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

24 Septembre 2009

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

1- Il suffit de supprimer la sortie nop du code ci-dessous.

Algorithme de descente : fichier Descente.m

```
function [x,nop] = Descente (L,b)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function qui  resoud le systme L x = b
% avec vérification de la compatibilité des dimensions
%
% Entree : L matrice pleine triangulaire inférieure
%          b vecteur second membre
% Sortie : x vecteur solution
%          nop nombre d'operations
%
% GRANDS SYSTEMES LINEAIRES
%  M1 MFA      2009-2010
% J.-B. APOUNG K. Labo Math AN-EDP Univ d'ORSAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  if(m~=n || n ~= length(b) )
%  error('données non compatibles');
%  end
%  if(prod(diag(L))==0)
%  error(' matrice non inversible ');
%  end
nop=0;
[m,n] = size(L);
x = zeros(n,1);
for k = 1:n
   x(k) = b(k) - L(k,1:k-1) * x(1:k-1,1); nop = nop + k-1;
   x(k) = x(k) / L(k,k); nop = nop + 1;
end
```

2- Il suffit de supprimer la sortie nop du code ci-dessous.

Algorithme de remontée : fichier Remontee.m

```
function [x,nop] = Remontee (U,b)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function qui  le systme U x = b
% avec vérification de la compatibilité des dimensions
% % Entree : U matrice pleine triangulaire inférieure
%           b vecteur second membre
% sortie   : x vecteur solution
%            nop nombre d'operations
%
% GRANDS SYSTEMES LINEAIRES
% M1 MFA          2009-2010
% J.-B. APOUNG K. Lab Math AN-EDP Univ d'ORSAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nop=0;
[m,n] = size(U);
x = zeros(n,1);
for k = n:-1:1
 x(k) = b(k) - U(k,k+1:n)*x(k+1:n,1); nop=nop+n-k;
 x(k) = x(k)/U(k,k);nop=nop+1;
end
```

3- Il suffit de supprimer la sortie nop du code ci-dessous.

Algorithme de décomposition LU : decompLU.m

```
function [L,U,nop] = decompLU (A)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function qui effectue la décomposition LU
%
% Entree : A matrice régulière de mineurs principaus non nuls
% Sortie : L matrice triangulaire inferieure à diagonale unité
%          U matrice triangulaire supérieure
%          nop nombre d'opérations
% GRANDS SYSTEMES LINEAIRES
% M1 MFA          2009-2010
% J.-B. APOUNG K. Lab Math AN-EDP Univ d'ORSAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[m,n] = size(A);
if(m ~= n) error('matrice non carrée');end
% on ne veut pas détruire la matrice on en fait une copie
AA = A;
nop = 0;

for k=1:n
   if(AA(k,k) ==0) error('pivot nul'); end
   for i = k+1:n
       AA(i,k)=AA(i,k)/AA(k,k); nop=nop+1;
       for j = k+1:n
           AA(i,j) = AA(i,j) - AA(i,k) * AA(k,j); nop = nop+1;
       end
   end
end
U=triu(AA); L = AA - U + diag(ones(n,1));
```

a- La factorisation LU échoue sur la matrice `A=[2,4,-4,1 ; 3,6,1,-2 ; -1,1,2,3 ; 1,1,-4, 1]`.
On évalue les mineurs principaux de A : `det(A(1 :1,1 :1)), det(A(1 :2,1 :2)), det(A(1 :3,1 :3))`.
On constate que l'un des mineurs de A (en particulier `det(A(1 :2,1 :2))`) est nul. A n'admet alors pas de décomposition LU.
b- Avec `P=[0,1,0,0 ; 0,0,1,0 ; 1,0,0,0 ; 0,0,-0, 1]`, on constate que la factorisation LU de PA est possible, on détermine `[L,U]=decompLU(PA)`. Ainsi pour résoudre $Ax = b$, on ecrit $PA = Pb$ soit $LU = Pb$. d'où la résolution `y=Descente(Pb) ; x=Remontee(y)`.
4- a- Admettre le résultat. Pour la démonstration, voir cours.
(A = LU possible) $\Longleftrightarrow$ (tous les n-1 mineurs principaux de A sont non nuls).
(PA= LU possible) $\Longleftrightarrow$ (A inversible).

b- Il suffit de supprimer la sortie nop du code ci-dessous.

Algorithme de décomposition PA=LU : decompLUP.m

```
function [L,U,P,nop] = decompLUP ( A )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function qui effectue la décomposition LU
%
% Entree : A matrice régulière
% Sortie : L matrice triangulaire inferieure à diagonale unité
%          U matrice triangulaire supérieure
%          P vecteur de la permutation
%          nop nombre d'opérations
% GRANDS SYSTEMES LINEAIRES
% M1 MFA          2009-2010
% J.-B. APOUNG K. Lab Math AN-EDP Univ d'ORSAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,n] = size(A);
if(m ~= n) error('matrice non carrée');end
% on ne veut pas détruire la matrice on en fait une copie
AA = A;
nop = 0;
P=1:n;

for k=1:n
%recherche du pivot par ligne
[vpivot,ipivot] = max(abs(AA(k:n,k)));
ipivot=k-1+ ipivot;
%permutation des lignes
ip=P(ipivot); P(ipivot) = P(k);       P(k) = ip;
   v=AA(k,:); AA(k,:) = AA(ipivot,:);   AA(ipivot,:)= v;
%factorisation
   for i = k+1:n
       AA(i,k)=AA(i,k)/AA(k,k); nop=nop+1;
       for j = k+1:n
           AA(i,j) = AA(i,j) - AA(i,k) * AA(k,j); nop = nop+1;
       end
   end
end
```

Exercice - 2 *Complexité de la factorisation LU*

1- a-

Décomposition LU avec stockage du nombre d'opération : decompLUP_nop.m

```
function [L,U,nop] = decompLU_nop(A)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function qui effectue la décomposition LU
%
% Entree : A matrice régulière de mineurs principaux non nuls
% Sortie : L matrice triangulaire inferieure à diagonale unité
%          U matrice triangulaire supérieure
%          nop nombre d'opérations
% GRANDS SYSTEMES LINEAIRES
% M1 MFA          2009-2010
% J.-B. APOUNG K. Lab Math AN-EDP Univ d'ORSAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[m,n] = size(A);
if(m ~= n) error('matrice non carrée');end
% on ne veut pas détruire la matrice on en fait une copie
AA = A;
nop = 0;

for k=1:n
   if(AA(k,k) ==0) error('pivot nul'); end
   for i = k+1:n
       AA(i,k)=AA(i,k)/AA(k,k); nop=nop+1;
       for j = k+1:n
           AA(i,j) = AA(i,j) - AA(i,k) * AA(k,j); nop = nop+1;
       end
   end
end
U=triu(AA); L = AA - U + diag(ones(n,1));
```

b- Toute sous-matrice principale d'une matrice symétrique définie positive est une matrice symétrique définie positive; elle est donc régulière. Par conséquent la décomposition LU de toute matrice symétrique définie positive est possible.

En exécutant le script ci-dessous (sous MATLAB taper : `nopLU`). On constate que les deux courbes ($Nop(n)$ et $n^3/3$) sont superposées lorsque n est grand. On conclut que la complexité asymptotique de l'algorithme de la décomposition LU est $n^3/3$.

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');
%log(x),log(Nop)
%plot(x,Nop,"*;Nop LU;",x,ni,"-;n^3 / 3;");
```

c- On remplace dans le script précédent `plot` par `loglog`. On obtient une droite de pente 3 qui intersecte (calcul à faire à la main en donnant l'équation cartésienne de la droite de pente 3 passant par le point le plus à doite.) l'axe des ordonnées en $-1.1010 \approx -log(3)$. On tire les mêmes conclusions.

Exercice - 3 *Phénomène de remplissage dans la factorisation LU*

1-

Algorithme de décomposition LU : decompLU_eco.m

```
function [A] = decompLU_eco (A)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function qui effectue la décomposition LU
%
% Entree : A matrice régulière de mineurs principaus non nuls
% Sortie : A contenant U dans sa partie trianglaire supérieure et
%          L(sauf sa diagonale) dans sa partie triangulaire inferieure
%
% GRANDS SYSTEMES LINEAIRES
% M1 MFA          2009-2010
% J.-B. APOUNG K. Lab Math AN-EDP Univ d'ORSAY
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,n] = size(A);
if(m ~= n) error('matrice non carrée');end
for k=1:n
   if(A(k,k) ==0) error('pivot nul'); end
   for i = k+1:n
       A(i,k)=A(i,k)/A(k,k);
       for j = k+1:n
           A(i,j) = A(i,j) - A(i,k) * A(k,j);
       end
   end
end
```

2- **a-** La commande (spy(A)) dessine la matrice A en mettant des croix sur les positions des éléments non nuls. On constate que la matrice obtenue avec la factorisation LU a beaucoup plus d'éléments non nuls (elle est presque pleine).

b- La commande `length(find(A))` retourne ne nombre d'éléments non nuls de la matrice A. On peut utiliser le script suivant

test du remplissage de la factorisation LU

```
x=[]; y=[];z=[];
for n=1:10
  A=eye(n);A(1,:)=1;A(:,1)=1;A(1,1)=n;
  B=decompLU_eco(A);spy(B);pause();
  x(n)=n;
  y(n)=length(find(A));
  z(n)=length(find(B));
end
plot(x,y,"*-",x,z,"+-");
legend('remplissage de A','remplissage LU A');
```

Quelques notes

1- Codes de génération des matrices particulières pour ce TP :

Génération d'une matrice symétrique définie positive

```
function [A] = MatSdp(n)
A = MatSymetrique(n);
[P,D]=eig(A); D = abs(D);
A = P * (D + norm(D)*eye(size(D)))*inv(P);
```

Génération d'une matrice triangulaire inférieure régulière

```
function [A] = MatReguliereI(n)
B = tril(rand(n,n));
A = B + norm(B,'inf')*eye(n);
```

Génération d'une matrice triangulaire Supérieure régulière

```
function [A] = MatReguliereS(n)
B = triu(rand(n,n));
A = B + norm(B,'inf')*eye(n);
```