Méthodes Itératives

3SN HPC-BD

TP2 Méthodes Multigrilles

Année scolaire 2023–2024

Carola Kruse Ronan Guivarch

1 Introduction

Dans ce TP, nous allons d'abord implémenter une méthode multigrilles en 1D dans Matlab et ensuite utiliser FreeFem pour construire un solveur multigrilles en 2D.

2 Multigrilles 1D (Matlab)

Nous allons développer un solveur multigrilles pour résoudre le problème de Poisson avec des conditions aux limites de Dirichlet homogènes dans une dimension

$$\begin{aligned} -u'' &= f, && \text{in } \Omega = [0,1] \\ u &= 0, && \text{on } \Gamma \end{aligned}$$

Dans la suite, nous supposons que la solution exacte de ce problème de Poisson est donné par

$$u(x) = x(1-x). \tag{1}$$

Il est facile de vérifier que cette solution respecte les conditions aux limites de Dirichlet (comment ?). Nous utilisons cette solution pour trouver le membre de droite

$$f = -u'' = 2. \tag{2}$$

En choisissant une solution exacte, il est possible de calculer les erreurs introduites par la discrétisaton par différences finies. Elle peut ainsi être utilisée pour valider l'implémentation de la discrétisation.

2.1 Maillage

Nous nous concentrons sur un solveur à deux niveaux, ça veut dire que nous avons une hiérarchie de seulement deux maillages. Pour cela, soit N le nombre d'éléments dans l'intervalle $[0,1]$. Nous allons donc définir les deux domaines suivants avec $h = \frac{1}{N}$

$$\begin{aligned} &\Omega_h, && x_i = ih, && i = 0,..,N && \text{(fine grid)}, \\ &\Omega_{2h}, && x_i = 2ih, && i = 0,..,N/2 && \text{(coarse grid)}. \end{aligned}$$

En règle générale, nous devrions construire ce maillage et utiliser les informations pour obtenir la matrice et les opérateurs de transfert (voir partie **FreeFem**). Pour cet exemple simple , ce n'est par contre pas nécessaire et tous les calculs peuvent être faits à la main.

2.2 Discrétisation par différences finies

Nous utilisons l'approche *différence centrée*

$$u_i'' \approx \frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i, \quad i = 1, .., N-1 \tag{3}$$

Avec cette discrétisation, nous obtenons, sur la grille fine, le système linéaire

$$A\mathbf{u} = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{pmatrix} \tag{4}$$

Comme nous avons des conditions aux limites de Dirichlet homogènes, ce n'est pas nécessaire d'inclure les deux degrés de libertés u_0 et u_N dans la matrice. La matrice du problème, sur la grille fine, est donc de taille $N-1$.

Travail demandé :

- Écrivez une fonction Matlab avec N en entrée et $A \in \mathbb{R}^{N-1 \times N-1}$ en sortie

```
function A = getMatrixA(N)
% Your code here
```

Astuce: Vous pouvez utiliser la commande `diag(vec,m)` de Matlab, où `vec` correspond au vecteur des valeurs sur la diagonale, et $m = 0$ est la diagonale, $m = 1$ et $m = -1$ sont la super- et sous-diagonale (`help diag` pour plus d'information).

Pour obtenir le membre de droite dans Matlab, nous créons un vecteur de longueur $N-1$ et le remplissons avec la valeur 2.

```
rhsf = 2*ones(N-1,1);
```

Nous sommes maintenant en mesure de calculer la solution (jusqu'à précision de machine) du système linéaire (4), que nous allons utiliser comme référence pour la solution obtenue par la méthode de multigrilles. Pour cela, nous faisons

```
sol_ref = A \ rhsf;
```

2.3 Lisseur

Pendant le cours, nous avons vu deux lisseurs, la méthode de Jacobi pondérée et de Gauss-Seidel pondérée. Ici, nous utilisons la méthode de Jacobi pondérée

$$\mathbf{u}^{(m+1)} = (I - \omega D^{-1}A)\mathbf{u}^m + \omega D^{-1}\mathbf{f}, \tag{5}$$

avec $0 \leq \omega \leq \frac{2}{\lambda_{max}(D^{-1}A)}$

Travail demandé : Implémentez une fonction qui applique m itérations de Jacobi pondérée à une fonction $u^{(m)}$ avec $u^{(m+1)}$ en sortie

```
function ump1 = weighted_jacobi(A,um,f,omega,m)
% Your code here
```

Ensuite, nous illustrons l'effet du lisseur sur un vecteur. D'abord, nous créons un vecteur correspondant au k-ème vecteur propre de la matrice A sur un maillage avec 64 éléments. Après, nous appliquons m itérations du Jacobi.

```
clear all;
N = 64;
h = 1/N;
A = getMatrixA(N);
omega = 2/3;
rhsf = zeros(N-1,1);
j = 2:N;

k = 6;
% kieme vecteur propre
um = sin(j/N*k*pi);

m = 10;
ump1 = weighted_jacobi(A,um',rhsf,omega,m);

x=h:h:1-h;
plot(x,um,x, ump1);
legend('um','ump1');
title('Damping effect of weighted Jacobi method')
```

Travail demandé : Variez

- le nombre d'itérations de Jacobi,
- le paramètre ω dans la méthode de Jacobi.
- le vecteur propre en utilisant $k \in \{3, 12, 48\}$,

Qu'est-ce que vous observez ?

2.4 Opérateurs de transfert

Pour transférer les informations d'un maillage à l'autre, il nous faut des opérateurs de transfert inter-grilles. Dans ce simple exemple, nous utilisons l'interpolation linéaire pour la prolongation

$I_{2h}^{h} : \mathbb{R}^{\frac{N}{2}-1} \to \mathbb{R}^{N-1}$ avec

$$v_{2j}^{h} = v_{j}^{2h},$$
$$v_{2j+1}^{h} = \frac{1}{2}(v_{j}^{2h} + v_{j+1}^{2h})$$

et *full weighting* pour la restriction $I_{h}^{2h} : \mathbb{R}^{N-1} \to \mathbb{R}^{\frac{N}{2}-1}$ avec

$$v_{j}^{2h} = \frac{1}{4}(v_{2j-1}^{h} + 2v_{2j}^{h} + v_{2j+1}^{h})$$

Travail demandé : Implémentez l'opérateur d'interpolation I2hh et l'opérateur de restriction Ih2h dans une forme matricielle comme vu en cours. Quel est l'avantage de choisir le couple interpolation et full weighting ?

```
% Interpolation
function I2hh = interpol(N)
% Your code here
```

```
% Restriction

Ih2h = ?
```

2.5 Schéma de multigrilles à deux niveaux

Nous pouvons maintenant définir la méthode de multigrilles sur deux niveaux. Remplissez le code suivant. Nous commençons avec le setup des éléments que nous avons besoin :

```
clear all;
% Setup maillage
N = 64;
h = 1/N;

% Setup Jacobi
omega = 2/3;

% Setup of the fine and coarse grid matrix
% and the right-hand side
Ah = getMatrixA(N);
A2h = getMatrixA(N/2);

rhsf = 2*ones(N-1,1);

% Compute direct solution of linear system
sol_ref = ?

% Setup interpolation matrix
I2hh = interpol(N);
Ih2h = ?
```

Maintenant, nous définissons une méthode de multigrilles en 1D. Completez le code suivant :

```
% Initial vector
v(1:N-1,1) = 0;

% Vector to store the error at each iteration.
% We do 10 iterations.
err(1:10) = 0;

% Multigrid iterations with 2 pre-smoothing steps

for i=1:10
  v = weighted_jacobi(Ah,v,rhsf,omega,2);

  % residual on fine grid
  res_h = ?

  % Restriction of residual to coarser grid
  res_2h = ?

  % Solve the coarse grid error equation
  e_2h = ?

  % Interpolate the coarse grid error to the fine grid
  e_h = ?

  % Update the approximate fine grid solution
  v = v + ?

  % Compute the error with respect to the direct
  % solution of the linear system
  err(i) = norm(sol_ref-v);
end
```

- Que constatez-vous pour l'erreur calculée ?
- Peut-on facilement généraliser ce système à plus que deux niveaux ? Quelle ligne devrait être remplacée ?

Remarque : Veuillez noter que la solution directe du système linéaire n'est généralement pas disponible. Au lieu de cela, nous pouvons calculer le résidu à chaque étape du processus itératif pour constater une réduction de l'erreur et définir un critère d'arrêt

$$q^{(m)} = \frac{\|res_h^{(m)}\|_2}{\|res_h^{(0)}\|_2} = \frac{\|b - Ax^{(m)}\|_2}{\|b - Ax^{(0)}\|_2}$$

Dans l'exemple, nous avons utilisé l'erreur algébrique à des fins d'illustration.

3 Multigrille 2D (FreeFem)

Le but de cette partie est de développer un solveur multi-grilles géométriques en utilisant les outils mis à disposition par le langage FreeFEM. Ce langage permet entre autre de construire des matrices de passage pour aller d'un fespace `Vh` à un fespace `Zh` en utilisant la fonction `interpolate` :

```
matrix R = interpolate(Zh, Vh);
```

On obtient ainsi une matrice creuse de taille `Zh.ndof` $\times$ `Vh.ndof` qui permet de manipuler des fonctions éléments finis vivant dans l'un des deux espaces EF. Nous allons voir dans ce TP comment :

- utiliser les méthodes multi-grilles dans le cadre d'une méthode itérative de point fixe, cf. section 3.1,
- préconditionner une méthode de Krylov avec une méthode multi-grilles, cf. TP3.

Dans la suite du TP, nous allons considérer le système à résoudre suivant :

$$Ax = b, \tag{6}$$

où A est une matrice carrée, b un second membre, et x la solution. On suppose que cette représentation algébrique provient de la discrétisation de l'équation de Poisson en deux dimensions :

$$\begin{aligned} -\Delta u &= f \quad \text{dans } \Omega, \\ u &= 0 \quad \text{sur } \Gamma, \end{aligned} \tag{7}$$

avec $f(x, y) = 2y(1 - y) + 2x(1 - x)$.

En FreeFEM, il est possible de définir des tableaux indexés par des entiers de la façon suivante : `type[int]`, où `type` peut représenter un maillage (`mesh`), un entier (`int`), un réel (`real`), ou bien encore un tableau de réels (`real[int]`). De plus, les tableaux de type réel disposent d'un membre `.l2` **permettant de calculer leur norme** l_2. Ce formalisme va nous permettre de définir une hiérarchie de grille.

3.1 Méthode de point fixe

Une méthode de point fixe avec un opérateur A et un préconditionneur M^{-1} peut s'écrire :

Input: solution initiale x_0, second membre b
for $i = 0, 1, \ldots$ **do**
  $r_i = b - Ax_i$
  $e_i = M^{-1} r_i$
  $x_{i+1} = x_i + e$
  $i = i + 1$

Un squelette de code vous est fourni pour cette partie (fichier `TP_TG_base.edp`). Nous vous conseillons de faire des **copier/coller successifs** des différentes étapes pour vérifier pas à pas que vous avez un code correct au niveau syntaxique (les n° des TODO correspondent aux différentes étapes du travail demandé). Enfin, pensez à utiliser ce qui a été fait au TP1.

Travail demandé :

1. générer deux maillages d'un même domaine (par exemple un carré) : l'un fin, l'autre grossier (pensez à utiliser la variable `coarsening`).
2. définir un espace éléments finis sur chacun des maillages (cf TP1), puis construire les opérateurs d'interpolation et de restriction nécessaire aux transferts inter-grilles.
3. implémenter une méthode deux grilles pour résoudre eq. (6) avec un lisseur de type *weighted* Jacobi, les paramètres suivant pourront être ajustés à l'exécution :
   (a) facteur de raffinement entre le maillage fin et le maillage grossier,
   (b) nombre d'applications du lisseur,
   (c) le facteur ω du *weigthed* Jacobi,
   (d) nombre d'itérations maximum,
   (e) critère d'arrêt en fonction de la norme du résidu.
4. Constatez que le nombre d'itérations varient peu suivant les grilles utilisées.
5. reprendre le problème en modifiant le second membre $f(x, y) = (1 + 5x)\sin(10x^2 + 20y^2)$ et en ne plaçant les conditions de Dirichlet homogènes que sur $\Gamma_1 \cup \Gamma_4$.

3.2 Index des fonctionnalités utilisées

Types :

- real, real[int], int
- mesh, mesh[int]
- matrix, matrix[int]
- varf
- func

Fonctions :

- square
- trunc
- plot
- interpolate
- set