Le chapitre V concerne la résolution des systèmes linéaires.

Volontairement, nous nous limitons à une présentation succincte.

Nous abordons

1. les méthodes directes,
2. les méthodes itératives,
3. la prise en compte des conditions aux limites.

V . RÉSOLUTION DES SYSTÈMES LINÉAIRES

La résolution des systèmes linéaires est l'étape qui sollicite le plus les ressources de l'ordinateur:

– d'un point de vue mémoire,

– d'un point de vue temps calcul.

Le **caractère creux** des matrices provenant de la résolution d'E.D.P. par la méthode des éléments finis est dû au fait que le support des fonctions de base utilisées est contenu dans un petit nombre d'éléments de la triangulation.

Il faudra donc tirer un profit maximum de la structure du système pour minimiser la place mémoire et le temps calcul.

V.1. Les étapes

Nous ne résolvons pas de systèmes quelconques, mais des systèmes obtenus par discrétisation d'E.D.P. Ceci va bien sûr influencer la façon de résoudre le système linéaire.

Supposons avoir à disposition les tableaux (matrices et seconds membres) élémentaires. Alors les étapes nécessaires pour résoudre le système sont les suivantes.

1. Calculer les *pointeurs* ou tableaux utilitaires nécessaires pour le stockage.

2. Assembler la matrice et le second membre du système.

3. Prendre en compte les conditions aux limites.

   Il est parfois nécessaire de modifier la matrice et le second membre du système pour prendre en compte les conditions aux limites.

4. Résoudre le système obtenu, *i.e.* trouver $x \in \mathcal{R}^{n \times m}$ tel que

$$Ax = b,$$

   où $A \in \mathcal{M}_n(\mathcal{R})$, $b \in \mathcal{R}^{n \times m}$ sont donnés.

   On distingue les méthodes directes et les méthodes itératives.

Nous nous intéressons aux deux classes de méthodes (directes et itératives) pour la résolution proprement dites, (i.e. factorisation, descente, remontée). Le stockage va dépendre de la méthode.

V.2. Les méthodes directes

Dans Modulef, nous pouvons utiliser les méthodes suivantes:

$$\text{Cholesky :} \quad A = L^tL,$$

$$\text{Crout :} \quad A = LD^tL,$$

$$\text{Gauss :} \quad A = LU, \quad \text{avec pivot.}$$

Remarque. Pour chacune des méthodes ci-dessus, le **profil** est conservé.

Considérons l'algorithme de Cholesky. Nous avons les expressions

$$L_{ij} = (A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk}) / L_{jj}$$

et

$$L_{ii} = (A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2)^{1/2}.$$

Si $A_{ij} \neq 0$, alors $L_{ij} \neq 0$ à priori.

D'autre part, si $A_{ij} = 0$, alors les deux situations ci-dessous peuvent avoir lieu:

1. soit il n'existe pas de $j_0 < j \leq i$ avec $A_{ij_0} \neq 0$, et $L_{ij} = 0$,

2. soit il existe $j_0 < j \leq i$ avec $A_{ij_0} \neq 0$, et $L_{ij} \neq 0$ à priori.

Conséquence. Pour les méthodes directes, nous allons utiliser un stockage profil. Ce stockage est mis en œuvre dans la **S.D. MUA**.

La S.D. MUA est composée de 6 tableaux dont l'organisation est la suivante.

1. MUA0: informations générales,
2. MUA1: descripteurs des tableaux supplémentaires éventuels,
3. MUA2: description générale de la matrice,
4. MUA3: pointeurs sur le dernier degré de liberté de chaque bloc ou page de la matrice,
5. MUA4: pointeurs sur le coefficient diagonal de chaque ligne,
6. MUA5: coefficients de la matrice.

La S.D. MUA : cas symétrique

Soit $A \in \mathcal{M}_n(\mathcal{R})$ une matrice symétrique, que l'on souhaite stocker dans une S.D. MUA.

Pour $i = 1, \ldots, n$, on pose

$$ml(i) = \min\{j, 1 \leq j \leq i \leq n, A_{ij} \neq 0\},$$
$$mc(i) = \min\{j, 1 \leq j \leq i \leq n, A_{ji} \neq 0\}.$$

Puisque A est symétrique, on a

$$ml(i) = mc(i) \equiv m(i).$$

Seuls les coefficients A_{ij} avec $m(i) \leq j \leq i$ sont stockés dans un tableau monodimensionnel.

Par exemple, nous considérons la matrice symétrique d'éléments

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

Alors pour cet exemple, nous avons

$$MUA4 = (0\ 1\ 3\ 6\ 9\ 12\ 18)$$

et

$$MUA5 = (1\ 2\ 3\ 4\ 5\ 6\ \ldots\ 18).$$

La valeur $m(i)$ est donnée par la formule

$$m(i) = (i+1) - MUA4(i+1) + MUA4(i).$$

L'adresse de A_{ij} dans $MUA5$ est donnée par l'expression

$$\text{adresse } = MUA4(i+1) - i + j.$$

La S.D. MUA : cas non symétrique

Soit $ml(i)$ et $mc(i)$ les quantités définies précédemment. Nous posons

$$m(i) = \min\{ml(i), mc(i)\}.$$

Nous imposons donc un profil symétrique.

Dans le cas d'une matrice A non symétrique, l'adressage est un peu plus compliqué que dans le cas symétrique.

Par exemple, nous considérons la matrice d'éléments

$$\begin{pmatrix} 1 & 3 & 7 & & & 25 \\ 2 & 4 & 8 & 12 & & 26 \\ 5 & 6 & 9 & 13 & 17 & 27 \\ & 10 & 11 & 14 & 18 & 28 \\ & & 15 & 16 & 19 & 29 \\ 20 & 21 & 22 & 23 & 24 & 30 \end{pmatrix}.$$

Alors les tableaux $MUA4$ et $MUA5$ se présentent de la façon suivante.

$$MUA4 = (0\ 1\ 4\ 9\ 14\ 19\ 30)$$

et

$$MUA5 = (1\ 2\ 3\ 4\ 5\ \ldots\ 30).$$

La valeur de $m(i)$ est donnée par

$$m(i) = i - \{MUA4(i+1) - MUA4(i) - 1\}/2.$$

L'adresse du coefficient A_{ij} dans $MUA5$ est donnée par: si $i \leq j$

$$\text{adresse } = MUA4(j+1) - j + i$$

et si $i > j$

$$\text{adresse } = MUA4(i+1) -$$

$$\{MUA4(i+1) - MUA4(i) - 1\}/2 - i + j.$$

Cette technique de stockage est bien adaptée à la factorisation des matrices creuses lorsque nous utilisons des méthodes directes.

Liste des modules nécessaires

PREPAC	préparation
ASMAPC	assemblage de A en MC
ASMBMC	assemblage de b en MC
CLIMPC	prise en compte des C.L.
CHOLPC	
ou	
CROUPC	factorisation en MC
ou	
GAUSPC	
DRCHPC	
ou	
DRCRPC	descente-remontée en MC
ou	
DRGAPC	

Remarque

Nous mentionnons ici *en mémoire centrale* (MC), car dans Modulef nous avons la possibilité de résoudre de très gros systèmes avec mémoire secondaire.

Nous devons utiliser dans ce cas la **S.D. HYMA** (pour hypermatrice).

Cette possibilité est désuète depuis l'utilisation d'ordinateurs à mémoire virtuelle. Dans ce cas la gestion de la mémoire (pagination) est gérée par le système d'exploitation.

Quelques mots sur les modules

PREPAC: calculer les tableaux $MUA3$ et $MUA4$ et le nombre de mots nécessaires au stockage des S.D. MUA et B; permettre le choix mémoire centrale ou mémoire secondaire; éviter un début d'exécution perdu faute de place.

ASMAPC: assembler en mémoire centrale dans la S.D.S. MUA la matrice

$$[A] = \sum_{T \in \mathcal{T}_h} {}^t[\mathcal{B}^T][A^T][\mathcal{B}^T]$$

où $[\mathcal{B}^T]$ est la matrice booléenne de passage de la numérotation globale à la numérotation locale.

ASMBMC: assembler en mémoire centrale dans la S.D.S. B le vecteur $\{b\}$

$$\{b\} = \sum_{T \in \mathcal{T}_h} {}^t[\mathcal{B}^T]\{b^T\}.$$

CLIMPC: prendre en compte les conditions aux limites de type *valeur au nœud* ou *en relation linéaire*. Nous verrons dans la section suivante comment prendre en compte effectivement les conditions aux limites *forcées*.

Optimisation du remplissage

À titre d'exemple, nous considérons le tp2.

Nous effectuons les calculs une fois avec renumérotation et une fois sans renumérotation.

Le profil de la matrice du système a été tracé à l'aide de TRPRXX.

Il est bien évident au vu de cet exemple, qu'il est important de renuméroter pour optimiser le profil.

Considérons le schéma à 5 points dans la carré:

$$
\begin{array}{ccccc}
 & & N(i) & & \\
 & & | & & \\
O(i) & - & i & - & E(i) \\
 & & | & & \\
 & & S(i) & &
\end{array}
$$

Avec la numérotation *naturelle*, nous avons:

$$
\begin{array}{c}
N(i) = i + m, \quad O(i) = i - 1, \\
E(i) = i + 1, \quad S(i) = i - m.
\end{array}
$$

Le nombre de coefficients à stocker pour chaque ligne vaut $m + 1$ pour tout sommet interne.

Pour diminuer la largeur de bande, il suffit d'imposer aux voisins du sommet i les numéros les plus proches de i.

Ce principe est utilisé dans la méthode de renumérotation de Cuthill-MacKee.

Partant d'un sommet, de numéro 1, on recherche les voisins dans S_2. Dans S_2, on prend tour à tour ceux qui ont le moins de voisins.

On recommence avec les sommets restants.

Remarque

• On peut modifier la méthode en inversant la numérotation. Le résultat est en général meilleur.

• On peut rechercher à minimiser le nombre de termes à stocker: numérotation *RED-AND-BLACK*.

V.3. Conditions aux limites

La prise en compte des conditions aux limites *forcées*, est faite par le module CLIMPC.

Les conditions aux limites peuvent être de deux types:

$$u_{\text{nœud},dl} = \beta, \tag{1}$$

$$\sum_{i=1}^{m} \alpha_i u_{\text{nœud}i,dli} = \beta. \tag{2}$$

Étudions le cas 1.

L'assemblage est fait indépendamment des conditions aux limites. Nous sommes ainsi amenés à résoudre

$$[A]\{u\} = \{b\}$$

sous la contrainte

$$u_{ni,dli} = \overline{u}_{ni,dli}.$$

Supposons que nous voulions fixer $u_i = \overline{u}_i$. La ligne i du système s'écrit

$$\sum_j A_{ij} u_j = b_i.$$

Nous rajoutons alors

$$\sum_{j \neq i} A_{ij} u_j + VTG * A_{ii} * u_i = VTG * A_{ii} * \overline{u}_i.$$

Nota Bene. Dans certains cas, cette méthode peut se révéler désastreuse, car elle détériore le conditionnement du système.

Une autre méthode consiste à annuler les termes nondiagonaux sur la i° ligne et à rajouter 1 sur le terme diagonal.

Étudions le cas 2.

Lorsque nous avons des conditions aux limites en relation linéaire, nous effectuons un changement de variables pour nous ramener au cas précédent.

Supposons avoir l'équation

$$\alpha_1 u_{i_1} + \alpha_2 u_{i_2} + \ldots + \alpha_m u_{i_m} = \beta.$$

Nous posons

$$\begin{cases} U_{i_1} = u_{i_1}, \\ \vdots \\ U_{i_{m-1}} = u_{i_{m-1}}, \\ \\ U_{i_m} = \alpha_1 u_{i_1} + \ldots + \alpha_m u_{i_m}. \end{cases}$$

On peut alors fixer la variable U_{i_m} à β et

$$u_{i_m} = \frac{U_{i_m}}{\alpha_{i_m}} - \frac{\alpha_1}{\alpha_{i_m}} U_{i_1} - \ldots - \frac{\alpha_{i_{m-1}}}{\alpha_{i_m}} U_{i_{m-1}}.$$

Remarques.

• On prend α_{i_m} le maximum des α_i. Donc une même inconnue ne peut pas apparaître dans plusieurs conditions en relation linéaire avec un coefficient maximal.

• Avec cette méthode, le profil est modifié. C'est pourquoi PREPAC a besoin de la S.D. BDCL.

V.4. Les méthodes itératives

Les méthodes que l'on retrouve dans Modulef, sont les méthodes classiques, de type gradient conjugué, pour la résolution d'un système linéaire. On distingue

– la méthode classique du gradient conjugué préconditionné ou non, utilisable lorsque la matrice A du système linéaire à résoudre est **symétrique, définie positive**.

– la méthode du double gradient conjugué accélérée préconditionnée par factorisation incomplète de la matrice A, dans le cas **non symétrique**.

La matrice est stockée dans la S.D. AMAT. Celle-ci est composée de 7 tableaux dont l'organisation est la suivante.

1. AMA0: informations générales,
2. AMA1: descripteurs des tableaux supplémentaires éventuels,
3. AMA2: description générale de la matrice,
4. AMA3: pointeurs sur la dernière ligne de chaque page,
5. AMA4: pointeurs sur le coefficient diagonal de chaque ligne,
6. AMA5: pointeurs sur les colonnes,
7. AMA6: coefficients de la matrice.

Considérons par exemple la matrice d'ordre 9 symétrique suivante.

$$\begin{pmatrix}
1 & & & & & & & & \\
2 & 3 & & & & & & & \\
4 & 5 & 6 & & & & & & \\
. & 7 & . & 8 & & & & & \\
. & . & 9 & 10 & 11 & & & & \\
. & 12 & . & 13 & . & 14 & & & \\
. & . & . & . & 15 & 16 & 17 & & \\
. & . & . & 18 & . & . & . & 19 & \\
. & . & . & . & . & . & . & . & 20
\end{pmatrix}$$

Remarquons que les valeurs sont les rangs dans le stockage et non pas les coefficients de la matrice.

Nous avons dans ce cas les tableaux ci-après.

$$\begin{aligned}
AMA3 &= (0\ 9),\\
AMA4 &= (0\ 1\ 3\ 6\ 8\ 11\ 14\ 17\ 19\ 20),\\
AMA5 &= (1\ 1\ 2\ 1\ 2\ 3\ 2\ 4\ 3\ 4\ 5\ \ldots\ 9),\\
AMA6 &= (A_{11}\ A_{21}\ A_{22}\ A_{31}\ \ldots A_{99}).
\end{aligned}$$

La ligne i est rangée dans $AMAT6$ entre les adresses

$$AMAT4(i) + 1 \qquad \text{et} \qquad AMAT4(i+1).$$

Dans le cas non symétrique, on adopte la même forme de stockage en gardant tous les coefficients non nuls de la ligne complète, en les rangeant par indice de colonne croissant, à l'exception du coefficient diagonal, placé en bout de ligne.

Ce mode de stockage est appelé **stockage compact**. Il est aussi désigné sous le nom de **stockage morse**.

V.5. Résolution par décomposition de domaines

La résolution numérique de problèmes de grande taille par des techniques de décomposition de domaines est très bien adaptée aux ordinateurs parallèles de la génération actuelle.

Cependant l'efficacité de ces techniques dépend fortement de l'algorithme choisi et de sa mise en œuvre.

L'approche proposée dans Modulef partage le domaine de calcul en sous-domaines non structurés de forme arbitraire, et réduit le problème initial à un problème d'interface.

L'opérateur associé (de Steklov-Poincaré au niveau continu, la matrice complément de Schur au niveau discret) est inversé par un algorithme de gradient conjugué préconditionné.

Cet algorithme exige à chaque étape la résolution sur chacun des sous-domaines d'un problème de Dirichlet et d'un problème de Neumann.

Pour utiliser cette méthode, on utilise les modules PRSDOM, SDOMVR.

PRSDOM: créer à partir de la S.D. NOPO des différents sous-domaines, les opérateurs nécessaires à l'algorithme du gradient conjugué.

SDOMVR: réaliser les itérations du gradient conjugé sur l'interface.