MA261. Introduction au calcul scientifique
Corrigé 3 : Résolution par la méthode de Cholesky des problèmes discrétisés

Le but est de résoudre les problèmes discrétisés 1d (fil, poutre), 2d (membrane), 3d (cavité), que l'on écrit sous la forme $\mathbb{A}_d\vec{u} = \vec{f}$, $d = 1, 2, 3$. Les matrices $(\mathbb{A}_d)_{d=1,2,3}$ étant *symétriques définie-positives*, on utilisera la méthode de Cholesky pour la résolution des systèmes linéaires.

Soit A une matrice de $\mathbb{R}^{N\times N}$ *symétrique définie-positive*, et $\vec{f}$ un élément de $\mathbb{R}^N$. La factorisation de Cholesky de la matrice A construit une matrice triangulaire inférieure L telle que $A = LL^\mathsf{T}$. En remplaçant A par sa factorisation, on peut résoudre le problème

$$A\vec{u} = \vec{f} \tag{3}$$

en deux temps :
- une *descente* : résoudre $L\vec{y} = \vec{f}$;
- une *remontée* : résoudre $L^\mathsf{T}\vec{u} = \vec{y}$.

Ceci est très avantageux, car chacun des deux problèmes fait intervenir une matrice triangulaire, pour laquelle la résolution du système linéaire associé est élémentaire...

Exercice 8 Méthode de Cholesky (I).
Ecrire une fonction Matlab pour calculer la solution de (3) à l'aide la méthode de Cholesky.
Aide : utiliser la fonction Matlab qui réalise la factorisation de Cholesky.

Corrigé 8 Dans le fichier `SolChol.m`, créér la fonction `SolChol` :

```
function u = SolChol(A,f)
% On suppose que la matrice A est SDP

% 1. Calcul de L triangulaire inferieure telle que A = L.L'
LT = chol(A);

% 2. Resolution de L.L'u = f, par une descente-remontee
% 2.1 Descente : resolution de Ly = f
y = LT'\f;
% 2.2 Remontee : resolution de L'u = y
u = LT\y;
```

Exercice 9 Résolution des problèmes discrétisés et étude de l'erreur.
Les problèmes statiques 1d (fil, poutre), 2d (membrane), 3d (cavité) sont respectivement posés dans $\Omega_d =]0, 1[^d$, pour $d = 1, 2, 3$: trouver u tel que

$$-\Delta_d u = f \text{ sur } \Omega_d, \quad u = 0 \text{ sur } \partial\Omega.$$

Ils sont discrétisés à l'aide de la méthode des différences finies dans $\mathbb{R}^N$ ($N = n^d$), pour aboutir à

$$\mathbb{A}_d\vec{u} = \vec{f} \text{ pour } \ d = 1, 2, 3.$$

1. Appliquer la méthode de Cholesky aux problèmes discrétisés.

2. Comment peut-on vérifier que la solution calculée est une bonne approximation de la solution exacte ?

Corrigé 9 1. Dans le fichier `Donnee1d.m`, créer la fonction :

```
function f = Donnee1d(n)
% Second membre aleatoire
f = rand(n,1);
```

Ensuite, pour n donné, on tape en ligne :

```
A1 = Laplaciend1d(n);
f1 = Donnee1d(n);
u1 = SolChol(A1,f1);
```

On procède de même en 2d et 3d...

2. Il faut comparer la solution calculée $\vec{u} = (u_i)_{1\leq i\leq N}$ aux valeurs prises par la solution u aux points de discrétisation, c'est-à-dire $(x_i)_{1\leq i\leq n}$ en 1d, $(x_i, y_j)_{1\leq i,j\leq n}$ en 2d, $(x_i, y_j, z_k)_{1\leq i,j,k\leq n}$ en 3d. D'après l'étude théorique de l'erreur, on sait que ces valeurs sont de plus en plus proches, lorsque n croît, ou ce qui est équivalent, lorsque le pas de discrétisation $h = 1/(n+1)$ decroît.

Pour cela, il faut connaître la solution exacte... Une façon simple de procéder est d'en choisir une (!!), puis de construire le second membre. Il convient de faire attention à ce que la solution retenue respecte bien la condition aux limites sur la frontière. Par exemple, on peut prendre (avec l, m, n entiers non nuls) :

(1d) $u_l(x) = \sin(l\pi x)$, ou $u(x) = x(1-x)v(x)$, avec v quelconque, etc. ;

(2d) $u_{l,m}(x, y) = \sin(l\pi x)\sin(m\pi y)$, ou $u(x) = x(1-x)y(1-y)v(x, y)$, etc. ;

(3d) $u_{l,m,n}(x, y, z) = \sin(l\pi x)\sin(m\pi y)\sin(n\pi z)$, etc. ;

Pour ce qui concerne la comparaison entre solutions exacte et approchée, on peut procéder comme ci-dessous.

- *En visualisant* les solutions à l'aide de `plot` ou `plot3`.
- *En construisant* le vecteur erreur $\vec{e} \in \mathbb{R}^N$ défini par

$$e_I = u_I - u(x_i)\ (1d)\ ;\ e_I = u_I - u(x_i, y_j)\ (2d)\ ;\ e_I = u_I - u(x_i, y_j, z_k)\ (3d).$$

A partir de là on peut calculer la norme de l'erreur, par exemple $\|\vec{e}\|_\infty$, et étudier son évolution en fonction de n.

Exercice 10 Etude de complexité.

1. Comment évaluer la complexité
   - d'un algorithme ?
   - sous Matlab ?
2. Evaluer la complexité de la descente-remontée, puis de la factorisation de Cholesky pour $A \in \mathbb{R}^{N\times N}$. Evaluer la complexité totale de la résolution du système linéaire $A\vec{u} = \vec{f}$ à l'aide de la méthode de Cholesky.
3. Choisir succcessivement $A = \mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3$, en faisant varier N, et comparer la complexité de la factorisation au temps d'exécution de la commande `LT = chol(A)` sous Matlab. Conclusion ?

4. Même question pour la résolution du système linéaire et la commande `A\f` .

Corrigé 10 1. Définition de la complexité
- d'un algorithme : nombre total d'opérations (ici $+, -, *, /$, voire $\sqrt{}$) en fonction de la taille du problème (ici $N = n^d$).
- sous Matlab : temps calcul, mesuré par `tic ; ... ;toc` ou à l'aide de `cputime`.

2. Algorithmes et complexités associées.
(a) Descente $L\vec{y} = \vec{f}$:

pour $i = 1, \ldots, N$ **faire**

$$y_i = [f_i - \sum_{j=1}^{i-1} L_{i,j} y_j]/L_{i,i}.$$

fin

L'algorithme consiste en une boucle unique sur i. Le calcul de la composante y_i requiert :
- pour $i = 1$: 1 division ;
- pour $i > 1$: 1 soustraction, $(i-1)$ multiplications, $(i-2)$ additions, 1 division.

Soit un total de $(2i-1)$ opérations élémentaires $(+, -, *, /)$. La complexité de la descente est donc égale à

$$\sum_{i=1}^{i=N} (2i-1) = 2\frac{N(N+1)}{2} - N = N^2 \text{ opérations.}$$

(b) Remontée $U\vec{u} = \vec{y}$, avec $U = L^{\mathsf{T}}$:

pour $i = N, \ldots, 1$ **(pas** -1**) faire**

$$u_i = [y_i - \sum_{j=i+1}^{N} U_{i,j} u_j]/U_{i,i}.$$

fin

De même que pour la descente, la complexité de la remontée est égale à N^2 opérations.
(c) Factorisation de Cholesky :

pour $i = 1, \ldots N$ **faire**
  pour $j = 1, \ldots i-1$ **faire**

$$L_{i,j} = [A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k}]/L_{j,j}$$

  fin

$$L_{i,i} = [A_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2]^{1/2}.$$

fin

L'algorithme consiste en une boucle imbriquée sur i et j. L'indice i varie de 1 à N. Pour une valeur de i fixée, on effectue tout d'abord une boucle sur j de 1 à $i-1$, plus le calcul pour $j=i$, avec pour i et j fixés $2j-1$ opérations. Soit au total i^2 opérations pour chaque i ; en effet,

$$\sum_{j=1}^{j=i}(2j-1) = 2\frac{i(i+1)}{2} - i = i^2 \text{ opérations.}$$

Si on considère maintenant la boucle sur i, on arrive à la complexité globale :

$$\sum_{i=1}^{i=N} i^2 = \frac{N(N+1)(2N+1)}{6} = \frac{1}{3}N^3 + \frac{1}{2}N^2 + \frac{1}{6}N \text{ opérations.}$$

(d) Complexité totale : on somme, pour trouver le terme dominant de

$$\frac{1}{3}N^3 \text{ opérations.}$$

3. Pour $A = \mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3$, on fait varier n (et donc N), et on étudie le temps calcul en échelle $\log - \log$ (commande Matlab `loglog`). On ne retrouve pas un comportement en $3\log N = 3d\log n$: Matlab est toujours meilleur, et utilise donc un autre algorithme... A titre indicatif, le log du temps calcul sous Matlab se comporte selon
   - pour $d=1$: $\approx 1.0\log N$;
   - pour $d=2$: $\approx 1.4\log N$;
   - pour $d=3$: $\approx 2.0\log N$.
4. *Idem.*

Exercice 11 Méthode de Cholesky (II).

1. Rappeler ce qu'est le profil (optimisé) d'une matrice symétrique définie-positive.
2. Ecrire les algorithmes de descente-remontée et de la factorisation de Cholesky, avec prise en compte du *profil*.
3. Evaluer la complexité de la résolution du système linéaire $\mathbb{A}_d\vec{u} = \vec{f}$ à l'aide de la méthode de Cholesky avec prise en compte du profil, pour les matrices $(\mathbb{A}_d)_{d=1,2,3}$. Conclusion ?

Corrigé 11 1. Le profil d'une matrice inversible $A \in \mathbb{R}^{N\times N}$ est déterminé par la donnée d'un vecteur $p^A \in \mathbb{R}^N$ tel que, pour $1 \le i \le N$:

$$A_{i,(p^A)_i} \neq 0 \text{ et } A_{i,j} = 0 \text{ pour } 1 \le j \le (p^A)_i - 1.$$

Pour une matrice définie-positive, on remarque que $A_{i,i} = (A\vec{e}_i, \vec{e}_i) > 0$, et ainsi $(p^A)_i \le i$ pour tout i. Et, si on définit le *profil optimisé* d'une matrice SDP comme étant :

$$P^A_{opt} = \left\{(i,j) \in \{1,\dots,N\}^2 \ : \ (p^A)_i \le j \le i\right\},$$

son intérêt principal est d'être conservé par la factorisation de Cholesky ! En d'autres termes, on a la propriété : $P^L_{opt} = P^A_{opt}$.

2. Algorithmes avec prise en compte du profil (noté p) :

(a) Descente $L\vec{y} = \vec{f}$:

pour $i = 1, \ldots, N$ **faire**

$$y_i = [f_i - \sum_{j=p_i}^{i-1} L_{i,j} y_j]/L_{i,i}.$$

fin

(b) Remontée $L^{\mathsf{T}}\vec{u} = \vec{y}$:

pour $i = N, \ldots, 1$ **(pas** -1**) faire**

$$u_i = [y_i - \sum_{j=i+1}^{q_i} L_{j,i} u_j]/L_{i,i}.$$

fin

Ci-dessus, le profil additionnel $q \in \mathbb{R}^N$ est défini, pour $1 \leq i \leq N$, par :

$$L_{q_i,i} \neq 0 \text{ et } L_{j,i} = 0 \text{ pour } q_i + 1 \leq j \leq N.$$

(c) Factorisation de Choleski :

pour $i = 1, \ldots N$ **faire**

pour $j = p_i, \ldots i-1$ **faire**

$$L_{i,j} = [A_{i,j} - \sum_{k=\max(p_i,p_j)}^{j-1} L_{i,k} L_{j,k}]/L_{j,j}$$

fin

$$L_{i,i} = [A_{i,i} - \sum_{k=p_i}^{i-1} L_{i,k}^2]^{1/2}.$$

fin

3. Complexité avec stockage profil.

(a) Bornes *a priori*.
Notons que, pour $d = 1, 2, 3$, on a respectivement, pour tout i variant de 1 à N,
– pour $d = 1$: $i - p_i \leq 1$;
– pour $d = 2$: $i - p_i \leq n = N^{1/2}$;
– pour $d = 3$: $i - p_i \leq n^2 = N^{2/3}$.
De façon plus compacte, si on pose $\bar{p}_d = n^{d-1} = N^{\frac{d-1}{d}}$, on peut écrire, pour $d = 1, 2, 3$ et pour tout i variant de 1 à N,

$$(i - p_i) \leq \bar{p}_d.$$

NB. Pour le profil additionnel (étape de remontée), on a également $(q_i - i) \leq \bar{p}_d$.

(b) Descente-remontée.
L'algorithme de descente consiste encore en une boucle unique sur i. Le calcul de la composante y_i requiert cette fois :

- pour $i = 1$: 1 division ;
- pour $i > 1$: 1 soustraction, $(i - p_i)$ multiplications, $(i - p_i - 1)$ additions, 1 division.

Soit un total de $2(i - p_i) + 1$ opérations élémentaires $(+, -, *, /)$. La complexité de la descente est donc inférieure à

$$\sum_{i=1}^{i=N} (2\bar{p}_d + 1) \leq (2\bar{p}_d + 1)N \text{ opérations.}$$

Comme le coût de la remontée est identique à celui de la descente ($(q_i - i) \leq \bar{p}_d$, pour tout i), on arrive à un coût global de descente-remontée de l'ordre de :
– pour $d = 1 : 6N$ opérations ;
– pour $d = 2 : 4N^{3/2}$ opérations ;
– pour $d = 3 : 4N^{5/3}$ opérations.

(c) Factorisation de Choleski.
Avec un stockage profil, on exécute encore une boucle imbriquée sur i et j. L'indice i varie toujours de 1 à N, mais, l'indice j varie seulement de p_i à i. De plus, pour i et j fixés, on effectue $2(j - \max(p_i, p_j)) + 1$ opérations, ce qui est toujours inférieur à

$$2(i - p_i) + 1 \text{ opérations.}$$

Ainsi, le nombre total d'opérations est inférieur à :

$$\sum_{i=1}^{i=N} \sum_{j=p_i}^{j=i} (2\bar{p}_d + 1) \leq \sum_{i=1}^{i=N} (2\bar{p}_d + 1)(1 + \bar{p}_d) = (2\bar{p}_d + 1)(1 + \bar{p}_d)N \text{ opérations.}$$

NB. Pour $d = 1$, on peut préciser l'estimation en distinguant pour i fixé le calcul des coefficients diagonaux de celui des coefficients extra-diagonaux, au nombre de *un* pour chaque catégorie : $L_{i,i}$ et $L_{i,i-1}$ (sauf pour $i = 1$). On se rend facilement compte qu'il faut bien *trois* opérations pour les premiers, mais seulement *une* pour les seconds, soit au total $(3 + 1)N = 4N$ opérations.
En conclusion, pour réaliser la factorisation de Cholesky avec un stockage profil, on arrive à un coût global de l'ordre de
– pour $d = 1 : 4N$ opérations ;
– pour $d = 2 : 2N^2$ opérations ;
– pour $d = 3 : 2N^{7/3}$ opérations.

(d) Résolution de $\mathbb{A}_d u = f$: la partie prépondérante de la résolution étant la factorisation (sauf en 1d), on retrouve
– pour $d = 1 : 10N$ opérations ;
– pour $d = 2 : 2N^2$ opérations ;
– pour $d = 3 : 2N^{7/3}$ opérations.

Sous Matlab, on a bien un comportement linéaire en 1d... En 2d et 3d, les comportements restent meilleurs que respectivement $2.0 \log N$ et que $2.3 \log N$.