A propos des méthodes de décomposition de type GAUSS

Résumé :

Ce document présente quelques aspects liés à la méthode de décomposition de GAUSS.

Après une rapide présentation, nous rappelons les principaux avantages et inconvénients liés à cette méthode directe. Puis, nous détaillons l'implémentation de l'algorithme LDL^T mis en oeuvre dans le *Code Aster*.

GAUSS (1777 - 1855) est à l'origine de toutes les méthodes directes de résolution numérique de systèmes linéaires. Qu'il en soit ici remercié.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table des matières

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Généralités sur les méthodes de type GAUSS

1.1 Présentation de la méthode.

Partant de la constatation qu'il est facile de résoudre le système $A.x=b$ lorsque A est une matrice triangulaire inférieure ou supérieure, on cherche à décomposer la matrice initiale pleine par une factorisation de matrices triangulaires.

Le principe de base est de rechercher une matrice régulière P , dite matrice de permutation, telle que le produit $P.A$ soit triangulaire, puis de résoudre $P.A.x=P.b$

Remarque :

Dans la pratique, P est déterminée par produits de matrices élémentaires de permutation $P=P^{(k)}...P^{(1)}$.
Les matrices $P^{(i)}$ *dépendent de la variante choisie, mais on ne calcule jamais explicitement la matrice* P *mais seulement* $P.A$ *et* $P.b$

La matrice A étant factorisée sous la forme générale L.U (L matrice triangulaire inférieure, U matrice triangulaire supérieure), nous sommes amenés à résoudre les deux systèmes linéaires :

$$\begin{cases} L.\,\mathrm{y} = \mathrm{b} \\ U.\,\mathrm{x} = \mathrm{y} \end{cases}$$

Remarque :

Dans la méthode dite ***d'élimination de GAUSS*** *, on réalise simultanément la factorisation de A et la résolution de* $L.y=b$

L'algorithme suivant réalise l'élimination de GAUSS et la résolution de $L.y=b$

à l'étape (p+1) nous avons

$$a_{ij}^{(p+1)}=a_{ij}^{(p)}-a_{ij}^{(p)}.\left(a_{pp}^{(p)}\right)^{-1}.a_{pj}^{(p)} \quad \text{pour} \quad \begin{matrix} p+1\le i\le n \\ p+1\le j\le n+1 \end{matrix}$$

$$a_{ij}^{(p+1)}=a_{ij}^{(p)} \quad \text{pour} \quad \begin{matrix} 1\le i\le p \\ 1\le j\le n+1 \end{matrix}$$

$$a_{ij}^{(p+1)}=0. \quad \text{pour} \quad \begin{matrix} p+1\le i\le n \\ 1\le j\le p \end{matrix}$$

Remarque :

Dans cette écriture de l'algorithme d'élimination de GAUSS, le second membre b est considéré comme une colonne supplémentaire de la matrice qui est alors traitée comme une matrice n x (n+1) .

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1.2 Notion de pivot

L'implémentation précédente de l'algorithme d'élimination de GAUSS suppose implicitement que les termes diagonaux appelés **pivots** ne sont pas nuls en cours de calcul.

Cas de pivots nuls : Une matrice régulière peut avoir un pivot nul.

exemple : la matrice $\begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}$

Pour remédier à cet inconvénient, on peut utiliser une **stratégie de pivotage**

- **pivotage complet** : cette stratégie consiste à choisir comme pivot, le plus grand terme dans le bloc restant puis d'effectuer une permutation de ligne et de colonne.

  On a alors le système $P.A.Q(Q^T x)=P.b$
  où P est la matrice de permutation des lignes et Q celle des colonnes.

  La solution trouvée est alors $y=Q^T x$, et il est donc nécessaire de conserver la matrice de permutation Q pour obtenir la solution cherchée $x=Q.y$

- **pivotage partiel** : le pivot est recherché comme étant le terme de valeur maximale, parmi les termes non encore traités, dans la colonne courante (la k-ième à l'étape k) puis on effectue une permutation de ligne.

1.3 Stabilité de la méthode

Définition : Une méthode numérique de résolution de système linéaire est dite mathématiquement **stable** lorsque "quelque soit la matrice A régulière, l'algorithme réussit".

Théorème : La méthode de GAUSS avec une stratégie de pivotage est mathématiquement stable pour toute matrice régulière.

Corollaire : Si au cours d'une factorisation de GAUSS avec pivotage, un pivot nul est détecté alors la matrice est singulière et ce système n'a pas de solution unique.

Théorème : La méthode de GAUSS (sans pivotage) est stable pour des matrices réelles définies positives.

Pour plus de détails, on consultera les livres de base que sont [bib13] [bib14] [bib6].

1.4 Unicité de la décomposition

Proposition : La décomposition de GAUSS n'est pas unique,
mais si l'on spécifie la diagonale de L ou de U alors on a unicité de la décomposition.

1.5 Une variante : la factorisation de CROUT

La méthode de factorisation de CROUT [1] est le même algorithme, qui nécessite le même nombre d'opérations et effectue le même remplissage de la matrice mais les calculs sont menés de façon différente.

Nous nous plaçons dans le cas où la matrice est factorisable, ce qui est toujours le cas à une permutation près des lignes et des colonnes dès lors que la matrice est régulière : on a donc
$A=LU$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Puis on procède par identification

$$\begin{cases} \text{pour } i \le j \quad a_{ij} = u_{ij} + \sum_{k=1}^{i-1} l_{ik} . u_{kj} \\ \text{pour } i > j \quad a_{ij} = \sum_{k=1}^{j} l_{ik} . u_{kj} \end{cases}$$

(l_{ij} et u_{ij} sont les éléments de L et U)

d'où les valeurs de u_{ij} et l_{ij} en fonction de a_{ij}

$$u_{1j} = a_{1j} \qquad j = 1, \ldots, n$$

$$l_{i1} = a_{i1} / u_{i1} \qquad i = 1, \ldots, n$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} . u_{kj} \qquad i \le j$$

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} . u_{kj} \right) \qquad i > j$$

Remarque :

L'ordre des calculs n'est pas arbitraire, il faut connaître les l_{ik} situés à gauche et les u_{kj} au dessus de chaque terme à calculer.

On voit alors qu'à la k-ième étape, on reporte sur la k-ième ligne toutes les contributions antérieures laissant inchangées les lignes k+1 à n.

Cette variante de CROUT est aussi appelée élimination de GAUSS par colonnes (ou active columm), et prévilégie l'opération de produit scalaire.

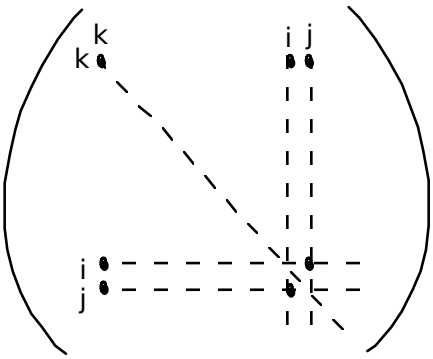
1.6 Cas des matrices symétriques

Proposition : La décomposition de GAUSS respecte la symétrie.

Il suffit de constater qu'à chaque étape les termes a_{ij} et a_{ji} reçoivent une même contribution.

En effet :

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)



par hypothèse de récurrence, on suppose la matrice symétrique à l'étape k et dès lors on a :

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - a_{ik}^{(k)} a_{kj}^{(k)} / a_{kk}^{(k)}$$
$$a_{ji}^{(k+1)} = a_{ji}^{(k)} - a_{jk}^{(k)} a_{ki}^{(k)} / a_{kk}^{(k)}$$

d'où la proposition.

Dès lors, une matrice A symétrique peut-être factorisée sous la forme $A = L D L^T$

où D est une matrice diagonale et L une matrice inférieure unitaire (c'est à dire à diagonale unitaire)

Cette décomposition, unique puisque l'on a fixé une diagonale, s'applique à toute matrice symétrique non singulière.

Si la matrice A est définie positive, alors les termes de la diagonale sont strictement positifs et l'on peut utiliser la forme dite de CHOLESKY $A = L L^T = (L D^{1/2}.D^{1/2} L^T)$.

Remarquons que la décomposition de CHOLESKY requiert n extractions de racine carrée (qui est une opération coûteuse en temps).

Dans le cas d'une factorisation LDL^T pour matrices symétriques, nous pouvons écrire l'algorithme de CROUT sous la forme suivante :

Boucle sur les colonnes ic=2,...,n
Boucle sur les contributions im=1, ¼,ic−1
$l_{ic,ic} \neg l_{ic,ic} - l_{ic,im} * l_{im,ic}$
Fin boucle
Boucle sur les lignes il=1,...,ic−1
Boucle sur les contributions im=1,... ,il−1
$l_{il,ic} \neg l_{il,ic} - l_{ic,im} * l_{im,il}$ ¿
Fin boucle
$l_{il,ic} \neg l_{il,ic} / l_{il,il}$
Fin boucle

2 Inconvénients des méthodes de type GAUSS

Les inconvénients des méthodes de type GAUSS sont essentiellement de trois types :

1) un nombre élevé d'opérations
2) un remplissage de la matrice
3) une perte de précision en cours de calcul

Les deux premiers points sont souvent qualifiés de défauts majeurs alors que le troisième est considéré comme un défaut mineur.

2.1 Le nombre d'opérations

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Pour un système plein de taille n, à la p-ième étape, nous devons effectuer pour calculer les nouveaux coefficients de la matrice et le second membre :

- (n-p) divisions
- (n-p+1)(n-p) additions et multiplications

Le nombre d'opérations est donc :

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

$\sum_{p=1}^{n-1}(n-\mathrm{p}+1)(n-p)=\sum_{q=1}^{n-1}q^2+\sum_{q=1}^{n-1}q=\left(\frac{n(n-1)(n-2)}{6}+\frac{n(n-1}{2}\right.$ additions et autant de multiplications.

Soit $\frac{1}{3}\quad n(n-1)\quad \left(n+\frac{1}{2}\right)$ opérations auxquelles il convient d'ajouter les n^2 opérations de la résolution du système triangulaire.

En résumé : Pour un grand système plein, l'algorithme de GAUSS nécessite de l'ordre de $\frac{1}{3}n^3$ opérations.

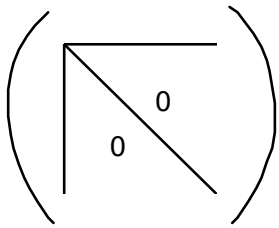
Remarque :

Dans le cas d'une matrice stockée bande, le nombre d'opérations est $n.b^2$ où b est la largeur de la bande.

2.2 Le remplissage de la matrice

Commençons par un exemple classique de matrice dite "flèche" que l'on rencontre par exemple en chimie [bib9].

Soit A la matrice telle que $a(1,i)\neq 0, a(i,1)\neq 0, a(i,i)\neq 0$ et tous ses autres termes sont nuls, la matrice a alors l'allure suivante :

A = 

Après la 1ère étape de factorisation (par l'algorithme de GAUSS), la matrice est pleine au sens où il n'y a plus de termes théoriquement nuls.

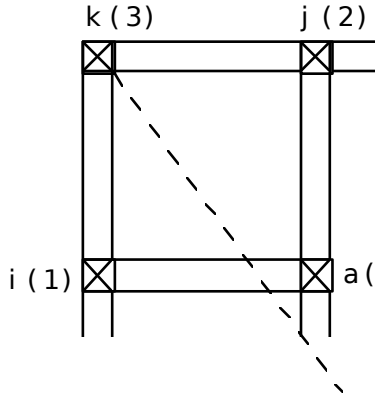
Regardons plus formellement le phénomène de remplissage du à l'algorithme ; pour cela récrivons l'algorithme :

```
Pour k variant de 1 à n - 1 faire % boucle sur les étapes
        Pour i variant de k + 1 à n faire % boucle sur les lignes
                Pour j variant de k + 1 à n faire % boucle sur les colonnes
                        a(i,j)=a(i,j)-a(i,k).a(k,j)/a(k,k)
                fin faire
        fin faire
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

fin faire

Ce que l'on peut schématiser graphiquement, à la k-ième étape, pour le calcul du terme a(i,j) par :




```
a(i,j) = a(i,j) - (1)*(2)/(3)
```

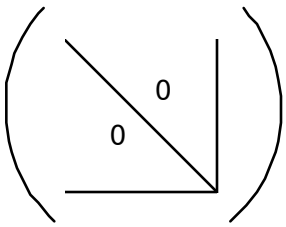
Le terme $a(i,j)$ est non nul à la fin de la k-ième étape :

- s'il était non nul au début de cette k-ième étape,
- ou si les termes $a(i,k)$ et $a(k,j)$ sont tous les deux non nuls au début de la k-ième étape, et ceci indépendamment de la valeur initiale du terme $a(i,j)$.

De plus, on voit que la méthode de GAUSS remplit le profil enveloppe de la matrice au cours des étapes de factorisation.

Dans l'exemple de la matrice "flèche" : le profil enveloppe est la matrice pleine, d'où le résultat constaté.

Cet exemple met en évidence l'importance de la numérotation des inconnues de la matrice puisque la matrice peut être récrite, après permutation des inconnues, sous la forme :



dont le profil enveloppe est la matrice elle même (il n'y a donc pas de remplissage).

Nous venons de voir l'importance de la numérotation des inconnues.

Nous ne saurions trop insister sur le fait que des algorithmes de renumérotation "optimale" doivent être utilisés pour minimiser le remplissage de la matrice.
Ces algorithmes reposent sur des heuristiques et sont spécialisés.

Parmi les algorithmes les plus couramment utilisés citons

Algorithmes	Objectifs
CUTHILL - Mc KEE	minimiser la largeur de la bande
Reverse CUTHILL - Mc KEE	minimiser le profil
Minimum Degré	minimiser les multiplications par 0

Formulation intrinsèque du remplissage

On peut donner une formulation intrinsèque du remplissage lors de l'élimination d'inconnue en termes de graphe [bib5].

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

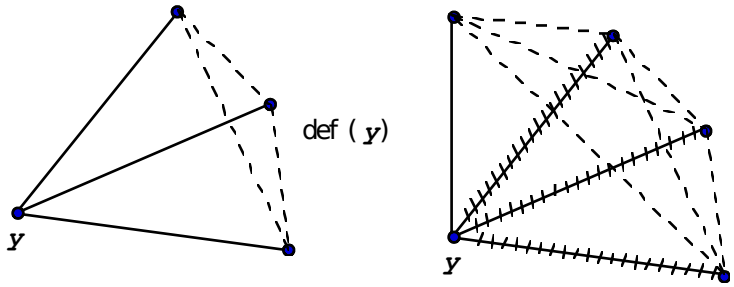
Soit une matrice A à laquelle nous associons le graphe $G(X,E)$, où X est l'ensemble des noeuds et E l'ensemble des arêtes non orientées.

Le problème d'élimination d'une inconnue de la matrice est alors équivalent à éliminer un noeud du graphe.

Définition : Soient $x,y \in X$, on dira que x et y sont **adjacents** si et seulement si $x,y \in E$

Si Y est un sous-ensemble de nœuds de $X(X \supset Y)$ nous pouvons définir les ensembles suivants :

l'ensemble des noeuds adjacents à Y	$Adj(Y)=x/x \in X-Y \text{ et } \exists\ y \in Y \text{ tel que } x,y \in E$
l'ensemble des cotés incidents à Y	$Inc(Y)=x,y/y \in Y, x \in Adj(Y)$
l'ensemble de définition de X	$\mathrm{Def}(X)=y,z/y,z \in Adj(X), y \neq z \text{ et } z \notin Adj(Y)$



- en hachuré : ce que l'on élimine,
- en pointillé : ce que l'on rajoute (remplissage)

Opération d'élimination de y

L'élimination de y consiste alors à considérer le sous-ensemble

$$Elim(y,G)=X-y,(E-Inc(y))U \overset{\text{def}}{=} (y)$$

Il faut donc bien considérer le remplissage qui est lié à $\mathrm{Def}(y)$.

Pour minimiser le remplissage on peut utiliser une heuristique consistant à éliminer le y de degré minimal, le degré de {y} étant le cardinal de l'ensemble Adj(y). C'est l'idée maîtresse de l'utilisation de l'algorithme du minimum degré avant une factorisation de type GAUSS.

Cette approche est utilisée dans la méthode multi-frontale mise en oeuvre dans *Aster* [bib15].

2.3 La perte de précision en cours de calcul

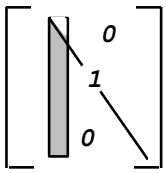
Le problème provient du fait qu'en cours d'algorithme les pivots décroissent et qu'ils sont utilisés comme dénominateur pour les étapes suivantes [bib13].

2.3.1 Etude sommaire de la perte de précision

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Notons $A^{(k)}$ la matrice à l'étape k (c'est-à-dire après l'élimination de la k-ième variable) ; avec par convention $A^{(o)}=A.$

Nous pouvons alors écrire que $A^{(1)}$ vérifie $L^{(1)}.A^{(1)}=A^{(0)}$

en prenant $L_{(1)}$ = 

par identification

Puis après (n-1) factorisations de ce type nous obtenons

$$(L^{(n-1)}\bullet\ldots\bullet L^{(1)})U=L.U=A^{(0)}$$

Du fait des erreurs E sur la décomposition, il convient d'écrire : $L.U=A^{(0)}+E$

Les coefficients de la matrice d'erreur E peuvent être évalués [bib12] en tenant compte de l'erreur commise sur l'opération flottante que nous noterons fl :

$$|e_{ij}|\leq 3\,.\,\varepsilon\,.\,m_{ij}.\max_{k,i,j}|a_{ij}^{(k)}|\;\forall_{\,i,j}\qquad\forall\,i,j$$

avec m_{ij} : nombre de termes tel que $a_{ik}^{(k)}.a_{kj}^{(k)}\neq 0$
ε : erreur relative des opérations machine

En effet, en se plaçant dans le cas où l'élimination de GAUSS "réussit" (par exemple : la matrice est définie positive ou bien on utilise une technique de pivotage).

Notons $\eta_{ik}=\beta\left(a_{ik}^{(k-1)}\Big/a_{kk}^{(k-1)}\right)\equiv\left(a_{ik}^{(k-1)}\Big/a_{kk}^{(k-1)}\right).\left(1+\varepsilon_1\right)$ pour $i>k,k=1,\ldots,n-1$

avec $|\varepsilon_1|<\varepsilon$

Le terme $a_{ij}^{(k)}$ est alors évalué par :

$$a_{ij}^{(k)}=\beta\left(a_{ij}^{(k-1)}-\eta_{ik}.a_{kj}^{(k-1)}\right)\quad\text{pour}\quad i,j>k,k=1,\ldots,n-1$$

Soit encore $a_{ij}^{(k)}=\beta\left(a_{ij}^{(k-1)}-\eta_{ik}.a_{kj}^{(k-1)}\right)$ avec $|\varepsilon_2|,|\varepsilon_3|<\varepsilon$

La "perturbation" e_{ij}^{k} subie par $a_{ij}^{(k)}$ peut alors être évaluée ; à partir de la définition de m_{ii} nous en déduisons la relation :

$$|e_{ij}^{k}|\equiv|\varepsilon_1\,.\,a_{ij}^{(k-1)}|\leq\varepsilon\,.|a_{ij}|\quad\text{pour}\quad i>k,k=1,\ldots,n-1$$

et de l'évaluation première de $a_{ij}^{(k)}$ nous déduisons :

$$|\mu_{ik}.a_{kj}^{(k-1)}|\equiv\left(a_{ij}^{(k-1)}-a_{ij}^{(k)}/\left(1+\varepsilon_3\right)\right)/\left(1+\varepsilon_2\right)$$

et finalement

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

$$|e_{kj}^{(k)}| \equiv |a_{ij}^{(k)}\left(1-\frac{1}{\left(1+e_2\right)\left(1+e_3\right)}\right)-a_{ij}^{(k-1)}\left(1-\frac{1}{\left(1+e_2\right)}\right)|$$

$$|e_{kj}^{(k)}| \leq 3.|a_{ij}|.\varepsilon$$

or la décomposition $L.U=A(0)+E$ nous indique que e_{ij} est la somme des erreurs.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2.3.2 Estimation de l'erreur sur la solution

Nous avons résolu le système

$$L.U\,x = A(0).x + E.x$$

où $E.x$ est le terme d'erreur induit par les erreurs d'arrondi/troncature dans les opérations de la factorisation.
$A(0)x$ est le second membre (en fait b).

La solution approchée trouvée $\tilde{x}$ qui approxime la vraie solution x est :

$$\tilde{x} = (A+E)^{-1}b$$

On montre alors que l'évaluation de l'erreur sur x est liée au **conditionnement de** A.

Posons le problème sous la forme : $(A+\delta A)(x+\delta x)=b$

En supposant $\delta A.\delta x$ petit on a : $\delta x = A-1.\delta A.x$

d'où en normant $\frac{\|\delta x\|}{\|x\|} \le Cond(A).\frac{\|dA\|}{\|A\|}$

où $Cond(A)$ = $\left(\|A\|.\|A\|\right)^{-1}$ est le conditionnement de la matrice A.

Remarque :

On constate que l'erreur induite sur le second membre est faible et ne perturbe la solution qu'à travers un mauvais conditionnement de la matrice A .

En effet, si l'on considère le système $A(x+\delta x)=b+\delta b,$

du fait des égalités : $\delta x = A-1.\delta b \; et \; Ax=b$

on a $\|\delta x\| \le \|A^{-1}\|.\|\delta b\| \text{et} \|b\| < \|A\|.\|x\|$

et donc $\|\frac{\delta x}{x}\| \le \left(\|A\|.\|A^{-1}\|\right)\|\frac{\delta b}{b}\|$

Remarque :

Si l'on considère les variations sur $A, x,$ *et* b *simultanément, on a l'estimation suivante [bib14] :*

$$\frac{\|\delta x\|}{\|x\|} \le \frac{Cond(A)}{1-\|A^{-1}\|.\|\delta A\|}.\left(\frac{\|\delta A\|}{\|A\|}+\frac{\|\delta b\|}{\|b\|}\right)$$

Quelques remarques sur le conditionnement

- le conditionnement n'est défini que pour une matrice régulière,
- le conditionnement dépend de la norme choisie sur $\mathbb{R}^n$,
- quelque soit la norme choisie, nous avons $1 \le Cond(A)$ et une matrice est d'autant mieux conditionnée que son conditionnement est proche de 1.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Si la norme **euclidienne** est choisie pour norme, alors

- le conditionnement d'une matrice A quelconque est

$$Cond(A) = |\frac{\mu_n}{\mu_1}|$$

où μ_1 et μ_n sont les valeurs singulières extrêmes de A (c'est-à-dire la plus petite et la plus grande des valeurs propres de $A^*.A$).

- Si la matrice A est symétrique (ou hermitienne) alors

$$Cond(A) = |\frac{\lambda_n}{\lambda_1}|$$

où λ_1 et λ_n sont les valeurs propres de module minimal et maximal de A .

2.3.3 Estimation du nombre de chiffres significatifs de la solution

Si l'on possède une précision de p chiffres (décimaux) significatifs, on a alors :

$$\frac{\|dA\|}{\|A\|} \simeq 10^{-p}$$

Si l'on désire une précision de s chiffres (décimaux) significatifs sur la solution

$$\frac{\|\delta x\|}{\|x\|} \leq 10^{-s}$$

d'où l'estimation du nombre de chiffres significatifs décimaux exacts de la solution

$$s \geq p - \log_{10}(Cond(A))$$

2.3.4 Méthode pour réduire le conditionnement

La méthode la plus simple est celle de la mise à l'échelle :

On "passe" de A à $\Phi_1.A.\Phi_2$ avec Φ_i matrice diagonale telle que Cond($\Phi_1.A.\Phi_2$) soit meilleur que $Cond(A)$.

Ceci est très théorique et il n'existe pas de méthode universelle pour déterminer $\Phi_1 \text{ et } \Phi_2$.

Notons que si la matrice A est symétrique et que l'on désire conserver cette propriété, il faut alors prendre $\Phi_1 = \Phi_2$.

2.3.5 Exemple de matrice mal conditionnée.

Cet exemple, très significatif et instructif est du à R.S. WILSON.

Soit le système $Ax = b$ avec :

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \text{ et } b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix} \text{ et dont la solution est } x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Si l'on perturbe le second membre de l'ordre de 0.5 % en prenant :

  $\tilde{b} = \left(32.1,\quad 22.9,\quad 33.1,\quad 30.9\right)$

  alors la solution est : $\tilde{x} = \left(9.2,\quad -12.6,\quad 4.5,\quad -1.1\right)$.

- Si la matrice est perturbée de l'ordre de 1 % :

$$\tilde{A} = \begin{pmatrix} 10. & 7 & 8.1 & 7.2 \\ 7.08 & 5.04 & 6 & 5 \\ 8 & 5.98 & 9.89 & 9 \\ 6.99 & 4.99 & 9 & 9.98 \end{pmatrix}$$

  alors la solution est $\tilde{x} = \left(-81,\quad 137,\quad -34,\quad 22\right)$

Remarques sur les propriétés de la matrice A :

- *Elle est symétrique, définie positive, de déterminant 1 et d'inverse "sympathique".*

$$A^{-1} = \begin{pmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{pmatrix}$$

- Son conditionnement au sens de la norme euclidienne est :

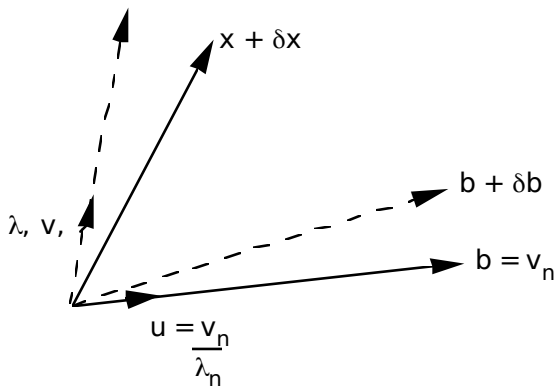
$$\mathrm{Cond}(A) = \frac{l_4}{l_1} = \frac{30.2887}{0.01015} = 2984.11$$

2.3.6 Une Interprétation géométrique du mauvais conditionnement

On peut donner une interprétation très simple du mauvais conditionnement d'un système linéaire $Ax = b$ dans le cas particulier où A est normale (i.e. $A^*.A = A.A^*$).

Soient λ_1 la plus petite valeur propre de la matrice A et λ_n sa plus grande valeur propre et soient v_1 et v_n les vecteurs propres associés.

pour $b = v_n \text{ et } \delta b$, on a $\|\frac{\delta x}{x}\| = \mathrm{cond}(A)\|\frac{\delta b}{b}\|$



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

d'où si $cond(A)$ = $|\frac{\lambda_n}{\lambda_1}|$ est grand, une petite perturbation δb de b entraîne une grande variation sur la solution v .

2.4 Critères de détermination d'un pivot nul

Définition : Un **système numériquement dégénéré** est un système pour lequel un pivot est nul ou n'a pas de chiffre significatif exact.

Notons qu'un système peut être dégénéré numériquement sans l'être mathématiquement.

Dans ces deux cas, il convient de ne pas résoudre le système d'où la nécessité de déterminer un critère d'arrêt dès que l'un des pivots n'a plus de chiffres significatifs exacts.

Soient A la matrice à factoriser et F la matrice de diagonale libre issue de la factorisation.

Critère 1 : Le critère le plus simple est de considérer que le pivot est nul dès qu'il est inférieur, en valeur absolue à un seuil donné.

$$|f_{ii}|<\varepsilon_1$$

ε_1 est un "nombre petit" en dessous duquel on considère que les valeurs sont arbitrairement nulles.

Critère 2 : Ce critère s'applique au nombre de chiffres significatifs encore disponibles. En constatant que l'on ne peut avoir plus de p chiffres significatifs sur une machine donnée, on vérifiera que la décroissance du pivot ne s'effectue pas dans un rapport supérieur à 10^{-p}.

$$|\frac{f_{ii}}{a_{ii}}|\leq\varepsilon_2=10^{-p}$$

Notons que le rapport $|\frac{f_{ii}}{a_{ii}}|$ est toujours inférieur ou égal à 1 du fait de la décroissance des pivots.

La cause de base d'un mauvais conditionnement numérique est l'erreur d'arrondi provoquée par l'introduction de grands nombres sans signification physique.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3 Méthode LDL^T par blocs mise en oeuvre dans *Aster*

Ce paragraphe détaille la mise en oeuvre dans *Aster* de la résolution du système linéaire `A.x = b` par la méthode de factorisation LDL^T de la matrice symétrique `A`.

La matrice `A` est stockée en profil (ou ligne de ciel) par bloc.

Principe de base : Une colonne de la matrice est contenue toute entière dans un seul bloc : nous ne segmentons pas les colonnes.

Les tableaux permettant la description de la matrice stockée profil par bloc sont :

- `HCOL` hauteur de colonne de la matrice
  `HCOL(i)` hauteur de la i-eme colonne
- `ADIA` adresse du terme diagonal dans son bloc
  `ADIA(i)` renvoie l'adresse du i-eme terme diagonal dans son bloc
- `ABLO` pointeur de bloc
  `ABLO(i+1)` renvoie le numéro de la dernière équation dans la numérotation globale contenu dans le i-eme bloc
  Par convention `ABLO(1) = 0` et le nombre d'équations dans le i-eme bloc est donné par la relation `ABLO(i+1) - ABLO(i) + 1`
  Le nombre total d'équation se déduit comme étant `ABLO(nombre_de_bloc + 1)`

Il est également nécessaire de mémoriser le nombre total de blocs utilisés pour contenir les coefficients de la matrice.

Remarque :

Formellement le tableau `HCOL` est inutile car il se déduit des tableaux `ADIA` et `ABLO`, mais il permet d'effectuer les calculs plus rapidement.

3.1 Mise en oeuvre de la factorisation

Les principales particularités de la mise en oeuvre de la factorisation de GAUSS par la variante de CROUT sous forme LDL^T d'une matrice symétrique stockée profil par bloc sont :

- la factorisation est effectuée en place, c'est à dire en écrasant la matrice initiale,
- la factorisation peut-être partielle,
- les critères de détections de pivot nuls peuvent être adaptés à la factorisation de matrices quasi-singulières,
- en cas de détection de pivot nul, ce pivot est remplacé par une très grande valeur (10^{40}) ce qui revient à introduire une condition de blocage par pénalisation.

Remarque :

On crée deux tableaux de travail :
- *un tableau qui contiendra la diagonale de la matrice factorisée (minimisation du nombre d'accès au bloc),*
- *un tableau pour la colonne courante (minimisation du nombre de calculs effectués).*

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
DEBUT Algorithme ;
    création d'un tableau intermédiaire pour la colonne courante
    création d'un tableau intermédiaire pour la colonne courante

    POUR  ibloc VARIANT_DE 1 A nombre_de_bloc FAIRE
```

- Détermination des colonnes de départ et de fin pour le bloc courant.
- Recherche de la plus petite équation en relation avec une équation contenue dans le bloc courant. Cette recherche se fait en exploitant le tableau HCOL
- Recherche du bloc d'appartenance de l'équation trouvé précédemment. Cette recherche se fait en exploitant le tableau ABLO.
- Requête en mode écriture du i-eme bloc.

```
    POUR jbloc VARIANT_DE plus_petit_concerne A ibloc-1 FAIRE
           Requête en mode lecture du j-eme bloc
           POUR  iequa  CONTENUE DANS LE i-eme bloc FAIRE
               calcul de l'adresse de départ de la colonne dans le bloc
               calcul de la hauteur de la colonne
               POUR jequa CONTENUE DANS LE j-eme bloc FAIRE
                   calcul de l'adresse de départ de la colonne dans le bloc
                   calcul de la longueur de la colonne
                   A(ibloc,jequa) = A(ibloc,jequa)- < A(ibloc,*), A(jbloc,*)
                   >
               FIN_POUR
           FIN_POUR
           libération  du j-eme bloc qui n'a pas été modifié
    FIN_POUR

    POUR  iequa  CONTENUE DANS LE i-eme bloc FAIRE
        calcul de l'adresse de départ de la colonne dans le bloc
        calcul de la longueur de la colonne
        POUR  jequa  CONTENUE DANS LE i-eme bloc et < iequa FAIRE
            calcul de l'adresse de départ de la colonne dans le bloc
            calcul de la hauteur de la colonne
            A(ibloc,lm) = A(ibloc,lm) - < A(ibloc,*) , A(jbloc,*) >
        FIN_FAIRE

    %   utilisation de la colonne iequa (calcul du pivot)
        calcul de l'adresse de départ de la colonne dans le bloc
        calcul de la hauteur de la colonne
        sauvegarde de la colonne: trav(i) ¬ A(ibloc,i)
        normalisation de la colonne en utilisant le tableau diagonal :
            A(ibloc,*) ← A(ibloc,*) / diag(*)
        calcul du terme diagonal et actualisation du tableau de travail:
            tabr8(iadia) = tabr8( iadia ) - < A(ibloc,*) , trav(*) >

        test du pivot par rapport à ε
        test du pivot par rapport au nombre de chiffres significatifs

    FIN_POUR

    libération du bloc courant que l'on vient de modifier
    FIN_POUR

    libération des tableaux de travail
FIN Algorithme ;
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Détermination de la colonne de départ et de fin pour le bloc courant.

Cet phase est due à la notion de factorisation partielle.

```
SI  (dernière colonne du bloc < début de factorisation) ALORS
    requête en mode lecture du i-eme bloc
    remplir le tableau de travail contenant la diagonale.
    libération du i-eme bloc
    ALLER_AU bloc suivant
SINON_SI (première colonne du bloc > fin de factorisation) ALORS
    SORTIR
SINON
    SI (première colonne du bloc < début de factorisation) ALORS
       % compléter le tableau "diagonal"
       requête en mode lecture du i-eme bloc
       remplir le tableau de travail contenant la diagonale.
    FIN_SI
    SI (dernière colonne du bloc > fin de factorisation) ALORS
       modification de la dernière équation à prendre en compte
    FIN_SI
FIN_SI
```

Remarque sur l'encombrement :

Il est obligatoire que l'on puisse avoir au moins simultanément en mémoire :
- *deux blocs de la matrice,*
- *deux vecteurs de travail de taille : le nombre d'équations du système à résoudre.*

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3.2 Mise en oeuvre de la résolution

La mise en oeuvre de la résolution simultanée de n seconds membres du système A.x = b, où la matrice A est symétrique et a été factorisée sous forme L D L^T (la résolution est en place)

Remarque :

On crée un tableau de travail qui contiendra la diagonale de la matrice factorisée, en vue de minimiser le nombre accès au bloc de la matrice et donc de limiter les lectures pour les grosses matrices ne pouvant résider totalement en mémoire.

```
DEBUT Algorithme ;
    Création d'un tableau pour stocker la diagonale pour éviter des lectures lors de l'étape de
    résolution diagonale.
    POUR ibloc VARIANT_DE 1 AU nombre_de_bloc   % résolution descendante et
                                                % remplissage du tableau diagonal
        requête en mode lecture du i-eme bloc
        POUR iequa contenue DANS LE BLOC
            Calcul de la hauteur de la colonne et
            calcul de l'adresse de départ de la colonne dans son bloc
            POUR chaque second membre FAIRE
                xsol(isol) = xsol(isol) - < x(isol) , U) >
    FIN_POUR
    sauvegarde du terme diagonal dans le tableau de travail
    FIN_POUR
    libération du i-eme bloc
FIN_POUR

POUR chaque second membre FAIRE      % résolution diagonale
    POUR toute les équations FAIRE
        xsol(iequa,isol) = xsol(iequa,isol) / diag(iequa-1)
    FIN_POUR

    POUR ibloc VARIANT_DE nombre_de_bloc à 1 PAR_PAS_DE -1% résolution remontante
        requête en mode lecture du i-eme bloc
        POUR iequa contenue DANS LE BLOC
            Calcul de la hauteur de la colonne et
            calcul de l'adresse de départ de la colonne dans son bloc
            POUR chaque second membre FAIRE
                xsol(ixx+i,isol)= xsol(ixx+i,isol)- xsol(isol)*L(ide+i)
            FIN_POUR
        FIN_POUR
        libération du i-eme bloc
    FIN_POUR

Libération de la zone de travail (c'est à dire du tableau diagonal)
FIN Algorithme ;
```

Remarque sur l'encombrement :

Il est nécessaire que l'on puisse avoir simultanément en mémoire :
- *un bloc de la matrice,*
- *un vecteur de travail de taille : le nombre d'équations du système à résoudre,*
- *les n seconds membres.*

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3.3 Mise à l'échelle

Il est possible de faire une mise à l'échelle de la matrice à factoriser ; cette mise à l'échelle simple se fait de façon à obtenir une matrice dont les termes diagonaux valent `1`.

La matrice diagonale est telle que :

$$\Phi_i = \begin{cases} \dfrac{1}{\sqrt{|a_{ii}|}} & si\ a_{ii} \neq 0 \\ 1 & si\ a_{ii} = 0 \end{cases}$$

Il est à noter que lors de la résolution, on n'obtient la solution du système qu'après déconditionnement.

En effet : le système initial est $Ax = b$

après multiplication à gauche par Φ , on a :

$$\Phi Ax = \Phi . b$$

Or le système résolu est :

$$\Phi A \Phi x = \Phi b$$

d'où la solution Φx obtenue qu'il faut "déconditionner".

3.4 Tests sur le pivot

Deux critères de détections de pivot nuls sont implémentés :

- le test en valeur absolue $|a_{ii}| < \varepsilon$, avec ε donné,
- le test en valeur relative sur le nombre de chiffres significatifs exacts.

Notons que ces tests peuvent être réduits à leur plus simple expression en fournissant un $\varepsilon = 0$. et en donnant, par convention, un nombre de chiffres significatifs exacts nul.

Cette option est rendue nécessaire par le fait que des algorithmes tels que les algorithmes de recherche de valeurs propres [R5.01.01] [R5.01.02] cherchent à factoriser des matrices quasi-singulière.

3.5 Factorisation de matrices complexes

L'algorithme implémenté dans *Aster* permet également de traiter les matrices **symétriques** à coefficients complexes.

L'algorithme implémenté ne traite pas les matrices hermitiennes, bien que ce soit théoriquement possible.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Annexe 1Méthodes de stockage classiques

A1.1 Matrice pleine

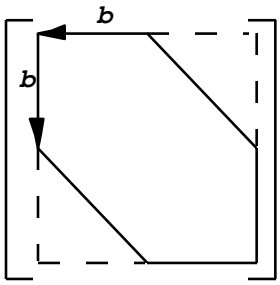
Une matrice pleine non symétrique de taille n possède n^2 coefficients.

Si la matrice est symétrique, on peut ne stocker que sa triangulaire inférieure ou supérieure soit $\frac{n(n+1)}{2}$ valeurs.

Aucun tableau de description de la matrice n'est nécessaire.

A1.2 Matrice bande

b + 1 = largeur de bande



Dans ce cas on stocke la bande (appelée parfois matrice redressée) dans un tableau rectangulaire n (2b + 1) ; on inclut alors les b (b + 1) valeurs nulles correspondant aux compléments des pointes.

Dans le cas d'une matrice symétrique, on peut ne stocker que n (b + 1) valeurs dont $\frac{b(b+1)}{2}$ valeurs nulles (inutiles).

Cette méthode nécessite uniquement de connaître la largeur de la bande.

A1.3 Matrice profil ou matrice à ligne de ciel

Cette technique consiste à stocker les termes de la matrice par colonnes et lignes de longueurs variables. Les termes extérieurs à la "ligne de ciel", qui est l'enveloppe des sommets des colonnes étant supposés n'avoir aucune contribution dans les calculs, ne sont pas stockés.

Le profil de la i-ème ligne (resp. colonne) est déterminé par :

$$\min \{j \text{ tel que } 1 \leq j \leq n \quad a_{ij} \neq 0\}$$
$$(\text{resp } \min \{j \text{ tel que } 1 \leq j \leq n \quad a_{ji} \neq 0\})$$

Si le profil est symétrique, on parle de matrice à profil symétrique.

Cette méthode de stockage nécessite des tableaux de stockage que nous allons détailler dans le cas d'une matrice à profil symétrique.

Classiquement, dans cette option de stockage, la matrice est rangée sous la forme d'un tableau mono dimensionné nécessitant un tableau de pointeur d'entrée de colonne ADIA pour explorer la matrice : l'entrée se fait par les termes diagonaux, le nombre de termes de la colonne est obtenu par différences de deux termes successifs : ADIA(i+1) - ADIA(i).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Si la matrice est non symétrique, mais à profil symétrique, il est nécessaire de stocker la i-ème ligne et la i-ème colonne. Classiquement, on les mets "bouts à bouts" et le nombre de termes de la colonne ou de la ligne est (`ADIA(i+1) - ADIA(i))/2`.

A1.4 Stockage par bloc

Les méthodes de stockage vues précédemment supposent implicitement que la matrice puisse résider en mémoire centrale, ce qui n'est pas toujours le cas.

D'où la notion de matrice stockée par bloc (ou segmentée sur disque).

Tous les stockages précédents peuvent être segmentés, mais nous n'exposerons que le cas de la matrice symétrique stockée profil.

Matrice profil stockée par bloc

Nous ne considérons ici que le cas des matrices symétriques, ce qui n'ôte rien à la généralité du propos.

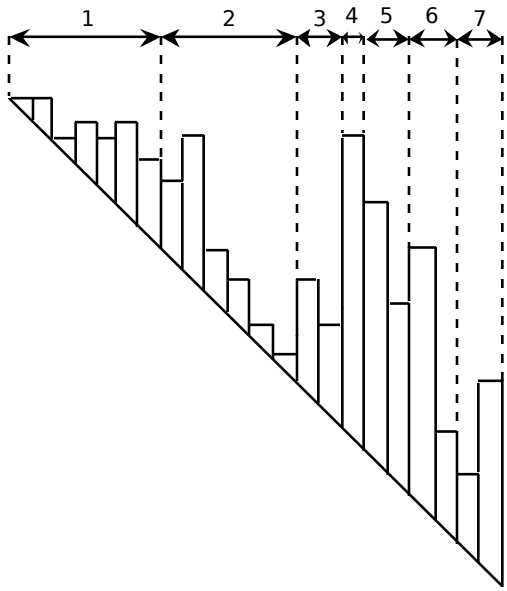


Figure A1.4-a : Taille maximale d'un bloc : 20 éléments

Dans cet exemple, nous supposons les blocs de même taille, pour utiliser des blocs de taille variable, il faut introduire un tableau supplémentaire contenant la taille de chaque bloc.

Nous considérons également qu'une colonne ne peut appartenir qu'à un seul bloc : "nous ne coupons pas les colonnes".

Pour gérer la matrice, il est toujours nécessaire de connaître l'adresse des termes diagonaux ; mais maintenant, cette adresse est relative au bloc d'appartenance de la colonne.

A ce tableau, il est nécessaire de joindre un tableau donnant les équations contenues dans un bloc.

Ce tableau dimensionné au nombre de bloc plus `1` contient la dernière équation du bloc.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Annexe 2Variations sur l'algorithme de GAUSS

Comme nous l'avons vu avec la variante de CROUT, il existe plusieurs implémentations de l'algorithme de GAUSS : elle consiste à effectuer les calculs des coefficients dans un ordre différent.

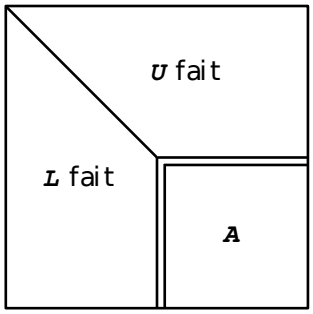
Schématiquement, on considère qu'il y a trois boucles imbriquées :

- boucle `i` sur les lignes,
- boucle `j` sur les colonnes,
- boucle `k` sur les étapes.

L'algorithme standard se caractérise par la séquence `kij`, mais il existe 5 autres permutations des indices qui donnent lieu à autant de variantes (ou d'algorithmes).

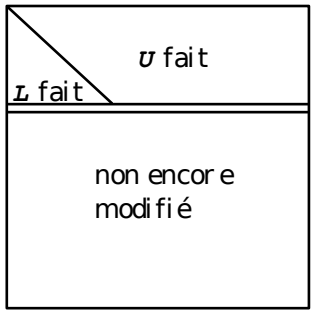
- l'algorithme de CROUT se caractérise par la séquence `jki`,
- l'algorithme correspondant à la séquence `ikj`, qui travaille par ligne, est connu sous le nom d'algorithme de "Doolitle".

Donnons ici une représentation graphique tirée de [bib10].



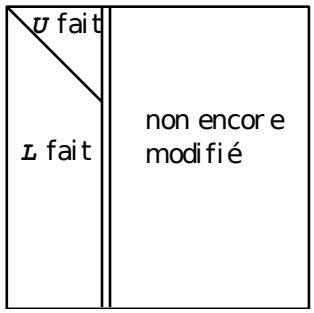
algorithmes `kij - kji`

on calcule la k-ième colonne et la `k + 1` ligne et l'on réactualise la sous-matrice `A`



algorithmes `ikj - ijk`

on calcule la i-ième ligne de `L` et `U`



algorithmes `jki - jik`

on calcule la j-ième ligne de `L` et `U`

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 Bibliographie

[1] P.D. CROUT A short method for evaluating determinants and solving systems of linear equations with real or complex coefficients.-AIEE Trans Vol 60, 1941, pp 1235 - 1240

[2] E. CUTHILL & J. Mc KEE "Reducing the band with of sparse symmetric matrices" Proc 24th Nat Conf Assoc Comput Mach ACM Publ (1969)

[3] E. CUTHILL "Several strategies for reducing the bandwith of the matrices" dans Sparse Matrices and their applications - D.J. ROSE & R.A. WILLOUGHBY Editeurs, Plenum Press, New York (1972) pp 157 - 166

[4] G. Von FUCHS, J.R. ROY, E. SCHREM Hypermatrix solution of large sets of symmetric positive definite linear equations - Computer methods in Applied Mechanics and Engineering (1972)

[5] A. GEORGE, D.R. Mc INTYRE "On the application of the minimum degree algorithm to finite element systems" SIAM J. Num. An. Vol 15, (1978) pp90 - 112

[6] G.H. GOLUB et C.F. Van LOAN Matrix computations. Johns Hopkins University Press - Baltimore (1983)

[7] B. M. IRONS Roundoff criteria in direct stiffness solutions - AIAA Journal 6 n° 7 pp 1308 -1312 (1968)

[8] B. M. IRONS A frontal solution program for finite elements analysis - Int Journal Num. Meth. Eng.,2, 1970

[9] O'LEARY & STEWART, Computing the eigenvalues and eigenvectors of symmetric arrowhead matrices, J. of comp physics 90, 497-505 (1990)

[10] J.M. ORTEGA "Introduction to parallel and vector solution of linear systems Plenum Press (1988)

[11] G. RADICATI di BROZOLO, M. VITALETTI Sparse matrix-vector product and storage representations on the IBM 3090 with vector facility - IBM-ECSEC Report G513 - 4098 (Rome) July 1986

[12] J.K. REID A note on the stability of gaussian elimination. J. Int Maths Applies, 1971, 8 pp 374 - 375

[13] J.H. WILKINSON Rounding Errors in Algebraic. Processes Her majesty's stationery office (1963)

[14] J.H. WILKINSON The algebraic eigenvalue problem Clarendon Press Oxford (1965)

[15] C. ROSE "Une méthode multifrontale pour la résolution directe de systèmes linéaires" Note EDF - DER HI-76/93/008 (1993)

[16] D. SELIGMANN "Algorithmes de résolution pour le problème généralisé aux valeurs propres" [R5.01.01] - Note EDF - DER HI-75/7815 (1992)

[17] D. SELIGMANN, R. MICHEL "Algorithmes de résolution pour le problème quadratique aux valeurs propres" [R5.01.02] - Note EDF - DER HI-75/7816 (1992)

5 Description des versions du document

Version Aster	Auteur(s) Organisme(s)	Description des modifications

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2,3	D.SELIGMANN EDF-R&D/AMA	Texte initial

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)