3 &= 0 \\ 4x_1 + x_2 - 2x_3 &= 2 \end{aligned}$$

- With augmented matrix

$$\mathbf{Ab} = \begin{pmatrix} 2 & 4 & -2 & 6 \\ 1 & -1 & 5 & 0 \\ 4 & 1 & -2 & 2 \end{pmatrix}$$

- Using the matrices:

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}, \qquad M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{7}{3} & 1 \end{pmatrix}$$

- We obtain

$$Ab' = M_2 M_1 Ab = \begin{pmatrix} 2 & 4 & -2 & 6 \\ 0 & -3 & 6 & -3 \\ 0 & 0 & -12 & -3 \end{pmatrix}$$

- The augmented matrix of a triangular system ready to be solved by backwards substitution.

Matrix Form

- The process of triangularization of a system matrix can be expressed also using block matrices. The idea is that a Frobenius matrix $\mathbf{M}_k$ of index k that annihilates the last $n-k$ elements of the kth column of $\mathbf{A}^{(k)}$ can be written in the form:

$$\mathbf{M}_k = \begin{pmatrix} \mathbf{I}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{M'}_k \end{pmatrix}$$

- Where $\mathbf{M'}_k$ is a Frobenius matrix of order k

Matrix Form

- Then:

$$\mathbf{A}^{(k+1)} = \mathbf{M}_k \mathbf{A}^{(k)} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M'}_k \end{pmatrix} \begin{pmatrix} \mathbf{A}_{11}^{(k)} & \mathbf{A}_{12}^{(k)} \\ \mathbf{0} & \mathbf{A}_{22}^{(k)} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11}^{(k)} & \mathbf{A}_{12}^{(k)} \\ \mathbf{0} & \mathbf{M'}_k \mathbf{A}_{22}^{(k)} \end{pmatrix}$$

- The first $k-1$ rows of $\mathbf{A}^{(k+1)}$ are the same as those of $\mathbf{A}^{(k)}$. Since the first row of $\mathbf{A}_{22}^{(k)}$ and $\mathbf{M'}_k\mathbf{A}_{22}^{(k)}$ are the same, the first k rows of $\mathbf{A}^{(k)}$ and $\mathbf{A}^{(k+1)}$ are the same, while $\mathbf{M'}_k\mathbf{A}_{22}^{(k)}$ will have zeros in the first column under the diagonal.

Matrix Form

- We can describe also the partial pivoting using matrix form. Using the permutation matrices

$$
P_{kl} = \begin{pmatrix}
1 & & & & & & & & & & \\
 & \ddots & & & & & & & & & \\
 & & 1 & & & & & & & & \\
 & & & 0 & \cdots & \cdots & \cdots & 1 & - & - & k \\
 & & & \vdots & 1 & & & & & & \\
 & & & \vdots & & \ddots & & & & & \\
 & & & \vdots & & & 1 & & & & \\
 & & & 1 & \cdots & \cdots & \cdots & 0 & - & - & l \\
 & & & | & & & & | & 1 & & \\
 & & & | & & & & | & & \ddots & \\
 & & & k & & & & l & & & 1
\end{pmatrix}
$$

- Pre-multiplying any $n \times n$ matrix by P_{kl} will interchange files k and l

Matrix Form

- Permutation matrices can be described more generally as follows. The identity matrix can be represented as:

$$\mathbf{I_n} = (\mathbf{e_1}, \mathbf{e_2}, \cdots, \mathbf{e_n})$$

- This is, as a $n \times n$ matrix containing the ordered vectors of the canonical basis as column vectors.

Matrix Form

- Consider now, any permutation of the integers $1,2,\ldots,n$. Say $I=\{i_1,i_2,\ldots,i_n\}$. The matrix:

$$\mathbf{P_I}=\left(\mathbf{e_{i_1}},\mathbf{e_{i_2}},\cdots,\mathbf{e_{i_n}}\right)$$

- Is a permutation matrix that will reorder the rows of any matrix **A** when pre-multiplying.
- If the permutation is just a transposition of the indices (*k*,*l*) we obtain the matrix P_{kl}

Matrix Form

- Note the following properties of permutation matrices:

$$\mathbf{P_{kl}} = \mathbf{P_{lk}} = \mathbf{P^T{}_{kl}}$$
$$\mathbf{P_{kl}^{-1}} = \mathbf{P_{kl}}$$
$$\det(\mathbf{P_{kl}}) = -1$$

- Note also that the effect of post-multiplying any conformal matrix $\mathbf{A}$ by P_{kl} will interchange columns k and l.

Matrix Form

- Using the permutation and the Frobenius matrices we can describe the Gaussian elimination process as follows. When transforming $\mathbf{A}^{(p)}$ to obtain $\mathbf{A}^{(p+1)}$ we will follow the following steps:

$$\mathbf{M}_p \mathbf{P}_{pr_p} \mathbf{A}^{(p)} = \mathbf{A}^{(p+1)}$$

- And the complete process is:

$$\mathbf{A}^{(n)} = \mathbf{M}_{n-1} \mathbf{P}_{(n-1)r_{(n-1)}} \cdots \mathbf{M}_2 \mathbf{P}_{2r_2} \mathbf{M}_1 \mathbf{P}_{1r_1} \mathbf{A}^{(1)}$$

Matrix Form

- These operations can be performed separately on the system matrix **A** and on the independent system vector **b**.
- The $\mathbf{M_k}$ matrices are non-singular, and they have inverses. The resultant matrix is an upper triangular matrix $\mathbf{A^{(n)}} = \mathbf{U}$. Hence

$$\mathbf{A} = \mathbf{P}_{\mathbf{1r_1}}\mathbf{M}_{\mathbf{1}}^{\mathbf{-1}} \cdots \mathbf{P}_{\mathbf{n-1,r_{n-1}}}\mathbf{M}_{\mathbf{n-1}}^{\mathbf{-1}}\mathbf{U} = \mathbf{LU}$$

- Where **L** is the product of lower triangular matrices.

LU-Decomposition

- Using Gaussian elimination, we can solve simultaneously several systems. In many situations, however, the independent terms are not always available from the beginning.
- We may want to solve the systems $\mathbf{Ax}_1 = \mathbf{b}_1$ and $\mathbf{Ax}_2 = \mathbf{b}_2$ where $\mathbf{b}_2$ can be some function of $\mathbf{x}_1$. In this situation we should start the elimination from the beginning, at a considerable additional cost.

LU-Decomposition

- Suppose we can find a decomposition of $\mathbf{A}$ into a lower and an upper triangular matrix:

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

- Then the system $\mathbf{Ax} = \mathbf{b}$ is equivalent to the system $\mathbf{LUx} = \mathbf{b}$, which decomposes into two triangular systems.

$$\mathbf{Ly} = \mathbf{b} \quad \text{and} \quad \mathbf{Ux} = \mathbf{y}$$

- We could then solve $\mathbf{Ax} = \mathbf{b}$ with $2 \cdot 1/2n^2$ operations

LU Theorem

- ***LU Theorem***. Let $\mathbf{A}$ be a given $n \times n$ matrix and denote by $\mathbf{A}_k$ the $k \times k$ matrix formed by the intersection of the first k tows and columns in $\mathbf{A}$. If $det(\mathbf{A}_k) \neq 0, k = 1,2, \dots, n-1$, then there exist a unique lower-triangular matrix $\mathbf{L} = (l_{ij})$ with $l_{ii} = 1$, $i = 1,2, \dots, n$ and a unique upper-triangular matrix $\mathbf{U} = (u_{ij})$ so that $\mathbf{LU} = \mathbf{A}$

LU Theorem

- The proof is by induction. For $n = 1$, the decomposition $a_{11} = 1 \cdot u_{11} = l_{11} \cdot u_{11}$ is unique. Suppose the theorem is true for $n = k - 1$. For $n = k$, we partition $\mathbf{A_k}, \mathbf{L_k}$, and $\mathbf{U_k}$ according to:

$$\mathbf{A}_k = \begin{pmatrix} \mathbf{A}_{k-1} & \mathbf{b} \\ \mathbf{c}^{\mathrm{T}} & a_{kk} \end{pmatrix}, \quad \mathbf{L}_k = \begin{pmatrix} \mathbf{L}_{k-1} & \mathbf{0} \\ \mathbf{l}^{\mathrm{T}} & 1 \end{pmatrix}, \quad \mathbf{U}_k = \begin{pmatrix} \mathbf{U}_{k-1} & \mathbf{u} \\ \mathbf{0} & u_{kk} \end{pmatrix}$$

- Where $\mathbf{b}, \mathbf{c}, \mathbf{l}$ and $\mathbf{u}$ are column vectors with $k - 1$ components

LU Theorem

- If we form the product $\mathbf{L_k U_k = A_k}$, we get:

$$\mathbf{L}_{k-1}\mathbf{U}_{k-1} = \mathbf{A}_{k-1}, \qquad \mathbf{L}_{k-1}\mathbf{u} = \mathbf{b}$$
$$\mathbf{l}^{\mathrm{T}}\mathbf{U}_{k-1} = \mathbf{c}^{\mathrm{T}}, \qquad \mathbf{l}^{\mathrm{T}}\mathbf{u} + u_{kk} = a_{kk}$$

- By the induction hypothesis, $\mathbf{L_{k-1}}$ and $\mathbf{U_{k-1}}$ are uniquely determined, and since:

$$\det\left(\mathbf{L}_{k-1}\right) \cdot \det\left(\mathbf{U}_{k-1}\right) = \det\left(\mathbf{A}_{k-1}\right) \neq 0$$

- they are nonsingular

LU Theorem

- It follows that $\mathbf{u}$ and $\mathbf{l}$ are uniquely determined by the triangular systems $\mathbf{L}_{k-1}\mathbf{u} = \mathbf{b}$ and $\mathbf{U}_{k-1}^{T}\mathbf{l} = \mathbf{c}$. Finally:

$$u_{kk} = a_{kk} - \mathbf{l}^{\mathrm{T}}\mathbf{u}$$

- Thus, $\mathbf{L}_k$ and $\mathbf{U}_k$ are uniquely determined. Note that if, for some k, $\det(\mathbf{A}_k) = 0$, there may not exist and **LU** decomposition

LU Theorem

- On the other hand:

$$\mathbf{A} = \mathbf{LU} \Rightarrow \det(\mathbf{A}) = \det(\mathbf{L}) \cdot \det(\mathbf{U}) = \det(\mathbf{U})$$

- But:

$$\det(\mathbf{U}) = u_{11}u_{22}\cdots u_{nn} = \det(\mathbf{A})$$

- Then

$$\det(\mathbf{A}) \neq 0 \Leftrightarrow u_{ii} \neq 0, \quad i = 1, 2, \ldots, n$$

- And particularly

$$\det(\mathbf{A_k}) = u_{11}u_{22}\cdots u_{kk} \neq 0$$

Compact Schemes

- When applying gauss elimination, we modify repeatedly the elements $a_{ij}^{(k)}$ of matrix **A**. This will introduce the risk of rounding errors. It will be better to obtain the **L** and **U** matrices directly. Writing:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ l_{n1} & \cdots & \cdots & l_{nn} \end{pmatrix} \bullet \begin{pmatrix} u_{11} & \cdots & \cdots & u_{1n} \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix}$$

Compact Schemes

- We obtain:

$$a_{ij} = \sum_{i=1}^{p} l_{ir} u_{rj}, \qquad p = \min\{i, j\}$$

- Note that we have n^2 equations $n^2 + n$ unknowns. Thus, we have n degrees of freedom to fix freely.
- Fixing these n unknowns we obtain different compact algorithms to build the **LU** decomposition.

Algorithm of Doolittle

- If we fix $l_{ii} = 1, i = 1, \dots, n$ we obtain the following system of equations:

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

- If $k = min\{i, j\} = i$:

$$a_{kj} = \sum_{r=1}^{k} l_{kr} u_{rj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + l_{kk} u_{kj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + u_{kj}$$

- And for $k = 1, \dots, n$

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}$$

Algorithm of Doolittle

- On the other hand, if $k = min\{i, j\} = j$ we have

$$a_{ik} = \sum_{r=1}^{k} l_{ir} u_{rk} = \sum_{r=1}^{k-1} l_{ir} u_{rk} + l_{ik} u_{kk}$$

- And we can write an equation for the elements of matrix **L**:

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

Algorithm of Doolittle

- These two equations allow to compute the directly the coefficients of the **L** and **U** matrices.
- Note, however, that we need the u_{rk} values with $r = 1, \dots, k-1$ to compute the value of l_{ik}. The order is important.
- We start computing the first row of **U** and then the first column of **L**. Then the second row of **U** and the second column of **L** and so on.

Algorithm of Crout

- We obtain a similar schema if we fix initially the diagonal values of matrix U and we make $u_{kk} = 1, k = 1, \dots, n$. Then, if $k = min\{i, j\} = j$

$$a_{ik} = \sum_{r=1}^{k} l_{ir} u_{rk} = l_{ik} + \sum_{r=1}^{k-1} l_{ir} u_{rk}$$

- Or, for $k = 1, \dots, n$:

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

Algorithm of Crout

- And similarly, if $k = min\{i, j\} = i$ results:

$$a_{kj} = \sum_{r=1}^{k} l_{kr} u_{rj} = \sum_{r=1}^{k-1} l_{kr} u_{kj} + l_{kk} u_{kj}$$

- And for $k = 1, \ldots, n$ we obtain

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

- The order of computation is also important. We start computing the first column of matrix **L**, then the first row of matrix **U** and so on.

Pivoting

- The Gaussian elimination and **LU** factorizations are equivalent. The choice of a small pivot at the *k*th reduction step will cause digits in significant terms $a_{ij}^{(p)}$ and $b_i^{(p)}$ to be lost when the much larger terms $\pi_{ip}^{(p)} a_{pj}^{(p)}$ or $\pi_{ip}^{(p)} b_i^{(p)}$ are subtracted. This is the reason why a pivoting strategy needs to be adopted when the coefficient matrix is neither symmetric positive nor diagonally dominant.

Pivoting

- In Crout's schema we compute the u_{kj} values dividing by the l_{kk}. If these values are small, we can amplify the rounding errors. In this case we can rearrange the rows of matrix **L** before computing the u_{kj} values as in the case of Gaussian elimination.
- Note that these row reordering is like pre-multiplying the **A** matrix by a permutation matrix **P** in order to ensure that $\det(\mathbf{A}_k) > 0$ for all $k = 1, \ldots, n$.

Pivoting

- In the Doolittle algorithm we start computing the rows of the U matrix. To reduce the risk of error amplification we will need to reorder columns. We can avoid somewhat this problem if before computing the u_{kj} for $j = k, \dots, n$ we use the row f_k which satisfies

$$\left| a_{f_k k} - \sum_{r=1}^{k-1} l_{f_k r} u_{rk} \right| = \max_{k \le f \le n} \left| a_{fk} - \sum_{r=1}^{k-1} l_{fr} u_{rk} \right|$$

- And then interchanging rows k and f_k.

Symmetric Matrices

- Symmetric matrices appear in many engineering problems.
- For symmetric matrices we will seek a factorization of the form $\mathbf{A} = \mathbf{LDL^T}$, where $\mathbf{L}$ is a lower unitary triangular matrix

$$A = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{nn-1} & 1 \end{pmatrix} \begin{pmatrix} d_{11} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_{nn} \end{pmatrix} \begin{pmatrix} 1 & l_{21} & \cdots & l_{n1} \\ 0 & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & l_{nn-1} \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Symmetric Matrices

- This is equivalent to a **LU** factorization with an upper triangular $\mathbf{U} = \mathbf{D}\mathbf{L}^{\mathrm{T}}$. We have $u_{ij} = d_{ii} \cdot l_{ji}$ and in particular $u_{ii} = d_{ii}$. Using the Doolittle formulation, we obtain:

$$d_{kk} = u_{kk} = a_{kk} - \sum_{r=1}^{k-1} l_{kr} u_{rk} = a_{kk} - \sum_{r=1}^{k-1} l_{kr}^2 d_{rr}, \quad k = 1, \ldots, n$$

- And for $k = 1, \ldots, n$

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

- While $l_{kk} = 1$.

Cholesky Factorization

- ***Theorem***. If $\mathbf{A}$ is a symmetric definite positive matrix:

$$\mathbf{x}^{T}\mathbf{A}\mathbf{x} > 0, \quad \text{for all } \mathbf{x} \neq \mathbf{0}.$$

- Then there is a unique lower triangular matrix with all diagonal elements positive, such that:

$$\mathbf{A} = \mathbf{L}^{T}\mathbf{L}$$

- This result is known as the Cholesky factorization.

Cholesky Factorization

- To obtain the algorithm we only need to compute the elements of the matrix **L**. We have:

$$\begin{pmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{nn-1} & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & l_{nn-1} \\ 0 & \cdots & 0 & l_{nn} \end{pmatrix}$$

- Note that the symmetric matrix **A** has $\frac{n(n-1)}{2}$ different entries, the same than for matrix **L**.

Cholesky Factorization

- Then if $k = min\{i, j\}$

$$a_{ik} = \sum_{r=1}^{k} l_{ir} l_{kr} = \sum_{r=1}^{k-1} l_{ir} l_{kr} + l_{ir} l_{kk}$$

- And for $k = 1, \ldots, n$ and $i = k + 1, \ldots, n$ we have:

$$l_{ik} = \frac{1}{l_{kk}} \left(a_{ik} + \sum_{r=1}^{k-1} l_{ir} l_{kr} \right)$$

Cholesky Factorization

- Finally, as we have

$$a_{kk} = \sum_{r=1}^{k-1} l^2{}_{kr} + l^2{}_{kk}$$

- We obtain

$$l_{kk} = \left(a_{kk} - \sum_{r=1}^{k-1} l^2{}_{kr} \right)^{1/2}$$

- We will compute the elements of matrix **L** column by column.