NUMERICAL METHODS AND PROGRAMMING 2024/2025

Direct and Iterative methods for LSE (PROBLEMS 4)

1.— Write an inversion algorithm for lower triangular matrices of order n and semi-bandwidth l. The initial matrix should be stored in a vector, and its inverse should be stored on top of the initial matrix.

Solution 1.

The structure of the matrices will be a priori:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & & & & & & \\ a_{21} & a_{22} & & & & & \\ \vdots & & \ddots & & & & \\ a_{1+l,1} & \dots & \dots & a_{1+l,1+l} & & & \\ & \ddots & & & \ddots & & \\ & & a_{n,n-l} & \dots & \dots & a_{n,n} \end{bmatrix} \quad ; \quad \boldsymbol{A}^{-1} = \boldsymbol{X} = \underbrace{\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{bmatrix}}_{\text{full matrix}}$$

Since $\boldsymbol{X}$ is the inverse of matrix $\boldsymbol{A}$:

$$\boldsymbol{A}\,\boldsymbol{X} = \boldsymbol{I} \Leftrightarrow \left\{ \begin{array}{c} \boldsymbol{A}\,\boldsymbol{x}^k = \boldsymbol{\delta}^k \\ k = 1, \dots, n \end{array} \right\}, \text{where} \quad \boldsymbol{x}^k = \begin{bmatrix} x_{1k} \\ x_{2k} \\ \vdots \\ x_{nk} \end{bmatrix} \quad ; \quad \boldsymbol{\delta}^k = \begin{bmatrix} \delta_{1k} \\ \delta_{2k} \\ \vdots \\ \delta_{nk} \end{bmatrix}$$

Thus, obtaining the inverse matrix can be posed as solving n systems of equations with matrix A and independent term $\boldsymbol{\delta}^k$.

$$\left. \begin{array}{l} x_{kk} = 1/a_{kk} \\ x_{ik} = -\left(\displaystyle\sum_{j=k}^{i-1} a_{ij}\, x_{jk} \right) / a_{ii} \quad ; \quad i = k+1, \dots, n \end{array} \right\} k = 1, \dots, n$$

The above algorithm can be modified to store the initial matrix in a vector, and the solution on the same vector, though we will be forced to store matrix $\boldsymbol{A}$ as a full triangular matrix rather than in band form, since $\boldsymbol{X}$ is full.

To store $\boldsymbol{A}$ in a vector:

$$a_{\alpha\beta} \longrightarrow v_\gamma \quad \text{with} \quad \gamma = \frac{\alpha(\alpha-1)}{2} + \beta$$

Introducing the change:

$$\left. \begin{array}{l} v_{\frac{k(k-1)}{2}+k} = \dfrac{1}{v_{\frac{k(k-1)}{2}+k}} \\ v_{\frac{i(i-1)}{2}+k} = -\left(\displaystyle\sum_{j=max\{k,i-l\}}^{i-1} v_{\frac{i(i-1)}{2}+j}\; v_{\frac{j(j-1)}{2}+k} \right) / v_{\frac{i(i-1)}{2}+i} \quad ; \quad i = k+1, \dots, n \end{array} \right\} k = 1, \dots, n$$

2.— Write an algorithm to solve linear systems of the form $\boldsymbol{K}\,\boldsymbol{u} = \boldsymbol{f}$, where $\boldsymbol{K}$ is a symmetric and positive-definite matrix with semi-bandwidth s, stored in a vector. Use the most suitable storage scheme and solution method.

Solution 2.

In this case, since the matrix of the system is symmetric and positive definite, the most appropriate method is to use a Cholesky factorization and solve the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ as follows:

$$\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{L}^t \longrightarrow \boldsymbol{L}\underbrace{(\boldsymbol{D}\overbrace{(\boldsymbol{L}^t\,\boldsymbol{x})}^{\boldsymbol{y}})}_{\boldsymbol{z}} = \boldsymbol{b} \longrightarrow \left\{\begin{array}{lcl} \boldsymbol{L}\,\boldsymbol{z} & = & \boldsymbol{b} \\ & \searrow & \\ \boldsymbol{D}\,\boldsymbol{y} & = & \boldsymbol{z} \\ & \searrow & \\ \boldsymbol{L}^t\,\boldsymbol{x} & = & \boldsymbol{y} \end{array}\right.$$

The general factorization approach for a matrix $\boldsymbol{A}$ is posed as follows.
Let:

$$\boldsymbol{A}_k = \begin{bmatrix} a_{11} & \dots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \dots & a_{kk} \end{bmatrix} \quad ; \quad \boldsymbol{A}_{k+1} = \left[\begin{array}{c|c} \boldsymbol{A}_k & \boldsymbol{f}_{k+1} \\ \hline \boldsymbol{f}_{k+1}^t & a_{k+1,k+1} \end{array}\right] \quad ; \quad \text{with} \quad \boldsymbol{f}_{k+1}^t = [a_{k+1,1}, \dots, a_{k+1,k}]$$

$$\boldsymbol{L}_k = \begin{bmatrix} l_{11} & \dots & l_{1k} \\ \vdots & & \vdots \\ l_{k1} & \dots & l_{kk} \end{bmatrix} \quad ; \quad \boldsymbol{L}_{k+1} = \left[\begin{array}{c|c} \boldsymbol{L}_k & \boldsymbol{0} \\ \hline \boldsymbol{l}_{k+1}^t & l_{k+1,k+1} \end{array}\right] \quad ; \quad \text{with} \quad \boldsymbol{l}_{k+1}^t = [l_{k+1,1}, \dots, l_{k+1,k}]$$

$$\boldsymbol{D}_k = \begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{kk} \end{bmatrix} \quad ; \quad \boldsymbol{D}_{k+1} = \left[\begin{array}{c|c} \boldsymbol{D}_k & \boldsymbol{0} \\ \hline \boldsymbol{0} & d_{k+1,k+1} \end{array}\right]$$

If we assume that $\boldsymbol{L}_k$ and $\boldsymbol{D}_k$ are known such that $\boldsymbol{A}_k = \boldsymbol{L}_k\boldsymbol{D}_k\boldsymbol{L}_k^t$, we can impose that $\boldsymbol{A}_{k+1} = \boldsymbol{L}_{k+1}\boldsymbol{D}_{k+1}\boldsymbol{L}_{k+1}^t$, i.e.,

$$\left[\begin{array}{c|c} \boldsymbol{A}_k & \boldsymbol{f}_{k+1} \\ \hline \boldsymbol{f}_{k+1}^t & a_{k+1,k+1} \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{L}_k\boldsymbol{D}_k\boldsymbol{L}_k^t & \boldsymbol{L}_k\boldsymbol{D}_k\boldsymbol{l}_{k+1} \\ \hline \boldsymbol{l}_{k+1}^t\boldsymbol{D}_k\boldsymbol{L}_k^t & \boldsymbol{l}_{k+1}^t\boldsymbol{D}_k\boldsymbol{l}_{k+1} + l_{k+1,k+1}^2 d_{k+1,k+1} \end{array}\right]$$

Thus, we must have:

$$\left\{\begin{array}{l} \boldsymbol{L}_k\boldsymbol{D}_k\boldsymbol{l}_{k+1} = \boldsymbol{f}_{k+1} \\ \\ a_{k+1,k+1} = \boldsymbol{l}_{k+1}^t\boldsymbol{D}_k\boldsymbol{l}_{k+1} + l_{k+1,k+1}^2 d_{k+1,k+1} \end{array}\right.$$

Then:

- $\boldsymbol{l}_{k+1}$ is obtained by solving the system $(\boldsymbol{L}_k\boldsymbol{D}_k)\boldsymbol{l}_{k+1} = \boldsymbol{f}_{k+1}$
- $l_{k+1,k+1} = 1$ arbitrarily.
- $d_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=1}^{k} l_{k+1,j}^2\, d_{jj}$

For $k = 1$, we directly obtain that $d_{11} = a_{11}$ and $l_{11} = 1$. Therefore, the general factorization algorithm for a symmetric matrix will be:

$$l_{11} = 1 \quad ; \quad d_{11} = a_{11}$$

do $k = 1, n-1$

$$l_{k+1,i} = a_{k+1,i} - \sum_{j=1}^{i-1} l_{ij}\, l_{k+1,j}; \qquad i = 1, \ldots, k$$

$$l_{k+1,i} = l_{k+1,i}/d_{ii} \qquad i = 1, \ldots, k$$

$$l_{k+1,k+1} = 1$$

$$d_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=1}^{k} l_{k+1,j}^2\, d_{jj}$$

enddo

And the resolution of the system:

$$z_i = b_i - \sum_{j=1}^{i-1} l_{ij}\, z_j \quad ; \quad i = 1, \ldots, n$$

$$y_i = z_i/d_{ii} \quad ; \quad i = 1, \ldots, n$$

$$x_i = y_i - \sum_{j=i+1}^{n} l_{ji}\, x_j \quad ; \quad i = n, \ldots, 1 \quad \text{(Back substitution)}$$

Since the diagonal of $\boldsymbol{L}$ and the elements of $\boldsymbol{A}$ are not reused, it is possible to store the information of the matrices $\boldsymbol{L}$ and $\boldsymbol{D}$ on top of $\boldsymbol{A}$. The same applies to the vectors $\boldsymbol{z}$, $\boldsymbol{y}$, and $\boldsymbol{x}$ as well as the independent term of the system $\boldsymbol{b}$.

Taking this into account, the factorization and solution of the system would be:

$$\text{do } k = 1, \ldots, n-1$$

$$a_{k+1,i} = a_{k+1,i} - \sum_{j=1}^{i-1} a_{ij}\, a_{k+1,j}; \qquad i = 2, \ldots, k$$

$$a_{k+1,i} = a_{k+1,i}/a_{ii} \qquad ; i = 1, \ldots, k$$

$$a_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=1}^{k} a_{k+1,j}^2\, a_{jj}$$

$$\text{enddo}$$

$$b_i = b_i - \sum_{j=1}^{i-1} a_{ij}\, b_j \quad ; \quad i = 2, \ldots, n$$

$$b_i = b_i/a_{ii} \quad ; \quad i = 1, \ldots, n$$

$$b_i = b_i - \sum_{j=i+1}^{n} a_{ji}\, b_j \quad ; \quad i = n-1, \ldots, 1 \quad \text{(Back substitution)}$$

It can also be verified that the semi-bandwidth of matrix $\boldsymbol{L}$ is the same as that of $\boldsymbol{A}$. Therefore, both before and after factorization, the elements a_{ij} outside the band are zero. By modifying the loops to avoid working with those elements, we get:

$$\text{do } k = 1, n-1$$

$$a_{k+1,i} = a_{k+1,i} - \sum_{j=\max\{i-s,k+1-s,1\}}^{i-1} a_{ij}\, a_{k+1,j} \quad ; \quad i = \max\{k+1-s+1, 2\}, \ldots, k$$

$$a_{k+1,i} = a_{k+1,i}/a_{ii} \quad ; \quad i = \max\{k+1-s, 1\}, \ldots, k$$

$$a_{k+1,k+1} = a_{k+1,k+1} - \sum_{j=\max\{k+1-s,1\}}^{k} a_{k+1,j}^2\, a_{jj}$$

$$\text{enddo}$$

$$b_i = b_i - \sum_{j=\max\{i-s,1\}}^{i-1} a_{ij}\, b_j \quad ; \quad i = 2, \ldots, n$$

$$b_i = b_i/a_{ii} \quad ; \quad i = 1, \ldots, n$$

$$b_i = b_i - \sum_{j=i+1}^{\min\{i+s,n\}} a_{ji}\, b_j \quad ; \quad i = n-1, \ldots, 1 \quad \text{(Back substitution)}$$

Additionally, we can store the banded matrix $\boldsymbol{A}$ in a vector such that the elements $a_{\alpha,\beta}$ are stored in a vector v_γ with $\gamma = (\alpha-1)(1+s) + (\beta - \alpha + 1 + s) = \beta + \alpha \cdot s$. Thus:

do $k = 1, n-1$

$$v_{(k+1)s+i} = v_{(k+1)s+i} - \sum_{j=\max\{i-s,k+1-s,1\}}^{i-1} v_{is+j}\, v_{(k+1)s+j} \quad ; \quad i = \max\{k+1-s+1, 2\}, \ldots, k$$

$$v_{(k+1)s+i} = v_{(k+1)s+i} / v_{is+i} \quad ; \quad i = \max\{k+1-s, 1\}, \ldots, k$$

$$v_{(k+1)s+(k+1)} = v_{(k+1)s+(k+1)} - \sum_{j=\max\{k+1-s,1\}}^{k} v_{(k+1)s+j}^2\, v_{js+j}$$

enddo

$$b_i = b_i - \sum_{j=\max\{i-s,1\}}^{i-1} v_{is+j}\, b_{(p-1)n+j} \quad ; \quad i = 2, \ldots, n$$

$$b_i = b_i / v_{is+i} \quad ; \quad i = 1, \ldots, n$$

$$b_i = b_i - \sum_{j=i+1}^{\min\{i+s,n\}} v_{js+i}\, b_{(p-1)n+j} \quad ; \quad i = n-1, \ldots, 1 \quad \text{(Back substitution)}$$

3.— Generalize the previous algorithm for solving m systems of equations with the same matrix and different right-hand sides $\{\boldsymbol{f}^1, \boldsymbol{f}^2, \ldots, \boldsymbol{f}^m\}$, where these are stored in a vector.

Solution 3.

We are asked to solve m systems of equations of the form:

$$\boldsymbol{A}\,\boldsymbol{x}^p = \boldsymbol{b}^p \qquad p = 1, \ldots, m$$

Since the matrix is the same, the factorization is identical to the previous problem and is done only once. Then, we only need to repeat the solution of the systems m times.

If we store the m right-hand sides in a single vector $\boldsymbol{g}$, the components f_α^p will be stored in g_γ with $\gamma = (p-1)n + \alpha$. The solution algorithm would then be:

do $p = 1, \ldots, m$

$$b_{(p-1)n+i} = b_{(p-1)n+i} - \sum_{j=\max\{i-s,1\}}^{i-1} v_{is+j}\, b_{(p-1)n+j} \quad ; \quad i = 2, \ldots, n$$

$$b_{(p-1)n+i} = b_{(p-1)n+i} / v_{is+i} \quad ; \quad i = 1, \ldots, n$$

$$b_{(p-1)n+i} = b_{(p-1)n+i} - \sum_{j=i+1}^{\min\{i+s,n\}} v_{js+i}\, b_{(p-1)n+j} \quad ; \quad i = n-1, \ldots, 1 \quad \text{(Back substitution)}$$

enddo

4.— To analyze a certain engineering problem, it is necessary to solve systems of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, where the matrix $\boldsymbol{A}$ is symmetric, positive definite, and has the form:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & & & & & & \\ a_{21} & a_{22} & & & & SYM & \\ a_{31} & 0 & a_{33} & & & & \\ a_{41} & 0 & 0 & a_{44} & & & \\ a_{51} & 0 & 0 & 0 & a_{55} & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ a_{n1} & 0 & \dots & \dots & \dots & 0 & a_{nn} \end{bmatrix}$$

$1000 \leq n \leq 5000$
$|a_{ii}| >> |a_{ij}| \ \forall j \neq i$

To solve the system of equations, a direct method is preferred, storing the intermediate operations on the matrix itself. Required:

a) What is the most economical storage scheme that can be used in these conditions? Why?

b) Which direct method is suitable to use in these conditions? Why?

c) Could it be advantageous, in terms of calculation time or storage, to use an iterative algorithm? Which one? Why?

d) Could it be advantageous, in terms of calculation time or storage, to use a semi-iterative algorithm? Which one? Why?

e) If the equations and/or unknowns are appropriately renumbered, is it possible to reduce the computational cost and storage associated with using a direct method? How?

Solution 4.a

If a direct method is used to solve the problem (Gaussian elimination for symmetric matrices or Cholesky factorization), the zero elements will no longer be zero during the solution process, so it will be necessary to store these zeros. Thus, the matrix should be stored as symmetric, either the lower triangular part or the upper triangular part, using $n(n+1)/2$ elements.

Solution 4.b

Both Gaussian elimination for symmetric matrices and Cholesky factorization would be suitable. In general, Cholesky factorization is preferred. Both methods will work since the matrix is positive definite.

Solution 4.c

The matrix is diagonally dominant ($|a_{ii}| >> |a_{ij}| \ \forall j \neq i$), so the Jacobi and Gauss-Seidel methods will converge.

It may be advantageous to use these methods because many elements of the matrix are zeros, meaning they do not need to be considered. The computational cost per iteration would then be T(n) compared to the computational cost T(n^3) of a direct method for a full matrix.

Solution 4.d

The matrix is positive definite, so the Conjugate Gradient method will converge to the solution. For the same reason mentioned in the previous section, the cost per iteration will be T(n). At most, n iterations will be carried out, so the total cost of a semi-iterative method would be T(n^2), which is always advantageous.

Solution 4.e

We can renumber the equations and unknowns as follows:

$$\begin{bmatrix} a_{nn} & & & & & & a_{n1} \\ & \ddots & & & & & \vdots \\ & & a_{55} & & & & a_{51} \\ & & & a_{44} & & & a_{41} \\ & & & & a_{33} & & a_{31} \\ & SYM & & & & a_{22} & a_{21} \\ & & & & & & a_{11} \end{bmatrix} \begin{bmatrix} x_n \\ \vdots \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_n \\ \vdots \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \end{bmatrix}$$

In this way, the system of equations remains the same, but the system matrix now has a symmetric profile, meaning it only requires storing $2n-1$ components, and the computation time becomes T(n).

5.— Derive the sufficient condition for the iterative algorithm

$$\boldsymbol{B}^{k+1} = \boldsymbol{B}^k[2\,\boldsymbol{I} - \boldsymbol{A}\,\boldsymbol{B}^k]; \qquad k = 0, \ldots$$

to converge to the inverse of matrix $\boldsymbol{A}$, with $\boldsymbol{B}^0$ as an initial approximation to $\boldsymbol{A}^{-1}$ such that:

$$\boldsymbol{A}\,\boldsymbol{B} = \boldsymbol{I} + \boldsymbol{E}; \qquad (\boldsymbol{E} = \text{error matrix})$$

Solution 5.

We define the error in iteration k as:

$$\boldsymbol{E}_k = \boldsymbol{A}\,\boldsymbol{B}_k - \boldsymbol{I}$$

therefore:

$$\begin{aligned} \boldsymbol{E}_{k+1} &= \boldsymbol{A}\,\boldsymbol{B}_{k+1} - \boldsymbol{I} = \boldsymbol{A}\,\boldsymbol{B}_k[2\boldsymbol{I} - \boldsymbol{A}\,\boldsymbol{B}_k] - \boldsymbol{I} \\ &= (\boldsymbol{E}_k + \boldsymbol{I})[2\boldsymbol{I} - (\boldsymbol{E}_k + \boldsymbol{I})] - \boldsymbol{I} \\ &= (\boldsymbol{E}_k + \boldsymbol{I})[\boldsymbol{E}_k + \boldsymbol{I}] - \boldsymbol{I} \\ &= -(\boldsymbol{E}_k)^2 + \boldsymbol{E}_k - \boldsymbol{E}_k + \boldsymbol{I} - \boldsymbol{I} \\ &= -(\boldsymbol{E}_k)^2 \end{aligned}$$

Thus, the error:

$$\boldsymbol{E}_0 = \boldsymbol{E}$$

$$\boldsymbol{E}_1 = -(\boldsymbol{E}_0)^2 = -\boldsymbol{E}^2$$

$$\boldsymbol{E}_2 = -(\boldsymbol{E}_1)^2 = -\boldsymbol{E}^4$$

$$\boldsymbol{E}_3 = -(\boldsymbol{E}_2)^2 = -\boldsymbol{E}^8$$

$$\vdots$$

$$\boldsymbol{E}_k = -(\boldsymbol{E}_{k-1})^2 = -\boldsymbol{E}^{(2^k)}$$

The convergence condition will be:

$$\lim_{k\to\infty} \boldsymbol{B}_k = \boldsymbol{A}^{-1} \Leftrightarrow \lim_{k\to\infty} \boldsymbol{E}_k = \boldsymbol{0} \Leftrightarrow \rho(\boldsymbol{E}) < 1$$

where $\rho(\boldsymbol{E})$ is the spectral radius of the matrix $\boldsymbol{E}$.
We can also check the order of convergence:

$$\begin{aligned}(\boldsymbol{B}_{k+1} - \boldsymbol{A}^{-1}) &= \boldsymbol{A}^{-1}(\boldsymbol{A}\,\boldsymbol{B}_{k+1} - \boldsymbol{I}) = \boldsymbol{A}^{-1}\boldsymbol{E}_{k+1} = -\boldsymbol{A}^{-1}(\boldsymbol{E}_k)^2 \\ &= -\boldsymbol{A}^{-1}(\boldsymbol{A}\,\boldsymbol{B}_k - \boldsymbol{I})^2 = -\boldsymbol{A}^{-1}(\boldsymbol{A}(\boldsymbol{B}_k - \boldsymbol{A}^{-1}))^2 \\ &= -\boldsymbol{A}^{-1}\,\boldsymbol{A}(\boldsymbol{B}_k - \boldsymbol{A}^{-1})\boldsymbol{A}(\boldsymbol{B}_k - \boldsymbol{A}^{-1}) = -(\boldsymbol{B}_k - \boldsymbol{A}^{-1})\boldsymbol{A}(\boldsymbol{B}_k - \boldsymbol{A}^{-1})\end{aligned}$$

$$\Rightarrow ||\boldsymbol{B}_{k+1} - \boldsymbol{A}^{-1}|| \leq ||\boldsymbol{A}||\,||\boldsymbol{B}_k - \boldsymbol{A}^{-1}||^2 \longrightarrow \text{Quadratic Convergence}$$

6.— To solve the system of equations:

$$\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 6 \\ 9 \end{Bmatrix}$$

The goal is to use the over-relaxed Gauss-Seidel method. We are asked to:

a) Study the convergence of the algorithm as a function of the relaxation coefficient α used (α = constant in all iterations).

b) Is there an optimal value of the relaxation coefficient (constant in all iterations) to make the convergence faster? If so, find it. (Suggestion: graph the spectral radius of the corresponding matrix as a function of the relaxation coefficient).

c) Perform the first five iterations for $\alpha = 1$ and for $\alpha = \frac{32}{31}$, starting from the initial approximation $x_1^0 = 0$, $x_2^0 = 0$.

d) In practice, could such an analysis be done for a system of several thousand equations? Why?

Solution 6.a

The scheme of the over-relaxed Gauss-Seidel algorithm will be:

$$\boldsymbol{x}k+1 = \boldsymbol{x}^k + \alpha\, \boldsymbol{C}^{-1}(\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}^k) \quad ; \quad \text{with} \quad \boldsymbol{C} = \begin{bmatrix} 4 & 0 \\ 1 & 4 \end{bmatrix}$$

We define the error at iteration k as:

$$\boldsymbol{e}^k = \boldsymbol{x} - \boldsymbol{x}^k \quad \longrightarrow \quad \boldsymbol{e}^{k+1} = \left[\boldsymbol{I} - \alpha\, \boldsymbol{C}^{-1}\boldsymbol{A}\right] \boldsymbol{e}^k$$

Therefore, the convergence condition is $\rho(\boldsymbol{I} - \alpha\, \boldsymbol{C}^{-1}\boldsymbol{A}) < 1$, and the smaller it is, the faster the convergence will be.

To study the convergence, we examine the eigenvalues of $(\boldsymbol{I} - \alpha\, \boldsymbol{C}^{-1}\boldsymbol{A})$ as a function of α:

$$(\boldsymbol{I} - \alpha\, \boldsymbol{C}^{-1}\boldsymbol{A})\boldsymbol{u} = \lambda \boldsymbol{u} \quad \Leftrightarrow \quad det(\boldsymbol{I} - \alpha\, \boldsymbol{C}^{-1}\boldsymbol{A} - \lambda\, \boldsymbol{I}) = 0 \Leftrightarrow$$

$$\Leftrightarrow det((1-\lambda)\boldsymbol{C} - \alpha\boldsymbol{A}) = 0 \quad \Leftrightarrow \quad det(\alpha\boldsymbol{A} + (\lambda-1)\boldsymbol{C}) = 0 \Leftrightarrow$$

$$\Leftrightarrow \begin{vmatrix} 4\alpha + (\lambda-1)4 & 1\alpha + (\lambda-1)0 \\ 1\alpha + (\lambda-1)1 & 4\alpha + (\lambda-1)4 \end{vmatrix} = \begin{vmatrix} 4(\alpha+\lambda-1) & \alpha \\ (\alpha+\lambda-1) & 4(\alpha+\lambda-1) \end{vmatrix} = 0 \Leftrightarrow$$

$$\Leftrightarrow 16(\alpha+\lambda-1)^2 - \alpha(\alpha+\lambda-1) = 0 \quad \Leftrightarrow \quad \begin{cases} (\alpha+\lambda-1) = 0 \quad \longrightarrow \quad \lambda = 1-\alpha \\ \text{or} \\ 16(\alpha+1-1) - \alpha = 0 \quad \longrightarrow \quad \lambda = 1 - \frac{15\alpha}{16} \end{cases}$$

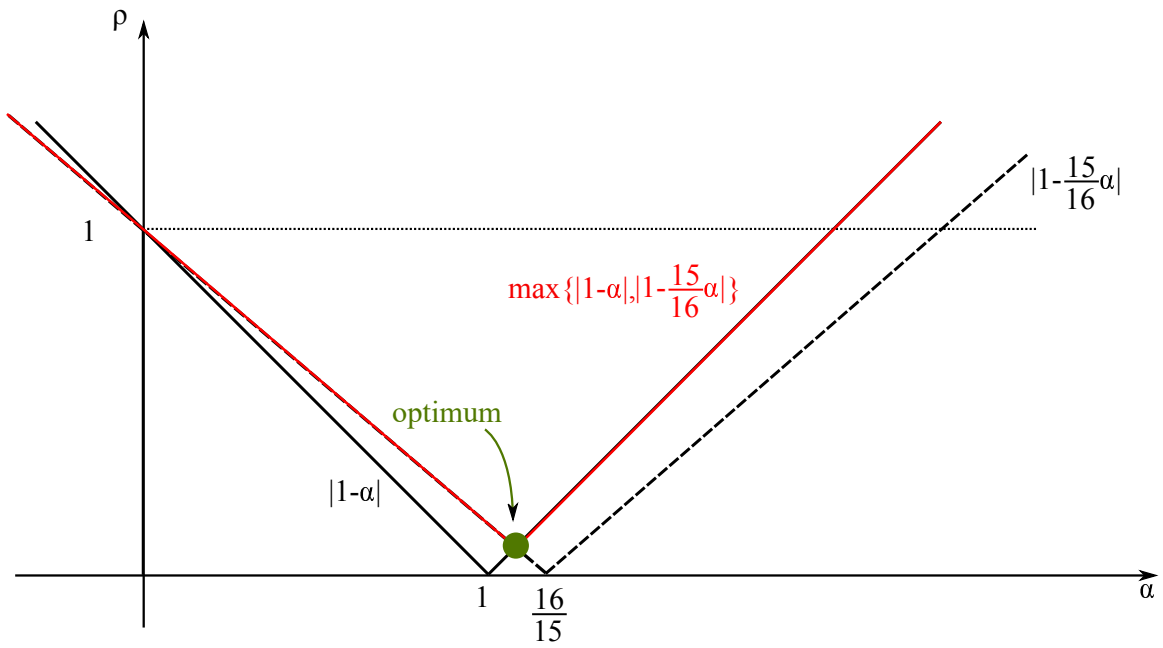
$$\rho(\boldsymbol{I} - \alpha\boldsymbol{C}^{-1}\boldsymbol{A}) = max\left\{|1-\alpha|, \left|1 - \frac{15\alpha}{16}\right|\right\}$$

Thus, for the algorithm to converge:

$$\left\{ \begin{array}{c} |1-\alpha| < 1 \Leftrightarrow -1 < 1-\alpha < 1 \Leftrightarrow -2 < -\alpha < 0 \\ and \\ \left|1 - \frac{15\alpha}{16}\right| < 1 \Leftrightarrow -1 < |1 - \frac{15\alpha}{16} < 1 \Leftrightarrow -2 < 1 - \frac{15\alpha}{16} < 0 \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{c} \alpha > 0 \\ \alpha < 2 \\ \alpha < \frac{32}{15} \end{array} \right\}$$

Therefore, the algorithm converges if $0 < \alpha < 2$.

Solution 6.b



The optimal value of α is the one that minimizes the spectral radius $\rho(\boldsymbol{I} - \alpha \boldsymbol{C}^{-1}\boldsymbol{A})$, that is:

$$\alpha - 1 = 1 - \frac{15\alpha}{16} \quad \Leftrightarrow \quad \alpha\left(1 + \frac{15}{16}\right) = 2 \quad \Leftrightarrow \quad \alpha = \frac{32}{31} = \alpha_{opt}$$

Solution 6.c

The most practical way to perform the calculations is as follows:

$$\left.\begin{array}{l} \boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha \boldsymbol{s}^k \\ \boldsymbol{C}\,\boldsymbol{s}^k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^k \end{array}\right\} \Rightarrow \left\{\begin{array}{l} \left\{\begin{array}{c} x_1^{k+1} \\ x_2^{k+1} \end{array}\right\} = \left\{\begin{array}{c} x_1^{k} \\ x_2^{k} \end{array}\right\} + \alpha \left\{\begin{array}{c} s_1^{k} \\ s_2^{k} \end{array}\right\} \\ \\ \left[\begin{array}{cc} 4 & 0 \\ 1 & 4 \end{array}\right] \left\{\begin{array}{c} s_1^{k} \\ s_2^{k} \end{array}\right\} = \left\{\begin{array}{c} 6 \\ 9 \end{array}\right\} - \left[\begin{array}{cc} 4 & 1 \\ 1 & 4 \end{array}\right] \left\{\begin{array}{c} x_1^{k} \\ x_2^{k} \end{array}\right\} \end{array}\right.$$

$$\alpha = 1 \Rightarrow \left\{\begin{array}{l} x_1^{k+1} = -x_2^k/4 + 3/2 \\ x_2^{k+1} = x_2^k/16 + 15/8 \end{array}\right.$$

$$\alpha = 32/31 \Rightarrow \left\{\begin{array}{l} x_1^{k+1} = 48/31 - 1/31\, x_1^k - 8/31\, x_2^k \\ x_2^{k+1} = 60/31 + 2/31\, x_2^k \end{array}\right.$$

Performing the first 5 iterations starting from the initial solution provided:

k	x_1^k ($\alpha = 1$)	x_2^k ($\alpha = 1$)	x_1^k ($\alpha = 32/31$)	x_2^k ($\alpha = 32/31$)
0	0.000000000	0.000000000	0.000000000	0.000000000
1	1.500000000	1.875000000	1.548387097	1.935483871
2	1.031250000	1.992187500	0.998959417	1.997918835
3	1.001953125	1.999511719	1.000570642	1.999932866
4	1.000122070	1.999969482	0.999998917	1.999997834
5	1.000007629	1.999998093	1.000000594	1.999999930

We observe that the results are very similar, and in both cases the method converges to the solution $\boldsymbol{x} = \{1, 2\}$.

It can also be observed that for the optimal α, the convergence is quadratic, although in the case of $\alpha = 1$, the speed is also fast.

Solution 6.d

For large systems, it is not possible to perform an analytical study as in the previous one because it is not feasible to analytically obtain the spectral radius of the matrix. However, it is possible to perform some iterations for different values of α for a system with the same matrix and known solution (for example, $\boldsymbol{Ax} = \boldsymbol{0}$), in order to experimentally obtain a value of α that results in a reasonably high convergence speed.

7.— Let a continuous beam be formed by n spans. The spans and supports are numbered consecutively, so that span $e \in \{1, \ldots, n\}$ extends from support $e - 1$ to support e. Let E be the elasticity modulus of the material, and let I_e and L_e be the moment of inertia of the section and the length corresponding to the e-th span, respectively. Given the moments $\{M_i\}_{i=0,\ldots,n}$ that act on the supports, the corresponding rotations $\{\omega_i\}_{i=0,\ldots,n}$ are to be calculated. The following is requested:

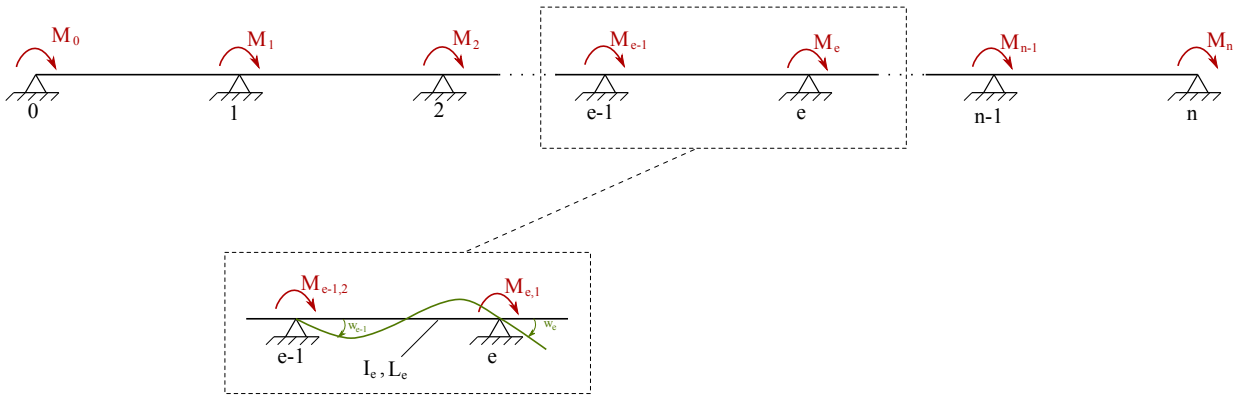
a) Formulate the system of linear equations that needs to be solved to calculate the rotations.

b) Verify that the coefficient matrix of the previous system is symmetric and positive definite.

c) Propose and fully develop the direct method considered most appropriate to solve the system.

d) Implement the selected method in a FORTRAN program that allows solving this type of problem.

e) Formulate the solution of the system using the iterative Gauss-Seidel method. Can it be ensured that this method will converge? Interpret the operation of the method from a structural point of view.



Solution 7.a

$$\left\{ \begin{array}{c} \omega_{e-1} = M_{e-1,2}\dfrac{L_e}{3EI_e} - M_{e,1}\dfrac{L_e}{6EI_e} \\ \omega_e = M_{e,1}\dfrac{L_e}{3EI_e} - M_{e-1,2}\dfrac{L_e}{6EI_e} \end{array} \right\} \Leftrightarrow$$

$$\Leftrightarrow \left\{ \begin{array}{c} \omega_{e-1} \\ \omega_e \end{array} \right\} = \frac{L_e}{6EI_e} \left[\begin{array}{cc} 2 & -1 \\ -1 & 2 \end{array} \right] \left\{ \begin{array}{c} M_{e-1,2} \\ M_{e,1} \end{array} \right\}$$

Therefore:

$$\Leftrightarrow \left\{ \begin{array}{c} M_{e-1,2} \\ M_{e,1} \end{array} \right\} = \frac{EI_e}{L_e} \left[\begin{array}{cc} 4 & 2 \\ 2 & 4 \end{array} \right] \left\{ \begin{array}{c} \omega_{e-1} \\ \omega_e \end{array} \right\}$$

By equilibrium of bending moments:

$$\left\{ \begin{array}{l} M_0 = M_{1,2} \\ M_i = M_{i,1} + M_{i,2} \quad ; \quad i = 1, \ldots, n-1 \\ M_n = M_{n,1} \end{array} \right.$$

Thus:

$$\overbrace{\left[\begin{array}{ccccccc} 4k_1 & 2k_1 & & & & & \\ 2k_1 & 4(k_1+k_2) & 2k_2 & & & & \\ & 2k_2 & 4(k_2+k_3) & 2k_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 2k_{n-1} & 4(k_{n-1}+k_n) & 2k_n & \\ & & & & 2k_n & 4k_n \end{array} \right]}^{\boldsymbol{K}} \overbrace{\left\{ \begin{array}{c} \omega_0 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_{n-1} \\ \omega_n \end{array} \right\}}^{\boldsymbol{\omega}} = \overbrace{\left\{ \begin{array}{c} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{array} \right\}}^{\boldsymbol{M}}$$

with $k_e = \frac{EI_e}{L_e}; e = 1, \ldots, n.$

Solution 7.b

As seen previously, the matrix $\boldsymbol{K}$ is symmetric.

Proving now that it is positive definite:

$$\boldsymbol{v}^t\,\boldsymbol{K}\,\boldsymbol{v} = \sum_{e=1}^{n} \left\{ \begin{array}{cc} v_{e-1} & v_e \end{array} \right\} \overbrace{\left[\begin{array}{cc} 4k_e & 2k_e \\ 2k_e & 4k_e \end{array} \right]}^{\boldsymbol{K}_e} \left\{ \begin{array}{c} v_{e-1} \\ v_e \end{array} \right\}$$

given that the element matrices $\boldsymbol{K}_e$ are all of them positive definite since their eigenvalues are $\lambda_1 = 6k_e$ and $\lambda_2 = 2k_e$, then:

$$\boldsymbol{v}^t\,\boldsymbol{K}\,\boldsymbol{v} \geq 0 \quad \forall \boldsymbol{v}$$

Then, at least, $\boldsymbol{K}$ is semi-definite positive.

The case $\boldsymbol{v}^t\,\boldsymbol{K}\,\boldsymbol{v} = 0$ appears if and only if:

$$\left\{ \begin{array}{cc} v_{e-1} & v_e \end{array} \right\} \left[\begin{array}{cc} 4k_e & 2k_e \\ 2k_e & 4k_e \end{array} \right] \left\{ \begin{array}{c} v_{e-1} \\ v_e \end{array} \right\} \quad \Leftrightarrow \quad v_{e-1} = v_e = 0 \quad ; \quad e = 1, \ldots, n$$

$$\Rightarrow \boldsymbol{v}^t\,\boldsymbol{K}\boldsymbol{v} > 0 \quad \forall \boldsymbol{v} \neq \boldsymbol{0}$$

Thus, $\boldsymbol{K}$ is positive definite.

Solution 7.c

Considering that the system matrix is symmetric, tridiagonal and positive definite, the most suited method is Cholesky decomposition adapted to tridiagonal matrices.

Decomposing $\boldsymbol{K} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^t$, the matrices will keep the bandwidth, therefore:

$$\boldsymbol{L} = \left[\begin{array}{ccccc} 1 & & & & \\ l_1 & 1 & & & \\ & l_2 & 1 & & \\ & & \ddots & \ddots & \\ & & & l_n & 1 \end{array} \right] \qquad \boldsymbol{D} = \left[\begin{array}{ccccc} d_0 & & & & \\ & d_1 & & & \\ & & d_2 & & \\ & & & \ddots & \\ & & & & d_n \end{array} \right]$$

So, the product of the three matrices will be:

$$\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}t = \left[\begin{array}{ccccc} d_0 & & & & \\ l_1 d_0 & l_1 d_0 l_1 + d_1 & & & SYM \\ & l_2 d_1 & l_2 d_1 l_2 + d_2 & & \\ & & \ddots & \ddots & \\ & & & l_n d_{n-1} & l_n d_{d-1} l_n + d_n \end{array} \right]$$

making the elements equal to those of $\boldsymbol{K}$:

$$
\begin{array}{l}
d_0 = 4\,k_1 \\
l_1\,d_0 = 2\,k_1 \qquad \longrightarrow \qquad l_1 = 2\,k_1/d_0 \\
l_1\,d_0\,l_1 + d_1 = 4(k_1+k_2) \qquad \longrightarrow \qquad d_1 = 4(k_1+k_2) - l_1\,d_0\,l_1 \\
l_2\,d_1 = 2\,k_2 \qquad \longrightarrow \qquad l_2 = 2\,k_2/d_1 \\
l_2\,d_1\,l_2 + d_2 = 4(k_2+k_3) \qquad \longrightarrow \qquad d_2 = 4(k_2+k_3) - l_2\,d_1\,l_2 \\
\vdots \\
l_{n-1}\,d_{n-2} = 2\,k_{n-1} \qquad \longrightarrow \qquad l_{n-1} = 2\,k_{n-1}/d_{n-2} \\
l_{n-1}\,d_{n-2}\,l_{n-1} + d_{n-1} = 4(k_{n-1}+k_n) \qquad \longrightarrow \qquad d_{n-1} = 4(k_{n-1}+k_n) - l_{n-1}\,d_{n-2}\,l_{n-1} \\
l_n\,d_{n-1} = 2\,k_n \qquad \longrightarrow \qquad l_n = 2\,k_n/d_{n-1} \\
l_n\,d_{n-1}\,l_n + d_n = 4\,k_n \qquad \longrightarrow \qquad d_n = 4\,k_n - l_n\,d_{n-1}\,l_n
\end{array}
$$

Therefore, the decomposition algorithm will be:

$$
\begin{array}{l}
d_0 = 4\,k_1 \\
\text{do } i = 1, n-1 \\
\quad l_i = 2\,k_i/d_{i-1} \\
\quad d_i = 4(k_i + k_{i+1}) - l_i\,d_{i-1}\,l_i \\
\text{enddo} \\
l_n = 2\,k_n/d_{n-1} \\
d_n = 4\,k_n - l_n\,d_{n-1}\,l_n
\end{array}
$$

The solution of the system is obtained following the scheme:

$$
\boldsymbol{K} = \boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{L}^t \longrightarrow \boldsymbol{L}\,\underbrace{(\boldsymbol{D}\,\overbrace{(\boldsymbol{L}^t\,\boldsymbol{\omega})}^{\boldsymbol{y}})}_{\boldsymbol{z}} = \boldsymbol{u} \longrightarrow \left\{ \begin{array}{lcl} \boldsymbol{L}\,\boldsymbol{z} & = & \boldsymbol{u} \\ & \searrow & \\ \boldsymbol{D}\,\boldsymbol{y} & = & \boldsymbol{z} \\ & \searrow & \\ \boldsymbol{L}^t\,\boldsymbol{\omega} & = & \boldsymbol{y} \end{array} \right.
$$

That, using only one vector, the algorithm is:

$$
\begin{array}{lll}
\omega_i = \omega_i - l_i\,\omega_{i-1} & ; \qquad & i = 1, \ldots, n \\
\omega_i = \omega_i / d_i & ; \qquad & i = 0, \ldots, n \\
\omega_i = \omega_i - l_{i+1}\,\omega_{i+1} & ; \qquad & i = n-1, \ldots, 0 \qquad \text{(backwards)}
\end{array}
$$

Solution 7.d

```
      IMPLICIT REAL*4 (A-H,O-Z)
      PARAMETER(MXN=100,MXKP=4*MXN+2)
      INTEGER*4 N
      DIMENSIONWORK(MXKP)

1     WRITE(6,2(A,$)')'NUMBER OF SPANS :'
      READ(5,*)N
      IF(N.GT.MXN)CALL CRITICALERROR('EXCESIVE NUMBER OF SPANS(N > MXN)'

      KPLIBRE = 1
      CALL DYNAMICALLOCATION(MXKP,KPLIBRE,N,KPK)
      CALL DYNAMICALLOCATION(MXKP,KPLIBRE,N,KPL)
      CALL DYNAMICALLOCATION(MXKP,KPLIBRE,N+1,KPD)
      CALL DYNAMICALLOCATION(MXKP,KPLIBRE,N+1,KPW)

      CALL READER(N,WORK(KPK),WORK(KPW)
      CALL SOLVER(N,WORK(KPK),WORK(KPL),WORK(KPD),WORK(KPW))
      CALL WRITER(N,WORK(KPW))

      END
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      SUBROUTINE READER(N,RK,W)
      IMPLICITREAL*8(A-H,O-Z)
      DIMENSION RK(N),W(0:N)

      DO I=1,N
        WRITE(6,100)I
100     FORMAT('SPAN',I5,'- - - - - -> STIFFNESS EI/L =',$)
        READ(5,*)RK(I)
      ENDDO

      DO I=1,N
        WRITE(6,110)I
110     FORMAT('NODE',I5,'- - - - - -> MOMENT =',$)
        READ(5,*)W(I)
      ENDDO

      RETURN
      END
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
      SUBROUTINE SOLVER(N,RK,RL,RD,W)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION RK(N),RL(N),RD(0:N),W(0:N)

      I=0
        RD(I)=4.0D+00*(RK(I+1)
      DO I=1,N-1
        RL(I)=2.0D+00*RK(I)/RD(I-1)
        RD(I)=4.0D+00*(RK(I)+RK(I+1))-RL(I)*RD(I-1)*RL(I)
      ENDDO
      I=N
        RL(I)=2.0D+00*RK(I)/RD(I-1)
        RD(I)=4.0D+00*(RK(I)-RL(I)*RD(I-1)*RL(I)

      DO I=1,N
        W(I)=W(I)-RL(I)*W(I-1)
      ENDDO
      DO I=0,N
        W(I)=W(I)/RD(I)
      ENDDO
      DO I=N-1,0,-1
      W(I)=W(I)-RL(I+1)*W(I+1)
      ENDDO

      RETURN
      END
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      SUBROUTINE WRITER(N,W)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION W(0:N)

      WRITE(6,200)
  200  FORMAT('NODEROTATIONS',/
     .      '==================',/
     .      'NODE ROT(RADIANS)',/
     .      '==================',/

      DO I=0,N
        WRITE(6,210)I,W(I)
  210  FORMAT(1X,I10,D15.6)
      ENDDO

      RETURN
      END
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
      SUBROUTINE DYNAMICALLOCATION(MXKP, KPLIBRE, NCOMPONENTES, KP

      KP = KPLIBRE
      KPLIBRE = KPLIBRE + NCOMPONENTES
      IF(KPLIBRE.GT.MXKP + 1)CALL CRITICALERROR('MEMORY INSUFFICIENT')

      RETURN
      END
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      SUBROUTINE CRITICALERROR(MESSAGE)
      CHARACTER * (*)MESSAGE

      WRITE(6, 100)MESSAGE
  100 FORMAT('ERROR :', A)

      STOP
      END
```

Solution 7.e

The solution of the system by the iterative Gauss-Seidel method would be:

$$\omega_0^{k+1} = (u_0 - 2\,k_1\,\omega_1^k)/(4\,k_1)$$

$$\omega_1^{k+1} = (u_1 - 2\,k_2\,\omega_0^{k+1} - 2\,k_2\,\omega_2^k)/(4(k_1 + k_2))$$

$$\vdots$$

$$\omega_i^{k+1} = (u_i - 2\,k_i\,\omega_{i-1}^{k+1} - 2\,k_{i+1}\,\omega_{i+1}^k)/(4(k_i + k_{i+1})) \qquad i = 1, \ldots, n-1$$

$$\vdots$$

$$\omega_{n-1}^{k+1} = (u_{n-1} - 2\,k_{n-1}\,\omega_{n-2}^{k+1} - 2\,k_n\,\omega_n^k)/(4(k_{n-1} + k_n))$$

$$\omega_n^k = (u_n - 2\,k_n\,\omega_{n-1}^{k+1})/(4\,k_n)$$

The method will work since the matrix $\boldsymbol{K}$ is diagonal dominant:

$$\begin{cases} |4\,k_1| > |2\,k_1| \\ |4(k_e + k_{e+1})| > |2\,k_e| + |2\,k_{e+1}| \\ |4\,k_n| > |2\,k_n| \end{cases}$$

From the structural analysis perspective the method can be understood as the following

- Some initial rotations are assumed.
- The following is repeated until convergence:
  - All supports are clamped with their current rotations.
  - Rotation is released in node 0 and its rotation is obtained.

– All supports are clamped again.
– Rotation is released in node 1 and its rotation is obtained.
– The process continues for the rest of the nodes.

This is the base of the former methods for structural analysis previous to matrix analysis of structures, such as Cross, Kani, etc.

8.— Let $\boldsymbol{A}$ be a symmetric and regular matrix of size n. We want to solve the solution to the following linear system of equations:

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

given $\boldsymbol{b}$:

a) Write the Gaussian algorithm without pivoting **avoiding useless operations**. Calculate the number of operations needed to solve the problem.

b) Can the Gauss algorithm with pivoting be used keeping the symmetry? Why?

Hint: Recall that at each step of the elimination process, when the terms of the k-th column below the pivot are canceled, only the following submatrix is recalculated.

$$\begin{pmatrix} a_{k+1,k+1} & \dots & a_{k+1,n} \\ \vdots & \ddots & \vdots \\ a_{n,k+1} & \dots & a_{n,n} \end{pmatrix}$$

Solution 8.a

Starting with the original system of equations:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \left\{ \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{array} \right\} = \left\{ \begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{array} \right\}$$

after the first normalization of the first equation and elimination of the next ones, the system has the following shape:

$$\begin{bmatrix} a_{11} & a_{12}/a_{11} & a_{13}/a_{11} & \dots & a_{1n}/a_{11} \\ 0 & a_{22} - a_{21}\frac{a_{12}}{a_{11}} & a_{23} - a_{21}\frac{a_{13}}{a_{11}} & \dots & a_{2n} - a_{21}\frac{a_{1n}}{a_{11}} \\ 0 & a_{32} - a_{31}\frac{a_{12}}{a_{11}} & a_{33} - a_{31}\frac{a_{13}}{a_{11}} & \dots & a_{3n} - a_{31}\frac{a_{1n}}{a_{11}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2} - a_{n1}\frac{a_{12}}{a_{11}} & a_{n3} - a_{n1}\frac{a_{13}}{a_{11}} & \dots & a_{nn} - a_{n1}\frac{a_{1n}}{a_{11}} \end{bmatrix} \left\{ \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{array} \right\} = \left\{ \begin{array}{c} b_1/a_{11} \\ b_2 - a_{21}\frac{b_1}{a_{11}} \\ b_3 - a_{31}\frac{b_1}{a_{11}} \\ \vdots \\ b_n - a_{n1}\frac{b_1}{a_{11}} \end{array} \right\}$$

It can be easily observed that if the matrix was initially symmetrical, the submatrix to be worked with in the next step is also symmetrical. The submatrices operated on at each step retain symmetry throughout the entire process.

Therefore we can modify Gauss's method to avoid storing off-diagonal elements twice. If we approach it by storing the upper triangular part of the matrix:

$$\text{Elimination}\begin{cases}\text{do } i=1,n-1\\ \quad\text{do } k=i+1,n\\ \quad\quad c=a_{ik}/a_ii\\ \quad\quad a_{kj}=a_{kj}-c\,a_{ij}\quad;\quad j=k,\ldots,n\\ \quad\text{enddo}\\ \text{enddo}\end{cases}$$

$$\text{Ind. term}\begin{cases}\text{do } i=1,n-1\\ \quad\text{do } k=i+1,n\\ \quad\quad c=a_{ik}/a_ii\\ \quad\quad v_k=b_k-c\,b_i\quad;\quad j=k,\ldots,n\\ \quad\text{enddo}\\ \text{enddo}\end{cases}$$

$$\text{Solution}\begin{cases}x_n=b_n/a_{nn}\\ x_i=\left(b_i-\sum_{j=i+1}^{n}a_{ij}\,x_j\right)/a_{ii};\quad i=n-1,\ldots,1\quad\text{(Backwards)}\end{cases}$$

Solution 8.b

We cannot pose a Gaussian algorithm with pivoting for symmetric matrices because pivoting destroys the symmetry of the matrix.

The Gaussian method is conceptually more complicated than the Cholesky method and has no advantage so it is not normally used. In general it is preferred to use the Cholesky method for systems with positive definite symmetric matrices.