Méthodes de Krylov préconditionnées

Dans tous les exercices, on s'intéressera au coût mémoire, à la vitesse de convergence (nombre d'itérations) et au temps de calcul associé aux choix algorithmiques (choix du préconditionneur) retenus pour la résolution d'un système linéaire

$$Ax^* = b$$

1 Préambule

Pour la résolution de systèmes linéaires creux de très grande taille, les méthodes itératives constituent une alternative aux méthodes directes (factorisation) dans le cas où la taille des facteurs devient trop importante (au regard de la mémoire disponible sur l'ordinateur cible), ou lorsque les temps de factorisation deviennent trop élevés ou encore lorsque la contrainte de précision numérique peut être relâchée (schéma non-linéaire, imprécision sur les données).

Dans ce contexte, les méthodes de Krylov constituent une alternative souvent retenue dans les grands codes de simulation. Ces méthodes (Gradient Conjugué dans le cas symétrique défini positif, GMRES non-symétrique général, ...) convergent d'autant plus vite que la matrice du système à résoudre est ≪proche≫ de la matrice identité.

Ceci est illustré en particulier par la borne suivante qui peut être établie pour la vitesse de convergence du gradient conjugué (voir cours) :

$$\|x_k - x^*\|_A \leq 2 \cdot \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|x_0 - x^*\|_A.$$

où $\kappa(A) = \dfrac{\lambda_{\max}}{\lambda_{\min}}$ et $\|x\|_A^2 = x^T A x$.

Afin de satisfaire à cette contrainte, des techniques de préconditionnement sont utilisées. Celles-ci consistent à résoudre un système linéaire équivalent :

$$M^{-1}Ax = M^{-1}b$$

où la matrice M est appelée préconditionneur.

Les idées qui gouvernent sa construction sont :

1. M doit être facile à construire,
2. M doit être facile à utiliser,
3. M doit être peu coûteuse en mémoire,
4. M doit être la meilleure approximation possible de A.

Parmi les préconditionneurs algébriques classiques, on compte les factorisations incomplètes de type Cholesky (matrice symétrique définie positive) et LU (matrice quelconque) qui permettent d'exprimer M sous la forme $M = M_1.M_2$.

2 Partie 1 : Résolution d'un système linéaire associé à une matrice issue de la discrétisation d'une EDP

On cherche à résoudre le système linéaire : $Ax = b$ où A est la matrice issue de la discrétisation par éléments finis d'une EDP de type elliptique (voir fichier pdf explicatif).

> Vous commencerez par vérifier que les matrices `mat0`, `mat1`, `mat2` et `mat3` sont symétriques définies positives ce qui justifie que l'algorithme de résolution de système linéaire que nous allons utiliser est le Gradient Conjugué (Préconditionné).

En utilisant la méthode du gradient conjugué (`pcg`), vous étudierez la vitesse de convergence pour différentes tailles du système linéaire (lorsque le maillage est raffiné).

Vous tracerez en particulier, l'historique de convergence de l'erreur inverse *normwise*

$$\eta_b^N(x_k) = \frac{\|r_k\|}{\|b\|} = \frac{\|b - Ax_k\|}{\|b\|}$$

où r_k est le résidu associé à l'itéré x_k de la k^{ieme} itération.

Vous utiliserez différents préconditionneurs :

1. aucun (préconditionneur identité),
2. préconditionneur de Jacobi (préconditionneur diagonal),
3. préconditionneur issu de la factorisation incomplète de Cholesky sans remplissage ($IC('0')$).
4. préconditionneur issu de la factorisation incomplète de Cholesky (IC) avec «threshold» (threshold = $1.e^{-3}$, $5e^{-4}$ et $1e^{-4}$ par exemple).

Pour les préconditionneurs issus de la factorisation incomplète, vous afficherez la structure de A et celle du facteur de Cholesky sur la même figure.

Fournitures partie 1

1. les fichiers `tubeG.m, tubeB.m, tubeF.m` et `setupC.m` de définition de l'EDP ainsi qu'un fichier pdf explicatif (utilisation de la PDE Toolbox de Matlab)
2. **le fichier `cholinc_n7.m` pour la factorisation incomplète de Cholesky avec «threshold».**
3. le fichier `tp.m` canevas à compléter aux endroits indiqués.

Fonctions Matlab qui peuvent être utiles

— **semilogy**
— **spy**
— **eye, diag, triu**
— **pcg**
— **ichol** pour la factorisation incomplète de Cholesky sans remplissage
— **cholinc_n7** pour la factorisation incomplète de Cholesky avec threshold

3 Partie 2 : Utilisation d'un préconditionneur multi-grille

Parmi les autres types de préconditionneur, on retrouve ceux construits en utilisant la méthode multigrille. Au lieu d'utiliser la méthode multigrille pour résoudre le système linéaire comme nous l'avons fait lors du TP2, nous allons construire un préconditionneur en nous appuyant sur ce qui est effectué dans **une itération de la méthode multigrille**.

Nous utilisons de nouveau **FreeFEM** et nous allons compléter le fichier **mg_precond.edp** fourni qui repart de la solution de la seconde partie du TP2.

Nous considérerons le cas du préconditionneur avec deux grilles.

Pour se faire, nous allons utiliser la routine du langage FreeFEM qui permet d'invoquer `PCG` de la façon suivante (voir fichier).

```
LinearCG(op, xF[], b[], precon = preconditioner, verbosity = 40,
         nbiter = it, eps = eps);
```

Travail demandé :

les paramètres `op` et `preconditioner` sont des fonctions FreeFEM à implémenter qui doivent retourner respectivement les vecteurs y et z tels que :

$$y = Ax \qquad z = M^{-1}v.$$

Implémenter ces fonctions en respectant le prototype suivant :

```
func real[int] op(real[int]& x) {
    VhFine y;
    /* TODO */
    return y[];
}

func real[int] preconditioner(real[int]& v) {
    VhFine z;
    /* TODO : adapter une itération du point fixe de la méthode multigrille */
    /* v en entrée joue le rôle de b, x -> résultat z */
    return z[];
}
```