Factorisation symbolique et réordonnancement des matrices creuses

À partir d'un code existant, écrit en FORTRAN 90 et dans lequel les types "matrice creuse" et "graphe" ont été prédéfinis ainsi que des opérations élémentaires s'y rapportant, l'objectif du TP est d'étudier l'influence du réordonnancement de la matrice sur le remplissage durant la factorisation de Gauss.

À partir d'un code permettant d'effectuer la factorisation symbolique d'une matrice, il sera demandé de développer des algorithmes de réordonnancement classiques (degré minimum et Cuthill Mc Kee) vus en cours.

Les codes développés seront validés sur un jeu de problèmes synthétiques et leur performance sera analysée sur des problèmes réels puis comparés aux outils fournis par matlab.

1 Factorisation symbolique et algorithme du degré minimum

Soit $G^k = (V^k, E^k)$ le graphe d'élimination (voir cours). Ce graphe permer de décrire les évolutions de la structure de la matrice durant la factorisation : le remplissage dans la matrice à l'étape k de la factorisation correspond à de nouvelles arêtes introduite à l'étape k de la construction du graphe d'élimination G^k. Dans l'Algorithme 1 chaque stratégie de choix de pivot (ligne α) conduira à une instance de cet algorithme générique :

— dans l'ordre naturel,
— dans un ordre donné (permutation donnée),
— dans l'ordre déterminé par celui de degré minimum dans le graphe d'élimination.

Algorithme 1 (Factorisation symbolique et degré minimum)

$\forall i \in [1 \cdots n] \quad t_i = |\mathrm{adj}_{G^0}(i)|$
For $k = 1$ **to** n **Do**
α) **CHOISIR un PIVOT** p
β) **For** *each* $i \in \mathrm{adj}_{G^{k-1}}(p)$ **Do**
    $\mathrm{adj}_{G^k}(i) = (\mathrm{adj}_{G^{k-1}}(i) \cup \mathrm{adj}_{G^{k-1}}(p)) \setminus \{i, p\}$
    (en comptant le nombre de nouvelles arêtes dans G^k)
    $t_i = |\mathrm{adj}_{G^k}(i)|$
  EndFor
  $V^k = V^{k-1} \setminus p$
EndFor

On notera que l'étape β de l'Algorithme 1 est réalisée par la procédure `elimination` du module `fsmdcm` et est décrite en Section 4.2.

2 Algorithme de Cuthill-McKee

L'algorithme de Cuthill-McKee correspond à un parcours en largeur du graphe associé à une matrice symétrique (voir Exercice sur le Breath first Search traversal of a graph fait en cours). Soit $G = (V, E)$ le graphe associé à la matrice creuse.

Algorithme 2 (Cuthill-McKee)

Choisir un noeud v de degré minimum dans G
Nombre de sets : `NBsets`$= 1$
$S \leftarrow \{v\}$ *et marquer tous les autres noeuds de V comme non visités*
while $S \neq V$ ***do***
  $S' \leftarrow$ *tous les noeuds de $V \setminus S$ adjacents à S et non visités*
  if S' *non vide* ***then***
    Marquer comme visités les noeuds de S'
    $S \leftarrow S \cup S'$ *;* `NBsets=NBsets`$+1$
  else
    Redémarrer d'un noeud non visité de degré minimum
  end if
end while

3 Travail à faire

1. Réordonnancement basé sur le degré minimum
   — Construire le graphe associé à la matrice `mat0` et appliquer l'algorithme du degré minimum **(sur papier)**.
   — Ecrire la subroutine `pivot_deg_min` du module `fsmdcm` qui renvoie un pivot de degré le plus faible du graphe.
   — Sur le modèle du code (`factorisation_symbolique`) permettant de réaliser la factorisation symbolique, écrire la suboutine `degre_minimum` qui renvoie outre le remplissage la permutation des pivots obtenue (voir interface de la routine). La subroutine `pivot_deg_min` sera utilisée.

   Les descriptions des structures de données et des codes fournis sont dans la section 4.
2. Réordonnancement basé sur l'algorithme de Cuthill-McKee
   — Appliquer l'algorithme de Cuthill-McKee au graphe de la matrice `mat0` **(sur papier)**.
   — L'interface de la subroutine `CMcK` qui implémente l'Algorithme 2 est fournie et devra être respectée.
   — Décrire les structures de données permettant d'implanter l'Algorithme 2 sans effectuer de copie du graphe ni d'allocation supplémentaire de tableau en mémoire.
   — Ecrire le code et le valider.
   — Vous noterez que dans le programme principal `validation.f90`, la permutation inverse de celle renvoyée par `CMcK` conduisant ainsi à l'ordering dit du **Reverse Cuthill-McKee ou RCM** est également testée.
3. Adapter/Compléter le fichier de commandes `mf.m` pour visualiser le comportement des permutations calculées dans le programme principal (`perm_matlabMD, perm_matlabCM, perm_matlabCM`).

Vous noterez les résultats obtenus sur chaque matrice de test dans le tableau fourni en Section 6.

4. Amélioration de l'algorithme de Cuthill-McKee
   Comme indiqué en cours, les algorithmes de Cuthill-McKee (et Reverse Cuthill-McKee) dépendent du choix d'un noeud de départ. Partant des trois remarques suivantes :
   — le remplissage est en général plus faible lorsque le nombre de niveaux est plus important.
   — le nombre de niveaux détectés par l'algorithme de Cuthill-McKee est en général plus élévé si le noeud de départ est situé en périphérie du graphe.
   — les noeuds du dernier niveau de l'algorithme de Cuthill-McKee ont tendance à être en périphérie du graphe.

   Proposer un algorithme permettant d'augmenter le nombre d'ensembles (`NBsets`) en exploitant l'algorithme (et l'interface proposée) de Cuthill-McKee. Modifier le programme principal (validation.f90) pour valider/expérimenter votre algorithme.

4 Descriptions des structures de données et des codes fournis

Le répertoire `Src` comprend un programme principal (fichier `validation.f90`), un module de définition du type "matrice creuse" (fichier `definition.f90`), un exemple de matrice creuse (fichier `mat0`) et un répertoire de matrices toutes au format CSC (Compressed Sparse Columns).

4.1 Structure de données associée au graphe

À toute matrice creuse on peut associer un graphe traduisant sa topologie. Le graphe (G) est une donnée déclarée dans le programme principal (*validation*). C'est un assemblage de cellules élémentaires dont le type (*cell_graphe*) est déclaré dans l'en-tête du module $definition$. Une cellule élémentaire est composée d'un entier (*indice*) et d'un pointeur vers une autre cellule élémentaire (*suivant*).

Le graphe résultant est ainsi un tableau de cellules. La dimension de ce tableau est égale au nombre de colonnes de la matrice. Pour un pivot donné, *indice* contient le degré du pivot et *suivant* pointe sur une liste chaînée de cellules identique à la liste d'adjacence du pivot.

Soit par exemple le pivot de numéro i. Pour une cellule de sa liste d'adjacence, *indice* contient un numéro de colonne j (i.e. le coefficient a_{ij} est donc non nul) et *suivant* pointe vers l'élément non nul suivant de la ligne i. L'exemple suivant est représentatif de la structure graphe :

$$
\begin{pmatrix}
\bullet & \bullet & \circ & \circ & \bullet & \circ \\
\bullet & \bullet & \circ & \bullet & \circ & \circ \\
\circ & \circ & \bullet & \circ & \circ & \bullet \\
\circ & \bullet & \circ & \bullet & \circ & \circ \\
\bullet & \circ & \circ & \circ & \bullet & \bullet \\
\circ & \circ & \bullet & \circ & \bullet & \bullet
\end{pmatrix}
\qquad
\begin{array}{|c|c|l}
\hline
2 & \rangle & \longrightarrow 2 \longrightarrow 5 \longrightarrow \mathit{null} \\ \hline
2 & \rangle & \longrightarrow 1 \longrightarrow 4 \longrightarrow \mathit{null} \\ \hline
1 & \rangle & \longrightarrow 6 \longrightarrow \mathit{null} \\ \hline
1 & \rangle & \longrightarrow 2 \longrightarrow \mathit{null} \\ \hline
2 & \rangle & \longrightarrow 1 \longrightarrow 6 \longrightarrow \mathit{null} \\ \hline
2 & \rangle & \longrightarrow 3 \longrightarrow 5 \longrightarrow \mathit{null} \\ \hline
\end{array}
$$

4.2 Description de la procédure *elimination* du module *definition*

```
SUBROUTINE elimination(g, dim, compt, pivot, trace)
```

Cette procédure possède comme paramètres d'entrée un graphe g comportant dim pivots et un numéro de pivot (*pivot*) à éliminer. Elle réalise l'élimination du pivot avec mise à jour du graphe. *compt*, de type entier, est un paramètre qui comptabilise le remplissage provoqué par cette élimination. Ce paramètre est mis à jour à chaque appel d'*elimination* : à la valeur transmise en entrée est ajouté le nouveau remplissage provoqué par l'élimination.

En sortie le champ *g(i)%indice* est passé à la valeur *-1* pour indiquer que le pivot a été éliminé. *trace*, de type booléen, fournit une trace de l'élimination si positionné à vrai.

Un exemple d'exécution de la procédure *elimination* est donné ci-dessous.

Elimination du noeud 1
dans l'exemple du 1.1.2
Fill-in en (2,5) et (5,2)
compt = 2

$$
\begin{pmatrix}
\bullet & \bullet & \circ & \circ & \bullet & \circ \\
\bullet & \bullet & \circ & \bullet & \otimes & \circ \\
\circ & \circ & \bullet & \circ & \circ & \bullet \\
\circ & \bullet & \circ & \bullet & \circ & \circ \\
\bullet & \otimes & \circ & \circ & \bullet & \bullet \\
\circ & \circ & \bullet & \circ & \bullet & \bullet
\end{pmatrix}
\qquad
\begin{array}{|c|c|l}
\hline
\textbf{-1} & \rangle & \longrightarrow \textbf{null} \\ \hline
2 & \rangle & \longrightarrow 4 \longrightarrow \boxed{\text{5 (fill-in)}} \longrightarrow \mathit{null} \\ \hline
1 & \rangle & \longrightarrow 6 \longrightarrow \mathit{null} \\ \hline
1 & \rangle & \longrightarrow 2 \longrightarrow \mathit{null} \\ \hline
2 & \rangle & \longrightarrow 6 \longrightarrow \boxed{\text{2 (fill-in)}} \longrightarrow \mathit{null} \\ \hline
2 & \rangle & \longrightarrow 3 \longrightarrow 5 \longrightarrow \mathit{null} \\ \hline
\end{array}
$$

Elimination du noeud 2
Fill-in en (4,5) et (5,4)
compt = compt + 2 = 4

$$\begin{pmatrix} \bullet & \bullet & \circ & \circ & \bullet & \circ \\ \bullet & \bullet & \circ & \bullet & \otimes & \circ \\ \circ & \circ & \bullet & \circ & \circ & \bullet \\ \circ & \bullet & \circ & \bullet & \otimes & \circ \\ \bullet & \otimes & \circ & \otimes & \bullet & \bullet \\ \circ & \circ & \bullet & \circ & \bullet & \bullet \end{pmatrix}$$

-1	$\rangle$	$\longrightarrow$ **null**
-1	$\rangle$	$\longrightarrow$ **null**
1	$\rangle$	$\longrightarrow$ 6 $\longrightarrow$ *null*
1	$\rangle$	$\longrightarrow$ $\boxed{\text{5 (fill-in)}}$ $\longrightarrow$ *null*
2	$\rangle$	$\longrightarrow$ 6 $\longrightarrow$ $\boxed{\text{4 (fill-in)}}$ $\longrightarrow$ *null*
2	$\rangle$	$\longrightarrow$ 3 $\longrightarrow$ 5 $\longrightarrow$ *null*

4.3 Programme principal

Le programme principal qui vous est fourni (fichier `validation.f90`) comprend : la référence au module `definition`, les déclarations de données et un embryon de code où la matrice est créée et éditée, puis le graphe également créé et édité.

Pour pouvoir accéder au fichier associé à votre matrice de test il vous suffit de lancer l'exécution en redirigeant l'entrée standard i.e. exécuter la commande :

```
validation    <mat0 ou validation < ../Matrices/nom_matrice
```

4.4 Expérimentation et visualisation en MATLAB

Pour que votre code FORTRAN génère les fichiers utilisés par le code matlab fourni, il faut que le booléen **printmatlab** soit positionné à *true* (ligne 35).

Le fichier matlab $Src/mf.m$ contient un ensemble de commandes MATLAB qui réalisent les opérations suivantes :

— Initialisation d'une matrice creuse, à partir du fichier `mat_matlab`, et sa visualisation ;
— Factorisation symbolique de cette matrice, visualisation de la matrice factorisée et calcul du remplissage total ;
— Chargement, à partir du fichier `perm_matlabMD`, d'un nouvel ordonnancement, ré-ordonnancement de la matrice, visualisation du résultat ;
— Factorisation de la matrice permutée et visualisation de la matrice factorisée ;
— Visualisation du remplissage.

5 Interface de la subroutine CMcK (Cuthill Mc Kee)

```
SUBROUTINE CMcK(g, dim, start, perm, nbSets, &
                    iperiph, trace, retour)
! Applique l'algorithme de Cuthill Mc Kee direct au graphe associé à
! une matrice creuse.
! Produit comme résultat un ordonnancement.
! (Utilise la procédure pivot_deg_min qui recherche dans un
! graphe le pivot de degré minimum.)
! ------------
! g        (inout): graphe représentant la matrice (g est modifié)
! dim      (in)   : nombre de noeuds dans de graphe
! start    (in)   : indice du noeud de depart de l'algorithme
!                    si (start == 0) alors le noeud de degre minimum sera
!                                   pris comme noeud de depart
! perm     (out)  : tableau de permutation
!                    perm(i)=j indique que le noeud j est le ieme pivot.
```

```
! nbSets   (out)  : nombre de niveaux traversés
! iperiph  (out)  : Indice de noeud d'un sommet du dernier set
! trace    (in)   : vrai si traces d'exécution demandée
! retour   (out)  : paramètre de retour, 0 si OK
!
        integer, intent(in) ::  dim,start
        type (cell\_graphe),dimension(dim),intent(inout)::g
        integer, intent(out)  :: perm(dim)
        integer, intent(out)  :: nbSets, iperiph
        logical, intent(in)   :: trace  ! vrai si impression demandee
        integer, intent(out)  :: retour
!   ...
        end SUBROUTINE CMcK
```

6 Données pour validation et tests

Matrice	Fact. symbolique	Degré minimum	CMcK direct(*)	CMcK inverse(*)
mat0	4	-	-	-
mat1	14	-	-	-
mat2	12	-	-	-
mat3	12	-	-	-
mat4	20	-	-	-
mat5	0	-	-	-
dwt_59.mat	500	-	-	-
dwt_66.mat	492	-	-	-
dwt_307.mat	13378	-	-	-
dwt_592.mat	53098	-	-	-
dwt_1007.mat	42926	-	-	-
dwt_2680.mat	612372	-	-	-

Quelques ordonnancements :

1. Degré minimum

   mat0 : —-

2. CMcK direct

   mat0 : ——