Informatique tronc commun – TP nº 13
Décomposition LU

Mercredi 30 mars 2016

1. **Lisez attentivement tout l'énoncé avant de commencer.**
2. Commencez la séance en créant un dossier au nom du TP dans le répertoire dédié à l'informatique de votre compte.
3. Vous ne rendrez pas de compte rendu : l'objectif est d'écrire *vous-même* le plus de programmes (fonctionnels !) possible.
4. Ayez toujours un crayon et un papier sous la main. Quand vous réfléchissez à une question, utilisez les !
5. Vous devez être autonome. Ainsi, avant de poser une question à l'enseignant, merci de commencer par :
   — relire l'énoncé du TP (beaucoup de réponses se trouvent dedans) ;
   — relire les passages du cours [1] relatifs à votre problème ;
   — effectuer une recherche dans l'aide disponible sur votre ordinateur (ou sur internet) concernant votre question.

   Il est alors raisonnable d'appeler votre enseignant pour lui demander des explications ou une confirmation !

On s'intéresse dans ce TP à la mise en œuvre de différentes méthodes de résolution de systèmes linéaires, ainsi qu'à l'étude de leur efficacité.

Vous pourrez trouver sur le site de la classe un document de test, intitulé `test_tp13.py`. Si vous l'enregistrez dans le même dossier que votre script (nommé par défaut `prog.py`), vous pourrez lancer des tests automatiques par le programme `py.test-3.4` (exemple : lancer la commande `py.test-3.4 > resultat.txt`).

1 Introduction

Lisez cette partie *chez vous* avant de commencer le TP.

1. Dans le cas fort improbable où vous ne vous en souviendriez pas.

1.1 Décomposition LU

Soit A une matrice carrée telle que toute sous-matrice principale (sous-matrice carrée calée dans le coin supérieur gauche) soit inversible.

On rappelle que :

— l'opération $L_i \leftarrow L_i + \alpha L_j$ effectuée sur les lignes d'une matrice A, revient à effectuer le produit $T(i,j,\alpha) \times A$, où $T(i,j,\alpha)$ est la matrice de transvection égale à l'identité, à l'exception du coefficient (i,j) qui vaut α ;

— l'opération $C_i \leftarrow C_i + \alpha C_j$ effectuée sur les colonnes d'une matrice A, revient à effectuer le produit $A \times T(j,i,\alpha)$.

On peut alors montrer que la matrice A admet une décomposition $A = LU$, où L est une matrice triangulaire inférieure et U une matrice triangulaire supérieure. La matrice L est obtenue en appliquant à la matrice identité les opérations sur les colonnes correspondant aux inverses des matrices d'opérations sur les lignes permettant de trianguler A par la méthode de Gauss (sans échange de ligne : toute sous-matrice de A étant principale, on peut montrer que si la matrice A est rendue triangulaire pour ses k premières colonnes, alors le coefficient diagonal de la colonne suivante n'est pas nul et peut être choisi comme pivot).

Plus précisément : si T_1, ... , T_p sont les p transvections successives à appliquer à A pour la rendre triangulaire supérieure, nous obtenons : $(T_p \times \ldots \times T_1) \times A = U$. Il faut alors poser $L = (T_p \times \ldots \times T_1)^{-1} = I_n \times T_1^{-1} \times \ldots \times T_p^{-1}$.

L'intérêt réside dans le fait qu'on ramène la résolution de $AX = B$ à la résolution de 2 systèmes triangulaires. Ceci s'avère notamment intéressant lorsque l'on a plusieurs systèmes à résoudre associés à la même matrice A.

1.2 Matrices dans Python avec `numpy`

Nous allons écrire un programme permettant de calculer cette décomposition LU, et le comparer à l'algorithme du pivot de Gauss classique.
Pour cela, nous écrirons les matrices sous forme de tableaux bi-dimensionnels de type `array`, en utilisant le module `numpy`, dont voici quelques exemples d'utilisations :

```
>>> import numpy as np
>>> A=np.array([[1, 2], [3, 4]])
>>> A
array([[1, 2],
       [3, 4]])
>>> B=np.eye(2)
>>> B[1,0]=2
>>> B
array([[ 1.,  0.],
       [ 2.,  1.]])
>>> C = A.dot(A) + 2*B.dot(A)
>>> C
array([[  9.,  14.],
       [ 25.,  38.]])
```

On a bien calculé $C = A^2 + 2BA$, avec $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ et $B = T(1, 0, 2)$.

Nous utiliserons aussi les fonctions définies dans le fichier `random_matrix`, disponible sur le site de la classe, ainsi que la fonction `clock` du module `time` pour chronométrer les temps d'éxécution.

2 Travail demandé

Q1 Écrire une fonction `trans_ligne` prenant en entrée un quadruplet `(A,i,j,alpha)`, où A est une matrice d'ordre n, $i, j \in [\![0, n[\![$, et $\alpha \in \mathbb{R}$, et qui multiplie A à gauche par la matrice $T(i, j, \alpha)$, c'est-à-dire qui applique à A la transvection $L_i \leftarrow L_i + \alpha L_j$. Ainsi cette fonction modifie A et ne renvoie rien.
Attention : par souci d'efficacité, il est préférable de voir cette opération comme une opération entre des lignes de A, plutôt que comme un produit matriciel. On s'arrangera autant que possible pour éviter les calculs inutiles sur les coefficients qu'on sait être nuls.

Q2 Écrire une fonction `trans_colonne` prenant en entrée un quadruplet `(A,i,j,alpha)`, où A est une matrice d'ordre n, $i, j \in [\![0, n[\![$, et $\alpha \in \mathbb{R}$, et qui multiplie A à droite par **l'inverse** de la matrice $T(i, j, \alpha)$, c'est-à-dire qui applique à A la transvection $C_j \leftarrow C_j + \frac{1}{\alpha} C_i$.
Les remarques de la question précédente sont toujours d'actualité.

Q3 Écrire une fonction `LU` retournant les deux matrices L et U de la décomposition ci-dessus, pour une matrice A donnée.

Q4 Écrire une fonction `resolution_sup` de résolution d'un système triangulaire supérieur.

Q5 Écrire une fonction `resolution_inf` de résolution d'un système triangulaire inférieur.

Q6 Soient $A \in \mathscr{M}_{100}(\mathbb{R})$ et $b_0, \ldots, b_{24} \in \mathscr{M}_{100,1}(\mathbb{R})$, choisies aléatoirement (les coefficients étant pris uniformément dans $[\![0, 100]\!]$).
Calculer :

1. le temps de résolution de 25 systèmes associés à A, en calculant une fois pour toutes la décomposition LU de A, puis en résolvant 2 systèmes triangulaires pour chaque second membre B ;
2. le temps de résolution de 25 systèmes associés à A, en reprenant un pivot complet pour chaque système ;
3. le temps de résolution de 25 systèmes associés à A, en calculant A^{-1} par la méthode du pivot, une fois pour toutes puis en calculant $A^{-1}B$ pour chaque second membre.

Q7 Plus généralement, tracer, pour $k \in [\![1, 25]\!]$, les 2 courbes correspondant au temps de réponse pour la résolution de k systèmes associés à $A \in \mathscr{M}_{100}(\mathbb{R})$, par décomposition LU, et par calcul de A^{-1}. Commenter.