TP Matlab 2

Décompositions

2.1 Systèmes linéaires ?

Les méthodes suivantes servent à résoudre des systèmes linéaires. Les systèmes linéaires sont, en pratique, des systèmes entrées (x) et sortie (b). Les sorties sont, en pratique, ce que l'on cherche à avoir. La question qui se pose est donc quelle est l'entrée à choisir ?

La première méthode, dite de Cholesky, vise originellement à résoudre des systèmes de mécanique. Cholesky était un militaire et polytechnicien. Il a développé la méthode pour améliorer la précision des tirs d'obus en fonction de la topographie du terrain.

La méthode LU n'est, finalement, qu'une généralisation de la décomposition de Cholesky pour des matrices quelconques.

Ces méthodes ont pour gros avantages de permettre de pré calculer des matrices qui permettront ensuite, pour un coup réduit (en terme de flops), de résoudre plusieurs fois un même système, pour des sorties différentes.

2.2 Algorithmes à implémenter

2.2.1 Décomposition de Choleski

Ouais, aujourd'hui, j'ai fait une décomposition de Cholesky.

On part d'une matrice symétrique définie positive A de dimension $N \times N$. C'est à dire que $A^t = A$, que toutes ses valeurs propres sont strictement positives et (donc) qu'elle est inversible.

On peut toujours trouver une matrice L triangulaire supérieur tel que

$$A = LL^t$$

On a donc :

$$\begin{aligned} a_{ij} &= (LL^t)_i\, j \\ &= \sum_{k=1}^{N} l_{ik} l_{jk} \\ &= \sum_{k=1}^{\min(i,j)} l_{ik} l_{jk} \end{aligned}$$

Ce qui se réduit à, avec $1 \leq i \leq j \leq N$:

$$a_{ij} = \sum_{k=1}^{i} l_{ik} l_{jk}$$

Pour identifier L, on procède par récurrence comme suit :

Récurrence
La $i$ème colonne s'obtient :

$$a_{ii} = \sum_{j=1}^{i} l_{ij}^2.$$

On a donc

$$l_{ii} = \sqrt{a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2} \Rightarrow \left\{ \begin{array}{lcl} a_{ij} & = & \sum_{k=1}^{i} l_{ik} l_{jk} \\ & = & l_{ii} l_{ji} + \sum_{k=1}^{i-1} l_{ik} l_{jk} \end{array} \right. .$$

Initialisation
pour $i = 1$, on a

$$l_{11} = \sqrt{a_{11}}$$

et

$$l_{j1} = \frac{a_{1j}}{l_{11}}.$$

D'où

$$l_{ji} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}}{l_{ii}}$$

2.2.2 Questions

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

1) A est elle symétrique définie positive ?
2) Implémenter l'algorithme de Cholesky.

2.2.3 Décomposition LU

Je suis le prince des décompositions.

Gauss-like

Cet algorithme est proche de celui du pivot de Gauss. L'idée est de trouver les coefficients de U et L directement à partir de ceux de A.

Cela fonctionne bien à condition de procéder dans l'ordre : 1ère ligne, puis 1ère colonne, puis 2ème ligne, puis 2ème colonne, etc.

```
N= taille de A
Initialiser U
Initialiser L
Pour i allant de 1 à N, faire
   Pour j allant de i à N, faire
```

$u_{ii} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$

```
   Fin j
   Pour j allant de i+1 à N, faire
```

$l_i i = \frac{a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki}}{u_{ii}}$

```
   Fin j
```

$l_{ii} = 1$

```
Fin i
```

Crout alors.

L'autre algorithme, de Crout, est plus subtil. Il se base sur une récurrence de multiplication de matrice triangulaire.

On définit (étape 0) :

$$A^{(0)} = A.$$

Ensuite, l'étape n : sur la $n$ième colonne de $A^{(n-1)}$, on élimine les éléments sous la diagonale en ajoutant à la $i$ème ligne, la $i$ème ligne multipliée par :

$$l_{in} = -\frac{a_{in}^{(n-1)}}{a_{nn}^{(n-1)}},$$

avec $i \in \{n+1, \ldots, N\}$ Ce qui revient à multiplier à gauche $A^{(n-1)}$ par L_n,

$$\mathrm{L}_n = \begin{pmatrix} 1 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & \vdots \\ \vdots & & \vdots & l_{i,n} & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \ldots & 0 & l_{N,n} & 0 & 0 & 1 \end{pmatrix}.$$

Et donc $A^{(n)} = L_n \times A^{(n-1)}$ Après $N-1$ itérations, tous les éléments sous diagonaux ont été éliminés, et donc $A^{(N-1)}$ est triangulaire supérieure. Or,

$$A = L_1^{-1} L_1 A^{(0)} = L_1^{-1} A^{(1)} = L_1^{-1} L_2^{-1} L_2 A^{(2)} = \ldots = L_1^{-1} L_2^{-1} \ldots L_{N-1}^{-1} A^{(N-1)}$$

On note $U := A^{(N-1)}$ et $L := L_1^{-1} L_2^{-1} \ldots L_{N-1}^{-1}$. U est bien triangulaire supérieure et L triangulaire inférieure (une multiplication de matrice triangulaire inférieure restant triangulaire inférieure).

2.2.4 Questions

1) Implémenter les deux algorithmes.
2) Le second algorithme peut ne pas converger. Dans quel cas ?
3) Trouver une solution.