ECOLE NATIONALE
SUPÉRIEURE
D'INFORMATIQUE

1/8/2016

TP ANALYSE NUMERIQUE

Résolution des systèmes linéaires par les méthodes de LU et Cholesky

AMRINE MOUSSAB AMINE
BELHAMRA YAHIA
G06
ANNÉE: 2015/2016

Sommaire:

1)Introduction :

Les systèmes d'équations algébriques jouent un rôle très important en ingénierie. On peut classer ces systèmes en deux grandes familles :

- Les systèmes linéaires ;
- Les systèmes non linéaires.

Dans ce TP on parler sur les progrès de l'informatique et de l'analyse numérique permettent d'aborder des problèmes de taille prodigieuse. On résout couramment aujourd'hui des systèmes de plusieurs centaines de milliers d'inconnues. On rencontre ces applications en mécanique de fluides, dans k »analyse de structures complexes. On peut par exemple calculer l'écoulement de l'air autour d'un avion ou l'écoulement de l'eau dans une turbine hydraulique complété. On peut aussi analyser la résistance de la carlingue d'un avion à différentes contraintes extérieures et en vérifier numériquement la solidité.

Ces calculs complexes requièrent des méthodes sophistiquées, dans ce TP nous allons aborder les principales méthodes de résolution des systèmes linéaire, à savoir la méthode de la **décomposition LU** et **CHOLESKY**.

2) Rappels des notions et des résultats théoriques :

a) Méthode LU :

Principe de la méthode:

Supposons un instant que nous ayons réussi a éxprimer la matrice A en un produit de deux matrices triangulaires L et U. la questions qu'on peut la posé est :

Comment cela nous permet-il de résoudre le système Ax = b ?

Enfaite il suffit de remarquer que :

$$A\vec{x} = LU\vec{x} = \vec{b}$$

Et de déposer Ux = y. La résolution de systéme linéaire se fait alors en deux étapes :

$$\begin{aligned} L\vec{y} &= \vec{b} \\ U\vec{x} &= \vec{y} \end{aligned}$$

Qui sont deux systemes triangulaires. On utilise d'abord une descente triangulaire sur la matrice L pour obtenir « y » et par la suite une remonte triangulaire sur la matrice U pour obtenir la solution recharchée « x ».

Methode de la factorisation LU :

Dans le cas ou le $\det(A) \neq 0$, la méthode de Guass est applicable et on a MA est triangulaire inférieure.

Alors, en posant U = MA et L = M^{-1}, on obtient :

A = LU

Supposons maintenant qu'on n'a jamais échangé des lignes.

Alors, L = $(E^1)^{-1} \cdots (E^{n-1})^{-1}$, avec :

$$E_k = \begin{pmatrix} 1 & & & \cdots & & \\ & \ddots & & & & \\ & & 1 & \ddots & & \\ \vdots & \vdots & -l_{(k+1)k} & \ddots & & \\ & & \vdots & & 1 & \\ & & -l_{nk} & & & 1 \end{pmatrix} \begin{array}{l} \\ \\ \ldots\ldots \text{ligne } k \\ \ldots\ldots \text{ligne } (k+1) \\ \\ \\ \end{array}$$

Et ou l'on a posé $l_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$

Condition suffisante pour la factorisation LU d’une matrice :

Soit A = (aij) et ∀k = 1, . . ., n, on a :

$$\nabla_k = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{bmatrix}$$

$\det(A) \neq 0$. Alors il existe une matrice L = (l_{ij}) triangulaire inferieure avec

l_{ii} = 1, ∀i = 1, . . ., n et une matrice U triangulaire superieure telles que A = LU. De plus, une telle decomposition est unique.

Un cas particulier ou det(∇_k) ≠ 0 pour tout k = 1, . . . , n, est celui d’une matrice symetrique definie positive (plus precisement on a dans ce cas det(∇_k) > 0).

b) Méthode de CHOLESKY:

Théorème

Factorisation de Cholesky d'une matrice — Si A est une matrice symétrique définie positive, il existe une matrice réelle triangulaire inférieure C telle que : A=C tC.

On cherche la matrice C :

$$C = \begin{bmatrix} C_{11} & & \\ \vdots & \ddots & \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}$$

De l'égalité A= C tC .

La méthode de Cholesky démarre du principe de décomposition de A en produit de 2 matrice C et sa transposé tC

A = C tC = L N N -1 U tel que N est une matrice diagonale
= C tC

Pour cela on procède soit par identification (qui consiste à remplir la matrice C à l’aide des éléments de la matrice A et des éléments déjà trouvé de C) ou en utilisant la méthode LU (C =LN tel que N est une matrice diagonale est les elements diagonaux sont égaux à la racine de elements de la diagonale de U => nii = uii

3) Algorithmes des méthodes :

a) Fonction additionnelle :

Cette methode sert a générer des matrices carrée aléatoire qui sont symétrique définit positive, elle est utilisée pour générer des matrices de grande taille pour le teste de la methode LU et CHOLESKY.

Code Matlab :

```
function A = MATRICE_SDP(N)
 bool = 0;
while( bool == 0)
    A = rand(N,N); % éléments extra
diago
    A= (A + A')/2; % symétrisation
    A = 10*(A*A');
    A = floor(A);
    if(min(eig(A)) > 0)
        bool = 1;
    end
end
end
```

b) Methode LU :

On a créer 2 fonctions une pour le calcule des deux matrices L et U ; et l'autre pour résoudre le système d'équation par la méthode LU :

Fonction LU_FACTORIZATION :

Cette fonction fait le calcule des deux matrice L et U :

```
function [ L , U ] = LU_FACTORIZATION(A)
   rows = size(A,1);
   columns = size(A,2);

   if( det(A) ~= 0 && rows == columns )
      L = eye(rows);
      for k=1:rows
         L(k+1:rows, k) = A(k+1:rows, k)/A(k, k);
         for i=k+1:rows
            A(i, :) =  A(i, :) - L(i, k)*A(k, :);
         End
      end
      U = A;
      A = L*U;
      elseif( det(A) == 0 )
          error('La matrice insere est non inversible');
      elseif( rows ~= columns )
          error('La matrice insere n est pas carrée');
      end
end
```

Fonction LU_FACTORIZATION_SOLUTION :

Cette fonction fait résoudre un système d'équation avec la méthode LU on utilisant la fonction précédente ou en calcule le temps du processus

```
function [ X , L , U ] = LU_FACTORIZATION_SOLUTION( A , b )

    tStart = tic;

    [ L , U ] = LU_FACTORIZATION(A);
    rows = size(A,1);
    Y = zeros(rows,1);
    for i = 1 : rows
        n = 0;
        for j = 1 : rows
           if i ~= j
              n = n + L(i,j)*Y(j,1);
           end
        end
        Y(i,1) = (b(i,1) - n)/L(i,i);
    end

    X = zeros(rows,1);
    for i = rows : -1 : 1
       n = 0;
       for j = 1 : rows
           if i ~= j
              n = n + U(i,j)*X(j,1);
           end
       end
       X(i,1) =  (Y(i,1) - n)/U(i,i);

    end
    tEnd = toc(tStart);
    fprintf('%d minutes and %f seconds\n',floor(tEnd/60),rem(tEnd,60));
end
```

c) Methode de CHOLESKY :

On a créer 2 fonctions une pour le calcule de la matrice C ; et l'autre pour résoudre le système d'équation par la méthode CHOLESKY :

Fonction Cholesky :

Cette fonction fait le calcule la matrice C :

```
function R = CHOLESKY(A)
    tic;
    rows = size(A,1);
    columns = size(A,2);
    eig_A = eig(A);
     if( det(A) ~= 0 && rows == columns && norm(A-
     transpose(A), 1) ==  0 && min(eig_A) > 0 )
        n=length(A);
        R=zeros(n);
        for k=1:n
            R(k,k) = sqrt(A(k,k) - R(k,:)*R(k,:)');
            for j = (k+1) : n
                R(j,k) = (A(j,k) - R(k,:) * R(j,:)')/R(k,k);
            end
        end
    elseif( det(A) == 0 )
        error('La matrice insere est non inversible');
    elseif( rows ~= columns )
        error('La matrice insere n est pas carrée');
    elseif(norm(A-transpose(A), 1) ~= 0)
        error('La matrice insere n est pas symetrique');
    elseif(min(eig_A) <= 0)
        error('La matrice insere n est pas definit positive');
    end
end
```

Fonction CHOLESKY_SOLUTION :

Cette fonction fait résoudre un système d'équation avec la méthode LU on utilisant la fonction précédente ou en calcule le temps du processus

```
function [ X , C ] = CHOLESKY_SOLUTION( A , b )
    tStart = tic;
    C = CHOLESKY(A);
    rows = size(A,1)
    Y = zeros(rows,1);
    for i = 1 : rows
        n = 0;
        for j = 1 : rows
           if i ~= j
              n = n + C(i,j)*Y(j,1);
           end
        end
        Y(i,1) = (b(i,1) - n)/C(i,i);
    end
    X = zeros(rows,1);
    T = C';
    for i = rows : -1 : 1
       n = 0;
       for j = 1 : rows
           if i ~= j
              n = n + T(i,j)*X(j,1);
           end
       end
       X(i,1) =  (Y(i,1) - n)/T(i,i);
    end
    tEnd = toc(tStart);
    fprintf('%d minutes and %f seconds\n',floor(tEnd/60),rem(tEnd,60));
end
```

4) *Jeu d'essai :*

a) Jeu d'essai pour la méthode LU :

Exemple 1 :

On déclare une matrice non inversible pour tester est ce qu'il est possible de la traiter ou non :

```
>> A = [ 10 3 9 10 ; 8 -5 3 8 ; 15 0 6 15 ; -10 8 3 -10 ]

A =

    10     3     9    10
     8    -5     3     8
    15     0     6    15
   -10     8     3   -10
```

Maintenant on va exécuter la fonction LU_FACTORISATION :

```
>> [L , U] = LU_FACTORIZATION(A)
Error using LU_FACTORIZATION (line 27)
La matrice insere est non inversible
```

Une erreur est survenue car la matrice A elle n'est pas inversible.

Exemple 2 :

On déclare une matrice non carrée pour tester est ce qu'il est possible de la traiter ou non :

```
>> A = 100*rand(4,5)

A =

    8.2858   21.1069    1.5138   24.6007   75.4390
   63.3161    0.1446   71.4978   26.6085   88.9239
   31.0979   79.7611   21.8579   13.2071   71.0436
   46.6354   36.8047   68.8717   34.4210   24.6475
```

Maintenant on va exécuter la fonction LU_FACTORISATION :

```
>> [ L , U ] = LU_FACTORIZATION(A)
Error using eig
For standard eigenproblem EIG(A), A must be
square.
```

Une erreur est survenue car la matrice A elle n’est pas carrée.

Exemple 3 (partie 1) :

On déclare une matrice a l’aide de la fonction « MATRICE_SDP » :

```
>> A = MATRICE_SDP(5)

A =

    15    12     8    13    13
    12    14    11    10    13
     8    11     9     7    10
    13    10     7    11    12
    13    13    10    12    15
```

Maintenant on va exécuter la fonction LU_FACTORISATION :

```
>> [ L , U ] = LU_FACTORIZATION(A)

L =

    1.0000         0         0         0         0
    0.8000    1.0000         0         0         0
    0.5333    1.0455    1.0000         0         0
    0.8667   -0.0909   -6.4000    1.0000         0
    0.8667    0.5909   -4.6000    1.1429    1.0000


U =

   15.0000   12.0000    8.0000   13.0000   13.0000
         0    4.4000    4.6000   -0.4000    2.6000
         0         0   -0.0758    0.4848    0.3485
         0         0         0    2.8000    3.2000
         0         0    0.0000    0.0000    0.1429
```

Donc on remarque que la matrice respecte les critères de la factorisation LU

Exemple 3 (partie 2) :

Avec la même matrice A de la partie 1 de l'exemple 3 on va résoudre le système A*x = b

On déclare le vecteur b :

```
>> b = [ 1 ; 0 ; 0 ; 0 ; 0 ]

b =

     1
     0
     0
     0
     0
```

Maintenant on va exécuter la fonction LU_FACTORIZATION_SOLUTION :

```
>> [ X , L , U ] = LU_FACTORIZATION_SOLUTION( A , b )
0 minutes and 0.003433 seconds

X =

   -0.5000
   -0.5000
    1.0000
    1.5000
   -1.0000


L =

    1.0000         0         0         0         0
    0.8000    1.0000         0         0         0
    0.5333    1.0455    1.0000         0         0
    0.8667   -0.0909   -6.4000    1.0000         0
    0.8667    0.5909   -4.6000    1.1429    1.0000


U =

   15.0000   12.0000    8.0000   13.0000   13.0000
         0    4.4000    4.6000   -0.4000    2.6000
         0         0   -0.0758    0.4848    0.3485
         0         0         0    2.8000    3.2000
         0         0    0.0000    0.0000    0.1429
```

b) Jeu d'essai pour la méthode CHOLESKY :

Exemple 1 :

On déclare une matrice non inversible pour tester est ce qu'il est possible de la traiter ou non :

```
>> A = [ 10 3 9 10 ; 8 -5 3 8 ; 15 0 6 15 ; -10 8 3 -10 ]

A =

    10     3     9    10
     8    -5     3     8
    15     0     6    15
   -10     8     3   -10
```

Maintenant on va exécuter la fonction CHOLESKY :

```
>> C = CHOLESKY(A)
Error using CHOLESKY (line 23)
La matrice insere est non inversible
```

Une erreur est survenue car la matrice A elle n'est pas inversible.

Exemple 2 :

On déclare une matrice non carrée pour tester est ce qu'il est possible de la traiter ou non :

```
>> A = 100*rand(4,5)

A =

    8.2858   21.1069    1.5138   24.6007   75.4390
   63.3161    0.1446   71.4978   26.6085   88.9239
   31.0979   79.7611   21.8579   13.2071   71.0436
   46.6354   36.8047   68.8717   34.4210   24.6475
```

Maintenant on va exécuter la fonction CHOLESKY :

```
>> C = CHOLESKY(A)
Error using eig
For standard eigenproblem EIG(A), A must be
square.
```

Une erreur est survenue car la matrice A elle n'est pas carrée.

Exemple 3 :

On déclare une matrice carrée qui n'est pas symétrique :

```
>> A = [ 10 3 9 10 ; 3 -5 0 8 ; 9 0 6 15 ; -10 8 15 -10 ]

A =

    10     3     9    10
     3    -5     0     8
     9     0     6    15
   -10     8    15   -10
```

Maintenant on va exécuter la fonction CHOLESKY :

```
>> C = CHOLESKY(A)
Error using CHOLESKY (line 27)
La matrice insere n est pas symetrique
```

Une erreur est survenue car la matrice A elle n'est pas symétrique.

Exemple 4 :

On déclare une matrice carrée qui est symétrique mais non définit positive :

```
>> A = [ 10 3 9 10 ; 3 -5 0 8 ; 9 0 6 15 ; 10 8 15 -10 ]

A =

    10     3     9    10
     3    -5     0     8
     9     0     6    15
    10     8    15   -10
```

Maintenant on va exécuter la fonction CHOLESKY :

```
>> C = CHOLESKY(A)
Error using CHOLESKY (line 29)
La matrice insere n est pas definit positive
```

Une erreur est survenue car la matrice A elle n'est pas définit positive.

Exemple 5 (partie 1) :

On déclare une matrice a l'aide de la fonction « MATRICE_SDP » :

```
>> A = MATRICE_SDP(5)

A =

    11     6    10    10     9
     6     6     8     7     7
    10     8    17     9    12
    10     7     9    15    11
     9     7    12    11    14
```

Maintenant on va exécuter la fonction CHOLESKY:

```
>> C = CHOLESKY(A)

C =

    3.3166         0         0         0         0
    1.8091    1.6514         0         0         0
    3.0151    1.5413    2.3523         0         0
    3.0151    0.9358   -0.6518    2.1467         0
    2.7136    1.2661    0.7935    1.0018    1.8439
```

Donc on remarque que la matrice respecte les critères de la factorisation LU

Exemple 5 (partie 2) :

Avec la même matrice A de la partie 1 de l'exemple 5 on va résoudre le système A*x = b

On déclare le vecteur b :

```
>> b = [ 1 ; 0 ; 0 ; 0 ; 0 ]

b =

     1
     0
     0
     0
     0
```

Maintenant on va exécuter la fonction CHOLESKY_SOLUTION :

```
>> [ X , C ] = CHOLESKY_SOLUTION(A,b)
0 minutes and 0.004927 seconds

X =

    0.3399
   -0.0065
   -0.1242
   -0.1634
    0.0196


C =

    3.3166         0         0         0         0
    1.8091    1.6514         0         0         0
    3.0151    1.5413    2.3523         0         0
    3.0151    0.9358   -0.6518    2.1467         0
    2.7136    1.2661    0.7935    1.0018    1.8439
```

5) *Estimation du temps d'exécution :*

On déclare une matrice de grande taille (20 * 20) à l'aide de la fonction « MATRICE_SDP » :

```
>> A = MATRICE_SDP(20)

A =

    73    60    58    54    51    63    53    58    58    62    56    64    56    61    60    44    60    55    55    57
    60    64    57    49    49    59    51    53    51    59    51    57    50    56    55    44    57    50    50    52
    58    57    66    52    49    61    51    51    52    57    52    61    48    61    55    41    56    47    54    54
    54    49    52    55    41    57    49    47    46    51    47    53    44    51    51    40    55    46    50    46
    51    49    49    41    50    51    44    47    48    49    43    53    47    49    45    38    50    46    43    47
    63    59    61    57    51    71    56    54    57    62    57    63    52    61    59    48    61    56    54    55
    53    51    51    49    44    56    59    48    50    53    53    59    46    53    50    42    59    53    51    50
    58    53    51    47    47    54    48    60    50    56    48    57    49    53    53    42    56    50    49    49
    58    51    52    46    48    57    50    50    60    50    51    57    49    55    55    40    58    49    50    50
    62    59    57    51    49    62    53    56    50    67    53    60    52    58    53    48    60    53    54    54
    56    51    52    47    43    57    53    48    51    53    58    56    46    55    52    45    56    50    50    49
    64    57    61    53    53    63    59    57    57    60    56    75    55    64    60    45    62    58    56    59
    56    50    48    44    47    52    46    49    49    52    46    55    54    52    48    40    54    48    47    45
    61    56    61    51    49    61    53    53    55    58    55    64    52    66    57    44    59    51    54    52
    60    55    55    51    45    59    50    53    55    53    52    60    48    57    70    42    59    52    55    49
    44    44    41    40    38    48    42    42    40    48    45    45    40    44    42    43    48    41    41    40
    60    57    56    55    50    61    59    56    58    60    56    62    54    59    59    48    69    56    59    53
    55    50    47    46    46    56    53    50    49    53    50    58    48    51    52    41    56    59    48    49
    55    50    54    50    43    54    51    49    50    54    50    56    47    54    55    41    59    48    56    49
    57    52    54    46    47    55    50    49    50    54    49    59    45    52    49    40    53    49    49    56
```

On va résoudre le système Ax = b avec les deux methode LU et CHOLESKY
Tel que :

```
>> b = [ 1 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ]

b =

     1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

a) Temps d'exécution Avec la méthode LU :

```
>> [ X , L , U ] = LU_FACTORIZATION_SOLUTION(A,b)
0 minutes and 0.130355 seconds

X =

    1.5479
   -0.1156
    0.7315
   -1.3648
   -0.0850
    0.8690
    0.4280
    0.1236
   -0.7751
   -1.4457
   -1.1126
    0.0376
   -0.5811
   -0.1444
   -0.4131
    1.2190
    0.7513
    0.0832
    0.6063
   -0.5275
```

b) Temps d'exécution Avec la méthode CHolesky :

```
>> [ X , C ] = CHOLESKY_SOLUTION(A,b)
0 minutes and 0.003506 seconds

X =

    1.5479
   -0.1156
    0.7315
   -1.3648
   -0.0850
    0.8690
    0.4280
    0.1236
   -0.7751
   -1.4457
   -1.1126
    0.0376
   -0.5811
   -0.1444
   -0.4131
    1.2190
    0.7513
    0.0832
    0.6063
   -0.5275
```

Ceci a été clairement remarqué dans les différents exemples exécutés sous Matlab. La fonction impressionnante tic…toc nous a clairement permis de voir la différence entre les 2 méthodes en terme de temps d'exécution et d'aboutir à l'efficacité de la 2ème.