Analyse numérique avancée – TP 1

M. Kern

24 février 2022

Exercice 1 : Méthodes directes et itératives

On veut comparer le coût d'une méthode directe et d'une méthode itérative pour la résolution d'un système linéaire $Ax = b$, où A est la matrice du Laplacien discrétisé par différences finies, d'abord en 2D, puis en 3D.

1 En 2D, comparer la solution pris par une méthode directe, par le gradient conjugué sans préconditionnement, puis avec un préconditionnement par factorisation incomplète.

Compléter le script `lap2Dcomp.m`. Afficher le nombre d'itérations du gradient conjugué.

2 Reprendre la question pour un problème 3D. Utiliser la fonction `lap3Dcomp`, avec une valeur plus petite de N ($N = 20$ ou $N = 30$).

Le script compare la structures des facteurs de Cholesky « naïf » et avec une renumérotation. Expliquer les arguments, commenter.

Exercice 2 : Convergence du gradient conjugué : influence du pas de discrétisation

On reprend l'exemple du Laplacien sur le carré unité (voir exercice Exercice 1 :). On compare la convergence du gradient conjugué :

1. sans préconditionnement ;
2. avec préconditionnement par factorisation incomplète (`L=ichol(A)`) ;
3. avec préconditionnement par factorisation incomplète modifiée (`L=ichol(A, struct('michol', 'on'))`) ;

1 Étudier la convergence des différentes méthodes en fonction du pas h de discrétisation par coté du carré. Comment le nombre d'itérations dépend-il de h ? Compléter le script `lapconv1.m`.

2 Fixer une valeur de h. Pour chaque méthode, calculer et visualiser les valeurs propres de A. Compléter le script `lapconv2.m`.

Commenter l'effet de la répartition des valeurs propres sur la convergence.

Exercice 3 : Préconditionnement par un solveur rapide de Fourier, méthode sans matrice

Cet exercice est tiré de [1]. On veut résoudre l'équation de diffusion hétérogène

$$\begin{aligned} -\mathrm{div}\,(a(x,y)\,\mathrm{grad} u) &= f \quad \text{dans } \Omega =]0,1[\times]0,1[\\ u &= 0 \quad \text{sur le bord de } \Omega. \end{aligned}$$

On utilise un méthode de différences finies sur une grille régulière, soit un schéma à 5 points, avec un pas de maillage $h = 1/(n+1)$ (et donc $N = n^2$ inconnues). Les inconnues sont $u_{ij} \approx u(x_i, x_j)$, avec $x_i = ih, i = 1, \ldots, n$. Il sera commode poser

$$u_{0,j} = u_{n+1,j} = u_{i,0} = u_{i,n+1} = 0$$

pour prendre en compte la condition de Dirichlet (mais ces valeurs ne sont pas stockées). On définit aussi

$$\alpha_{Ij} = a(x_i, x_j).$$

Avec ces notations, le produit matrice–vecteur s'écrit sous la forme :

$$\begin{aligned} (Au)_{ij} = -\frac{1}{2h^2}\Big(&(\alpha_{i+1j} + \alpha_{ij})(u_{i+1j} - u_{ij}) - (\alpha_{ij} + \alpha_{i-1j})(u_{ij} - u_{i-ij}) \\ &+(\alpha_{ij+1} + \alpha_{ij})(u_{ij+1} - u_{ij}) - (\alpha_{ij} + \alpha_{ij-1})(u_{ij} - u_{ij-1})\Big) \end{aligned} \tag{1}$$

Pour les 2 premières questions, on prendra $a(x, y) = \cos(x)$, et le second membre choisi de façon à ce que la solution discrète corresponde à $u_{\mathrm{ex}}(x, y) = xy(1-x)(1-y)x^{9/2}$.

1 Résoudre le problème sans utiliser de préconditionnement, en utilisant la fonction Matlab `pcg` en mode « sans matrice », c'est-à-dire que l'on écrira une fonction `y = fddiffhet(x)` implémentant la formule (1).

Indications : procéder sans écrire de boucle explicite. On devra passer d'une représentation « vecteur à n^2 composantes dans `pcg` à une représentation « fonction de grille à deux indices » dans la fonction `fddiffhet`. Il sera aussi commode d'utiliser des matrices de taille $(n+2) \times (n+2)$ pour prendre en compte les bords.

Modifier le script `precondfft.m` et `fddiffhet.m`

Étudier la croissance du nombre d'itérations en fonction de n (on pourra prendre n entre 10 et quelques centaines).

2 On utilise un préconditionneur basé sur la résolution du Laplacien (correspondant à $a = 1$) par une méthode dite « rapide ». Ces méthodes exploitent les fonctions propres de l'opérateur $-\Delta$ (qui sont des sinus, pour les conditions de Dirichlet) pour utiliser la FFT et résoudre le problème avec $a = 1$ pour un coût faible. Cette méthode est mise en oeuvre dans la fonction `fish2d`.

Modifier le script de la question précédente pour utiliser ce précondtionneur. Vérifier que le nombre d'itérations devient indépendant du pas de discrétisation h (donc de n).

Attention, la fonction `fish2d` doit être utilisée avec un nombre de points de la forme $n = 2^k - 1$, pour k entier. On pourra prendre $n = 31, 63, 127$,

3 Reprendre les questions précédentes dans le cas où la fonction a est $a(x,y) = \sqrt{0.1+x}$.

Exercice 4 : Convergence de GMRES pour un problème d'advection – diffusion

Cet exercice est également tiré de [1]. On résout le problème d'advection – diffusion (– réaction)

$$-\Delta u + a_1(x,y)\frac{\partial u}{\partial x} + a_2(x,y)\frac{\partial u}{\partial y} = f \quad \text{dans } \Omega =]0,1[\times]0,1[$$
$$u = 0 \quad \text{sur le bord de } \Omega,$$

où a_1, a_2 et f sont des fonctions données.

On discrétise le laplacien par le schéma à 5 points, et les dérivées d'ordre 1 par un schéma centré. À cause des termes d'ordre 1, la matrice obtenue est non-symétrique. On utilise donc GMRES pour résoudre ce système.

On prendra :

$$a_1(x,y) = 1, \quad a_2(x,y) = 20y,$$

et la même solution exacte que dans l'exercice précédent, soit $u_{\text{ex}}(x,y) = xy(1-x)(1-y)x^{9/2}$.

Comme dans l'exercie précédent, on utilise une version « sans matrice « du produit matrice vecteur.

1 Compléter le script `gmres_advdiff.m` en écrivant les fonctions anonymes réalisant le produit matrice vecteur, par une combinaison des les fonctions `lapmf` (pour le Laplacien), `dxmf` et `dymf` (pour les dérivées premières). Programmer la résolution par GMRES.

2 On préconditionne le système par sa partie « principale », soit le Laplacien, et le préconditionnement.

On peut implémenter le préconditionnement soit explicitement, soit implicitement. Les 2 méthodes doivent donner le même résultat ! La première version uilise le même produit matrice vecteur et la fonction `fish2d` pour le préconditionnement. La seconde modifie le produit matrice vecteur en intégrant le préconditionnement au produit matrice vecteur. Remarquer que le second membre doit être modifié, et que le préconditionnement est l'inverse exact de la fonction `lapmf`. Compléter le script `gmres_advdiff.m` en écrivant la fonction anonyme qui réalise le produit matrice vecteur incluant le préconditionnement.

Comparer la performance des méthodes (nombre d'itérations, erreur sur la solution) en faisant varier le nombre de points de grille, et la taille du coeficient d'advection.

Exercice 5 : Comparaison des méthodes itératives pour les problèmes non–symétriques

Cet exercice est tiré de l'article [2].

On compare 3 méthodes itératives : GMRES, QMR et LSQR (une implémentation efficace de la résolution de l'équation normale par le gradient conjugué) pour la résolution de systèmes avec des matrices non symétriques.

Les exemples, tirés de [2], sont :

Matrice C exemple en dimension 4

$$C = \begin{pmatrix} 0 & 1 & & \\ & 0 & 1 & \\ & & 0 & 1 \\ 1 & & & 0 \end{pmatrix}$$

Matrice B_1 bloc-diagonale, avec des blocs 2×2

$$B_1 = \begin{pmatrix} M_1 & & \\ & \ddots & \\ & & M_N \end{pmatrix}, \qquad \text{avec } M_j = \begin{pmatrix} 1 & j-1 \\ 0 & 1 \end{pmatrix}$$

Matrice D diagonale $D = \mathrm{diag}(x_1, \ldots, x_N)$, où les $(x_i)_{i=1,\ldots,N}$ sont des nombres aléatoires distribués uniformément entre 1 et κ, et κ est choisi tel que

$$\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2\sqrt{N}} = \delta, \qquad \delta \text{ fixé } (\delta = 10^{-6}).$$

Matrice S bloc diagonale de la forme

$$S = \begin{pmatrix} J & & & \\ & J & & \\ & & \ddots & \\ & & & J \end{pmatrix}, \qquad \text{avec } J = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Compléter la fonction `iters.m` pour appeler les trois méthodes de résolution pour chaque matrice. La fonction auxiliaire `getmat.m` permet de générer les matrices, et le script `howfast.m` orchestre le tout.

Commenter les résultats obtenus. Une des méthodes est-elle toujours meilleure que les autres ? Il pourra être utile de calculer (le plus souvent « à la main », sinon utiliser la comande `eig(full(A))`) les valeurs propres des matrices, et des matrices $A^T A$.

Références

[1] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in appleid mathematics*. SIAM, 1995.

[2] Noël M. Nachtigal, Satish C. Reddy, and Lloyd N. Trefethen. How fast are nonsymmetric matrix iterations ? *SIAM J. Matrix Anal. Appl.*, 13(3) :778–795, 1992. Iterative methods in numerical linear algebra (Copper Mountain, CO, 1990).