Maîtrise de Mathématiques Fondamentales et Applications
Grands Systèmes linéaires
feuille de TP 3

24 Septembre 2009

Exercice - 1 *Méthode de résolution par la factorisation LU*

Algorithme de résolution de systèmes triangulaires inférieures (à gauche) et supérieures (à droite)

```
-----------------------------------------------------------------------------------------
     ALGORITHME DE DESCENTE                  |        ALGORITHME DE REMONTEE
-----------------------------------------------------------------------------------------
Données: A,b. Résultat:x solution de A x= b| Données: A,b. Résultat:x solution de A x=b
-------------------------------------------|---------------------------------------------
pour i= 1 à n                              | pour i= n à 1
     s= 0                                  |      s = 0
     pour j = 1 à i-1                      |        pour j = i+1 à n
       s = s + A(i,j) * x(j)               |             s = s + A(i,j) * x(j)
     fin j                                 |        fin
  x(i) = (b(i) - s) /A(i,i)                |    x(i) = (b(i) - s)/A(i,i)
fin                                        |fin
```

1- Ecrire une une fonction MATLAB de prototype `function [x] = Descente (A, b)`, qui résoud un système linéaire triangulaire inférieure $Ax = b$.
Tester la fonction en générant aléatoirement des matrices triangulaires inférieures inversibles (voir TP1), de tailles n et en effectuant : `b = rand(n,1), norm(b - Descente(A,A*b))`.

2- Même question pour les matrices triangulaires supérieures inversibles. La fonction aurra pour prototype, `function [x] = Remontee (A, b)`.

3- Ecrire une fonction MATLAB de prototype `function [L,U] = decompLU (A)` qui effectue la décomposition LU d'une matrice carrée.

Algorithme de décomposition A = LU

```
Donnée: A.                                 |
Résultat: U triangulaire supérieure        |     Explications des notations et indications
L triangulaire inférieure (diagonale unité)|
-------------------------------------------| ---------------------------------------------
pour k = 1 à n                             |% : introduit un commentaire
  pour i = k+1 à n                         |
          A(i,k)= A(i,k)/ A(k,k)           |
          pour j = k+1 à n                 |
             A(i,j)=A(i,j)-A(i,k)*A(k,j)   |
          fin                              |
  fin                                      |
fin                                        | Commande MATLAB pour récupérer L et U
Récupérer L et U  à partir de A            | U=triu(A); L = A - U + diag(ones(n,1));
```

a- Evaluer cette fonction sur la matrice $A = \begin{bmatrix} 2 & 4 & -4 & 1 \\ 3 & 6 & 1 & -2 \\ -1 & 1 & 2 & 3 \\ 1 & 1 & 4 & 1 \end{bmatrix}$.
Conclure en calculant tous les mineurs principaux d'odre $k, k = 1 \ldots 3$ de A.

b- On considère la matrice de permutation $P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. .
Verifier que le produit PA admet une factorisation LU, en évaluant `[L,U] = decompLU(PA)`.

Expliquer alors comment résoudre un système linéaire dont la matrice est la matrice A ci-dessus.

4- La question précédente met en évidence la nécessité de pivot dans la factorisation LU de certaines matrices.

a- Montrer que pour toute matrice inversible $A \in \mathcal{M}_m(\mathbb{R})$, il exite une matrice de permutation P telle que la matrice PA admette une factorisation LU. Comparée à la factorisation LU, on a allégé les hypothèses sur la matrice A pour que la factorisation $PA = LU$ soit possible. Lesquelles ?

b- Ecrire une fonction MATLAB de prototype `[L,U,P] = decompLUP(A)`, qui effectue une factorisation LU d'une matrice régulière A selon une méthode de Gauss avec pivot partiel, c'est à dire avec une permutation de lignes. (*L'algorithme est donné ci-dessous*).

Algorithme de décomposition PA =LU

```
Donnée: A. Résultats: A contenant L,et U   |     Explications des notations et indications
et P vecteur de permutation des lignes     |
-------------------------------------------| ---------------------------------------------
pour i =1 à n P(i) = i fin                 |% : introduit un commentaire
pour k = 1 à n                             |ipivot est telle que |A(ipivot,k)|=max|A(i,k)| k<= i <= n
  recherche la ligne ipivot du pivot:      | commande MATLAB de recherche du pivot :
 permuter les lignes P(ipivot) et P(k)     |[vpivot,ipivot]=max(abs(A(k:n,k))); ipivot = k - 1 + ipivot;
pour i = k+1 à n                           | commande MATLAB pour permuter les lignes P(ipivot) et P(k) :
           A(i,k)= A(i,k)/AA(k,k)          |ip=P(ipivot); P(ipivot)=P(k);P(k)= ip;
          pour j = k+1 à n                 |v=A(k,:); A(k,:) = A(ipivot,:); A(ipivot,:)= v;
          A(i,j) = A(i,j)-A(i,k) * A(k,j)  |
          fin                              |
    fin                                    |
```

Exercice - 2 *Complexité de la méthode de décomposition LU*

1- Complexité de la factorisation.
a- Modifer la fonction `[L,U] = decompLU(A)` de l'exercice précédent, en une fonction `[L,U,nop] = decompLU_nop(A)` qui retourne en plus le nombre de multiplications et de divisions effectuées pendant la factorisation LU.
(*On Pourra initialiser nop à 0 en début de fonction et l'incrémenter, dans les boucles de la factorisation LU, à chaque fois qu'on rencontre une multiplication ou une division (algo-lignes 7 et 9)*).

b- Justifier l'existence de la factorisation LU d'une matrice symétrique définie positive. Générer des matrices symétriques définies positives de tailles $n = 1, \cdots 30$. Relever $Nop(n)$ fourni par la factorisation `decompLU_nop(A)` et représenter sur un même graphique la courbe $n \longmapsto Nop(n)$ et la courbe $n \longmapsto n^3/3$. Qu'observe-t-on ?
(*On pourra utiliser le script* MATLAB *ci-dessous qui réalise cette expérience*).

fichier nopLU.m de test de la complexité de la factorisation LU

```
x=[]; ni=[]; Nop=[];ind=0;
for i =5:20
  ind=ind+1; x(ind)= i;
  A=MatSdp(i);            % voir TP1
  [L,U,Nop(ind)] = decompLU_nop(A);
  ni(ind) = (i^3)/3. ;
end
plot(x,Nop,'*',x,ni,'-'); legend('Nop LU','n^3 / 3');
```

c- Reprendre la même expérience en ne représentant que la courbe $n \longmapsto Nop(n)$ en échelle logarithmique. Déterminer la pente de la droite la plus proche de cette courbe et conclure.
(*On utilisera la commande* MATLAB `loglog` *pour la représentation en échelle logarithmique).*

Exercice - 3 *Phénomène de remplissage*

1- Modifier la factorisation LU de sorte qu'elle retourne les facteurs L, U dans une matrice de la taille de A ; U étant stockée dans la partie supérieure et L (sauf sa diagonale) dans la partie inférieure. Le prototype de la fonction sera `function [A]=decompLU_eco(A)`.

2- On considère la matrice de taille n définie sous MATLAB , pour n donnée, par :
`A = eye(n) ; A( :,1)=1 ; A(1, :)=1 ; A(1,1)=n ;`

a- Pour diverses valeurs de n, afficher la matrice A (commande `spy`). Afficher aussi la matrice `B=decompLU_eco(A)`. Qu'observe-t-on ?

b- Que retourne la commande MATLAB `length(find(A))` ?
Comparer `length(find(A))` et `length(find(decompLU_eco(A)))`. Conclure.