

RIA

laboria

Institut de Recherche
d'Informatique
et d'Automatique

Domaine de Voluceau
Rocquencourt
B. P. 105-78150 - Le Chesnay
France
Tél. 954 90 20

laboratoire de recherche
en informatique
et automatique

SOME EXAMPLES OF IMPLEMENTATION AND OF APPLICATION OF THE FINITE ELEMENT METHOD

Bertrand MERCIER
Olivier PIRONNEAU

Rapport de Recherche N° 248

Août 1977



Chapter 3

Basic Concepts About the Implementation of the Finite
Element Method

Introduction

Our purpose is double. First to introduce the reader to the concepts necessary for the programming of a simple problem by elementary finite elements.

Then to introduce some more sophisticated methods necessary for the solution of large problems (that is problems requiring very much computing time and memory space).

We shall consider only linear problems, and direct methods for the solution of the final linear system. The case of non linear problems will be considered later, and the use of iterative methods for solving linear problems was considered in chapter 2.

Let us describe the main sections

- A - Basic concepts.
 - 1 - Form of the data necessary for the representation of a triangulation.
 - 2 - Stiffness submatrix of an element. Stiffness matrix. Algorithm for the "assembling" of the stiffness matrix.
 - 3 - Computation of a stiffness submatrix.
 - 4 - Solution of the linear system by the Choleski method - choice of a numbering for the nodes.
- B - Computation of the stiffness submatrix for more sophisticated elements.
- C - Various methods to compute automatically a triangulation of a domain.

- D - Description of two contradictory methods for the solution of large problems.
 - The frontal method
 - The sub-structure method

A - Basic Concepts

1. Form of the data necessary for the representation of a triangulation.

Let us consider a simple triangulation T .

The following informations are needed:

- The coordinates of the vertices.
- The knowledge of the vertices which are really joined.

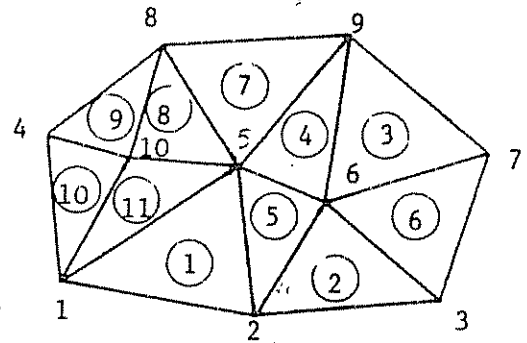


Fig. 1 Triangulation T .

For these purposes, it is convenient to introduce the following array: $M(3 \times n_T)$ where n_T is the number of triangles. We have $n_T = 11$ in the case of Fig. 1, and

$$M = \begin{pmatrix} 1 & 2 & 6 & 5 & 5 & 3 & 5 & 5 & 4 & 1 & 10 \\ 2 & 3 & 9 & 6 & 6 & 6 & 8 & 8 & 8 & 4 & 5 \\ 5 & 6 & 7 & 9 & 2 & 7 & 9 & 10 & 10 & 10 & 1 \end{pmatrix} = (m_{ij})_{\substack{1 \leq i \leq 3 \\ 1 \leq j \leq 10}}$$

m_{ij} is the number of the i^{th} vertex of the j^{th} triangle (see Fig. 1).

Let n_v be the number of vertices of the triangulation, m_{ij} is then an integer such that $1 \leq m_{ij} \leq n_v$.

We introduce then the array $Z = (z_{ij})_{\substack{1 \leq i \leq 2 \\ 1 \leq j \leq n_v}}$

such that z_{ij} is the i^{th} coordinate of the j^{th} vertex.

With these two data M and Z it is very easy to find the coordinates of the vertices of some triangle, and then to compute for example the area of this triangle.

As additional information about the triangulation, one may require to know if a vertex belongs to the boundary which can be specified by the data of a simple vector

$$L = (\ell_i)_{1 \leq i \leq n_v}$$

where $\ell_i = 1$ if the vertex i belongs to the boundary of the domain, and $\ell_i = 0$ otherwise.

In the following, we shall assume that the arrays M , Z and eventually L are given.

In fact the knowledge of the quantities M and Z is sufficient to draw the triangulation T , but it is sometimes easier to know the vector L .

Exercise 1:

Draw the triangulation T corresponding to the data

$$M = \begin{pmatrix} 2 & 2 & 6 & 3 & 4 & 4 & 1 & 5 \\ 7 & 5 & 5 & 6 & 6 & 9 & 7 & 7 \\ 5 & 8 & 8 & 8 & 5 & 5 & 9 & 9 \end{pmatrix}$$

$$Z = \begin{pmatrix} 0. & 1. & 1. & 0. & 0.5 & 0.5 & 0.5 & 1. & 0. \\ 0. & 0. & 1. & 1. & 0.5 & 1. & 0. & 0.5 & 0.5 \end{pmatrix}$$

Exercise 2:

Consider the domain $\Omega =]0,1[x]0,1[$, and the uniform mesh obtained by dividing Ω in $N \times N$ equal squares. Write the FORTRAN program which computes the data M and Z for the triangulation

obtained by dividing each square in two triangles by one of its diagonals.

(Note that in M, the order in which the 3 vertices of a triangle appear has no importance. However it can be convenient to order them anti-clockwise)

Other Types of Triangulations and Finite Elements

Let us consider the case of a "triangulation" made with quadrilateral elements (Fig. 2).

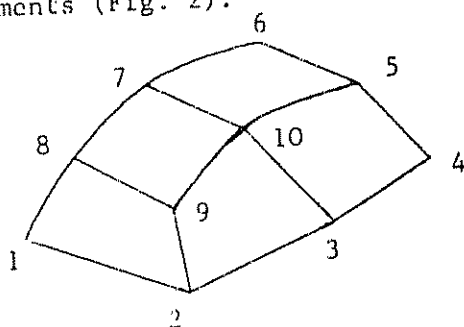


Fig. 2 A Triangulation composed with quadrilateral

In such a case the triangulation will be conveniently represented by the array

$$M = \begin{pmatrix} 1 & 2 & 3 & 5 & 9 \\ 2 & 3 & 4 & 6 & 10 \\ 9 & 10 & 5 & 7 & 7 \\ 8 & 9 & 10 & 10 & 8 \end{pmatrix}$$

(the array 7 will be the one you think). Note that the order in the columns of M is now essential, because in a quadrilateral one vertex is only connected to 2 other vertices.

For higher order elements it is convenient to number all the nodes corresponding to degrees of freedom, and not only vertices. As an example for Lagrange finite elements of order 2 (Fig. 3), the array M will be $6 \times n_T$ because there are 6 nodes per elements.

$$M = \begin{pmatrix} 1 & 3 & 3 & 3 & 5 \\ 3 & 11 & 9 & 5 & 7 \\ 13 & 13 & 11 & 9 & 9 \\ 14 & 12 & 10 & 18 & 8 \\ 15 & 14 & 16 & 17 & 18 \\ 2 & 16 & 17 & 4 & 6 \end{pmatrix}$$

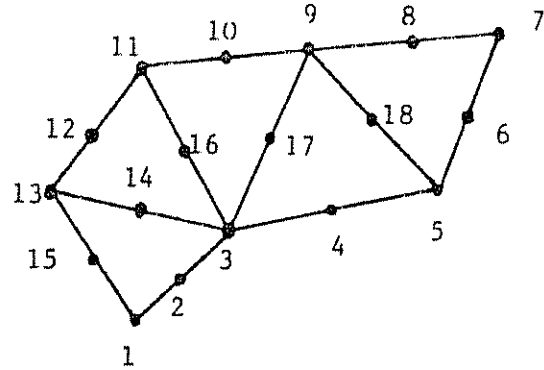


Fig.3 An example of triangulation with higher degree elements

(Note that the order in the columns of M is always the same, in some sense)

2. Stiffness Submatrix and Stiffness Matrix

Let us assume, for the sake of simplicity, that we want to solve the following variational problem (where $\Omega \subset \mathbb{R}^2$ is a polygonal domain).

$$(1) \quad \int_{\Omega} (uv + \text{grad } u \text{ grad } v) dx = \int_{\Omega} f v dx, \quad \forall v \in H^1(\Omega).$$

Note that (1) is the variational formulation of the Neumann problem introduced by LIONS.

Let \mathcal{T}_h be a triangulation of Ω , we introduce the finite element space V_h of the functions which are continuous on $\bar{\Omega}$, the restriction of which to any triangle of \mathcal{T}_h is affine. (We shall consider other elements further).

The approximate problem is then to find $u_h \in V_h$ such that

$$(2) \quad \int_{\Omega} (u_h v_h + \text{grad } u_h \text{ grad } v_h) dx = \int_{\Omega} f v_h dx \quad \forall v_h \in V_h.$$

Let $(W_i(x))_{1 \leq i \leq N}$ (N depending on h) be a basis of V_h and

$$(3) \quad u_h(x) = \sum_{i=1}^N y_i W_i(x).$$

We know that the components $(y_i)_{1 \leq i \leq N} = y \in \mathbb{R}^N$ are solutions of a linear system

$$(4) \quad Ay = b$$

where $A = (A_{ij})_{1 \leq i, j \leq N}$ is the so called "stiffness matrix", such that

$$(5) \quad A_{ij} = \int_{\Omega} (W_i W_j + \text{grad } W_i \text{ grad } W_j) dx, \quad 1 \leq i, j \leq N.$$

The vector $b = (b_i)_{1 \leq i \leq N} \in \mathbb{R}^N$ is such that

$$(6) \quad b_i = \int_{\Omega} f w_i dx, \quad 1 \leq i \leq N.$$

A simple basis of the space V_h is the following. For $i \in [1, N]$, W_i is this function of V_h , the value of which is 1 on the vertex number i , and 0 on any other vertex of the triangulation. (A function of V_h is uniquely determined by its values at the vertices of the triangulation).

However, the use of the basis function W_i and the formulae (5) is a very unefficient way to compute the stiffness matrix A , because one has to compute several times a quantity relative to one triangle!

To avoid this, we introduce the notion of stiffness submatrix. (Element stiffness matrix)

Stiffness Submatrix

We remark first that

$$(7) \int_{\Omega} (u_h v_h + \text{grad } u_h \text{ grad } v_h) dx = \sum_K \int_K (u_h v_h + \text{grad } u_h \text{ grad } v_h) dx.$$

Let us still denote by K the number of the triangle K ; the numbers of its 3 vertices are m_{1K} , m_{2K} , m_{3K} .

Let y_i (resp. z_i) denote the value of u_h (resp. v_h) at the vertices of the triangulation, and

$$\lambda_{\alpha}^K(x) \quad \alpha = 1, 2, 3$$

the barycentric coordinates associated to the nodes

$$m_{\alpha K}$$

$$\alpha = 1, 2, 3 .$$

Shape functions

The value $u_h(x)$ (resp. $v_h(x)$) for $x \in K$ is given by

$$u_h(x) = \sum_{\alpha=1}^3 y_{m_{\alpha K}} \lambda_{\alpha}^K(x)$$

$$\text{(resp. } v_h(x) = \sum_{\alpha=1}^3 z_{m_{\alpha K}} \lambda_{\alpha}^K(x) \text{)} .$$

We define then the stiffness submatrix A^K of K as the 3×3 matrix defined by

$$(8) \quad A_{\alpha\beta}^K = \int_K (\lambda_{\alpha}^K \lambda_{\beta}^K + \text{grad } \lambda_{\alpha}^K \text{ grad } \lambda_{\beta}^K) dx, \quad 1 \leq \alpha, \beta \leq 3.$$

From this definition follows the identity

$$(9) \quad \int_K (u_h v_h + \text{grad } u_h \text{ grad } v_h) dx = \sum_{\alpha, \beta=1}^3 y_{m_{\alpha K}} z_{m_{\beta K}} A_{\alpha\beta}^K .$$

Then, from the definition (5) of A , it follows that

$$(10) \quad \sum_{i,j=1}^N y_i z_j A_{ij} = \sum_{K \in T_h} \sum_{\alpha, \beta=1}^3 y_{m_{\alpha K}} z_{m_{\beta K}} A_{\alpha\beta}^K .$$

This formula being true for any $y, z \in \mathbb{R}^N$, we see that the stiffness matrix A can be computed from the submatrices A^K , for $K \in T_h$.

We can check that the following algorithm achieves the computation of A from the A^K 's (we use a language adapted from programming).

$$(11) \quad \left[\begin{array}{l} \bullet \text{ Make } A = 0 \text{ (} A \text{ is a } N \times N \text{ matrix).} \\ \bullet \text{ For each } K \in T_h \\ \quad \left[\begin{array}{l} \bullet \text{ compute } A^K \\ \bullet \text{ For } \alpha, \beta = 1, 2, 3, \text{ make} \\ \quad A_{m_{\alpha K}, m_{\beta K}} = A_{m_{\alpha K}, m_{\beta K}} + A_{\alpha\beta}^K . \end{array} \right. \end{array} \right.$$

The algorithm (11) is called "assembling" of the matrix A ; its advantage is that you only need to compute the quantities relative to one element once.

In Fig. 4, we represent the evolution of the matrix A during the assembling. The stiffness submatrices of the elements a, b, c, d , will be denoted by $\begin{pmatrix} a & a & a \\ a & a & a \\ a & a & a \end{pmatrix}, \begin{pmatrix} b & b & b \\ b & b & b \\ b & b & b \end{pmatrix}, \dots$ because we are not interested by the actual values of these coefficients.

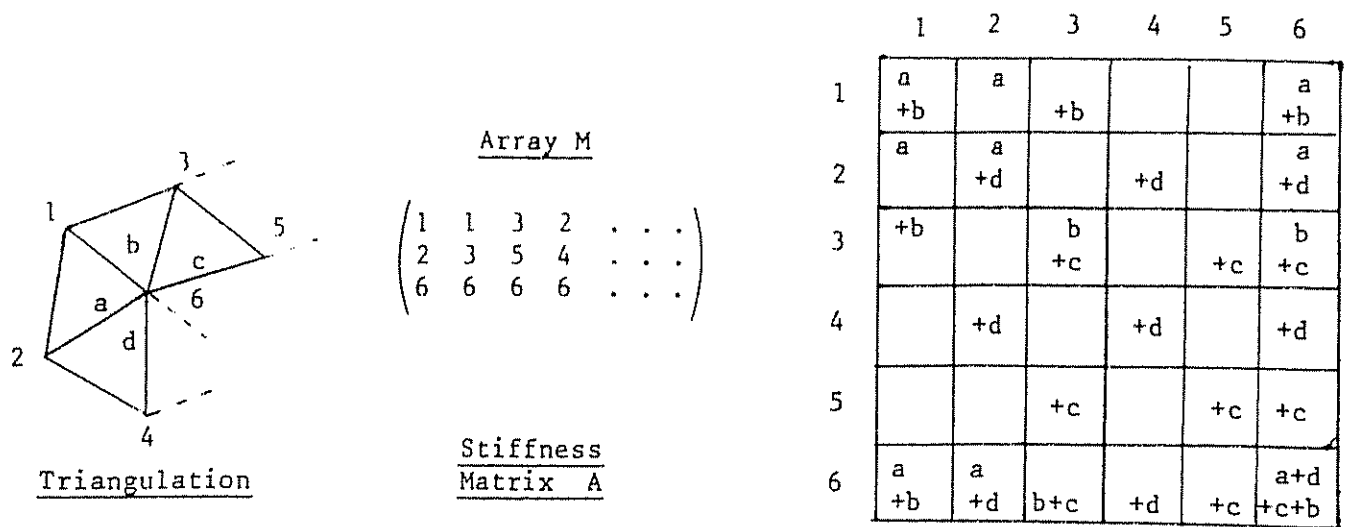


Fig. 4 An example of assembling

Exercise 3:

Write the algorithm (11) in FORTRAN. (The arrays $A(N, N)$, $AK(3, 3)$, $M(3, NT)$ are supposed to have been already declared by an order DIMENSION).

Remark 1:

Let us consider now the case of a Dirichlet boundary condition:

$$(12) \quad u_h = 0 \quad \text{on } \Gamma$$

In the variational foundation (2), v_h will satisfy the same boundary condition. Then the new stiffness matrix (the one to use in the linear system (4)) will be obtained from the one given by the assembling (11) by suppression of all the lines and columns relative to boundary nodes.

Of course, in a practical way, it is better to adapt the algorithm (11) to this new situation. For this purpose we introduce a vector noted iRA (in Fortran) such that for $1 \leq j \leq n$:

$iRA(j) = 0$ if the vertex j belongs to the boundary
 $iRA(j)$ is a new number ($\in [1, n_w]$ where n_w is the number of
internal vertices) otherwise.

The new assembling algorithm will be the following:

(13) $\left[\begin{array}{l} \bullet \text{ Make } A = 0 \\ \bullet \text{ For each } K \in T_h: \\ \quad - \text{ compute } A^K \\ \quad - \text{ For } \alpha, \beta = 1, 2, 3 \text{ compute} \\ \quad \quad n_\alpha = iRA(m_{\alpha K}) \\ \quad \quad n_\beta = iRA(m_{\beta K}) \\ \quad \text{if } n_\alpha = 0 \text{ or } n_\beta = 0, \text{ then do nothing;} \\ \quad \text{otherwise make} \\ \quad \quad A_{n_\alpha, n_\beta} = A_{n_\alpha, n_\beta} + A_{\alpha\beta}^K . \end{array} \right.$

Exercise 4 (analogous to exercise 3):

Write the algorithm (13) in FORTRAN.

Remark 2:

Note that the array iRA can be conveniently computed
from the data L defined in §1 by the following FORTRAN
instructions

```
j = 0
DØ 1 i=1, NV
iRA(i)=0
iF(L(i)) 2, 2, 1

2 j=j+1

iRA(i)=j
1CØNTINUE .
```

Remark 3:

Note that all these assembling algorithms can be easily extended to the case of other finite elements. The stiffness submatrix will only have a different size. As an example, the dimensions of A^K for a Lagrange finite element of type 2, will be 6 by 6 (instead of 3 x 3).

3. Computation of a Stiffness Submatrix

Let i, j, k denote the numbers of the vertices of a triangle K . We shall consider a reference element \hat{K} , the vertices of which are $(0,0), (1,0), (0,1)$. We consider the affine mapping F which associates i to $(0,0)$, j to $(1,0)$, and k to $(0,1)$ (Fig. 5).

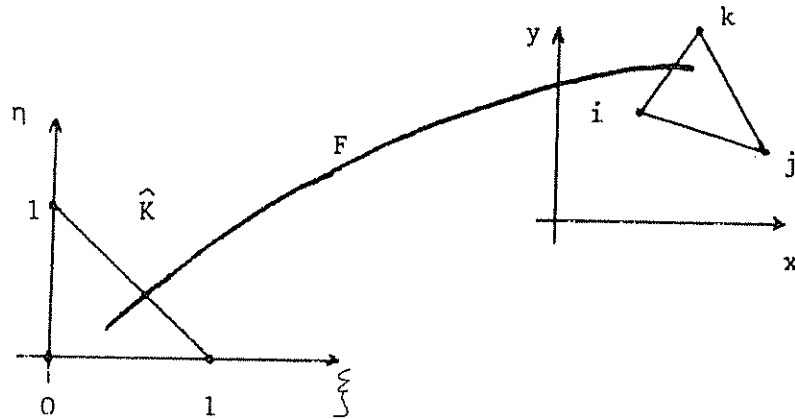


Fig. 5 The Affine Mapping F

Let $(x_i, y_i), (x_j, y_j), (x_k, y_k)$ denote the coordinates of the vertices i, j, k . We can easily check that

$$F(\xi, \eta) = B \begin{pmatrix} \xi \\ \eta \end{pmatrix} + c$$

where B is 2×2 matrix

$$(14) \quad B = \begin{pmatrix} x_j - x_i & x_k - x_i \\ y_j - y_i & y_k - y_i \end{pmatrix}$$

and

$$c = \begin{pmatrix} x_i \\ y_i \end{pmatrix}.$$

In fact ξ, η are connected to the barycentric coordinates in the following way:

$$(15) \quad \lambda_i^K = 1 - \xi - \eta; \quad \lambda_j^K = \xi; \quad \lambda_k^K = \eta.$$

The matrix B is called Jacobian matrix, and we have

$$(16) \quad |\det B| = \frac{\text{mes}(K)}{\text{mes}(\hat{K})} = 2 \text{mes}(K)$$

where $\text{mes}(K)$ denotes the area of the triangle K .

Moreover, if the nodes i, j, k are ordered anti clockwise, $\det B > 0$, and the absolute value is not necessary in (16).

Let now $v(x, y)$ be a function defined on K , and denote by $\hat{v}(\xi, \eta)$ the corresponding function defined on \hat{K} , such that

$$\hat{v}(\xi, \eta) = v(F(\xi, \eta)).$$

One can easily check that

$$\nabla \hat{v} \Big|_{\xi, \eta} = B^T \cdot \nabla v \Big|_{F(\xi, \eta)}$$

Then the following formula holds:

$$(17) \quad \int_K \nabla u \cdot \nabla v \, dx dy = \int_{\hat{K}} ((B^T)^{-1} \nabla u) \cdot ((B^T)^{-1} \nabla v) \cdot |\det B| \cdot d\xi d\eta$$

This formula can be used for the computation of the term

$$(18) \quad \int_K \nabla \lambda_{\alpha K}^K \cdot \nabla \lambda_{\beta K}^K \, dx dy$$

which occurs in the definition (8) of the stiffness submatrix A^K . In fact the formula (15) should be more rigorously written

$$\hat{\lambda}_i^K = 1 - \xi - \eta; \quad \hat{\lambda}_j^K = \xi; \quad \hat{\lambda}_k^K = \eta.$$

An easy computation gives

$$(19) \quad \nabla \hat{\lambda}_i^K = \begin{pmatrix} -1 \\ -1 \end{pmatrix}; \quad \nabla \hat{\lambda}_j^K = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad \nabla \hat{\lambda}_k^K = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Then the term to be computed in (18) is completely determined by (17) and (19). For example one has

$$\int_K \nabla \hat{\lambda}_i^K \nabla \hat{\lambda}_j^K \, dx dy = \frac{1}{2} \left((B^T)^{-1} \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right) \cdot \left((B^T)^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) |\det B|$$

The computation of the term $\int_K \hat{\lambda}_i^K \hat{\lambda}_j^K \, dx dy$ which occurs in (8) is in fact simpler from the formula:

$$(20) \quad \int_K (\lambda_i)^\alpha (\lambda_j)^\beta (\lambda_k)^\gamma \, dx dy = 2 \operatorname{mes}(K) \frac{\alpha! \beta! \gamma!}{(\alpha + \beta + \gamma + 2)!}$$

which gives immediately the result.

Remark 4:

If the variational formulation (1) changes, then one has to change in a corresponding way the definition (8) of the stiffness matrix A^K .

Exercise 5:

Consider the case of a Dirichlet problem: find $u \in H_0^1(\Omega)$ such that

$$(21) \int_{\Omega} \text{grad } u \cdot \text{grad } v \, dx = \int_{\Omega} f v \, dx \quad \forall v \in H_0^1(\Omega)$$

instead of the Neumann problem (1). We still use order 1 elements. Compute the stiffness matrix A^1 corresponding to the triangle T_1 , the vertices of which are

$$(a, b), \quad (a + h, b), \quad (a, b + h).$$

Compute then A^2 corresponding to the triangle T_2 where h is changed in $-h$.

Is the result depending on a and b ?

Exercise 6:

Use the results of the previous exercise and the assembling algorithm to find the 5th line of the stiffness matrix associated to the triangulation given in Fig.6.

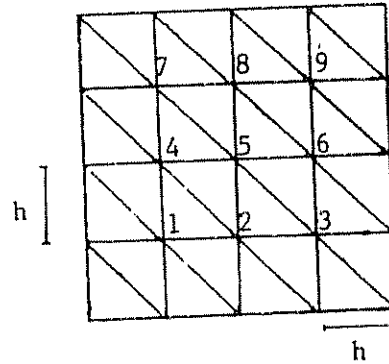


Fig. 6

(The result proves that this simple finite element method with such a uniform mesh is equivalent to a classical finite difference scheme).

4. Solution of the Final Linear System

The computation of the right hand side b occurring in the linear system (4) can be made during the assembling algorithm ((11) or (13)) in the same way that the stiffness matrix A is computed (you can check it as a very useful exercise).

For solving the linear system (4), one can think of using Cholesky's method, because the matrix A is symmetric and positive definite, and in fact it is possible to use it.

However the matrix A is sparse and considerable computing time can be saved by avoiding unnecessary operations on elements which are equal to zero.

The process is very easy when the matrix has a "band-structure" (Cf. Fig.7): For some number w called "band width" one has

$$|i - j| > w \Rightarrow A_{ij} = 0 .$$

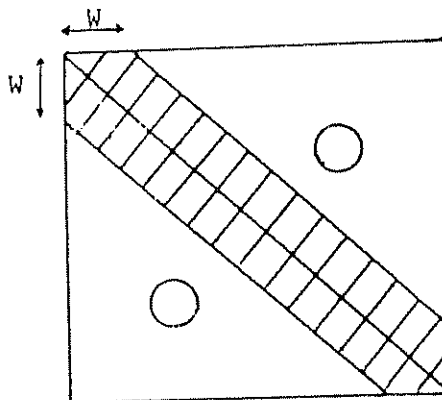


Fig. 7 Band-Structure

For such matrices it is very easy to avoid unnecessary operations, and there exists even a standard IBM scientific program adapted to this case.

The number of operations for a full $N \times N$ matrix (or considered as full) is asymptotically $\frac{N^3}{3}$ operations, while if the matrix has a band width w then this number reduces to $\frac{Nw^2}{2}$!

We shall see that the stiffness matrix A has a band structure if the nodes are suitably numbered.

In fact the stiffness matrix A has the following fundamental property:

If $A_{ij} \neq 0$, then there exists an element $K \in T_h$ such that the nodes i and j belong to K . In such a case we shall say that the nodes i and j are neighbours.

If we want to give to the matrix A a band structure, then it is sufficient that any two neighbours of the triangulation have numbers not too different. The worst case is when the first and the last node are neighbours.

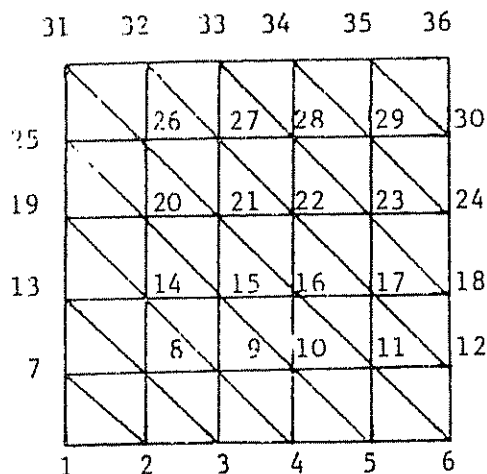


Fig. 8 The natural way of numbering the nodes for a uniform mesh on a square (here $n = 36$; $W = 6$)

For a uniform mesh on a square (Fig. 8) there is a way of numbering which produces a very small bandwidth: for a $n \times n$ square mesh, the bandwidth w can be equal to n . This gives a number of operations which is $\frac{n^4}{2}$ instead of $\frac{n^6}{3}$! For more complicated geometries the optimal numbering for the nodes looks very much the one of Fig. 9:

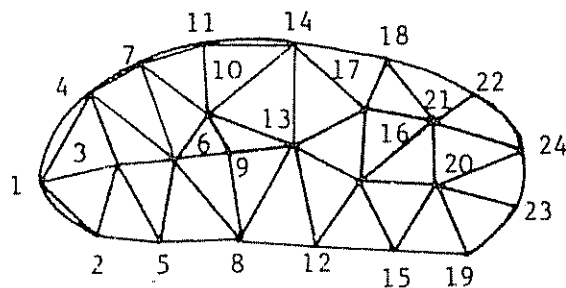


Fig. 9 - Numbering of the nodes in the case of a general geometry

Note that to evaluate W , one has just to compute the maximum of $|i-j|$ for i and j neighbours. In the case of Fig. 8, the maximum is reached for $i = 10$ and $j = 16$, then $W = 6$.

A simple (and not too bad) way of numbering the nodes is the Cuthill and McKee algorithm:

- give the number 1 to a boundary node
- give the following numbers to its neighbours
(2, 3, 4 in the case of Fig. 9)
- give the following numbers to the neighbours of the neighbours
(5, 6, 7 in the case of Fig. 9)
- etc. ...

We should now precise that the Cholesky algorithm conserves the band structure of the matrix A . That is, when one computes $A = LL^T$, where L is lower triangular, it is easy to see that L has the same bandwidth W than A .

Note that the application of Gaussian elimination would have the same effect (this because the 2 methods are equivalent in some sense).

Exercise 7:

Use the induction formulae for the computation of the factorized L in Cholesby method to prove that L has the same bandwidth than A .

Conclusion for part A

Now the reader should be able to program without difficulty a simple finite element program such as the solution of a Dirichlet problem on a circle, approximated by the simplest finite elements. (We advise the reader to use the standard IBM program for the Cholesky band solution). (*)

In the following we shall give some complements. First for more sophisticated elements, then for more complicated geometries, and, at last for large problems.

* cf. subprogram FEM1CF given as an annex to this chapter.