ÉCOLE INTERNATIONALE DES SCIENCES
DU TRAITEMENT DE L'INFORMATION

MATHÉMATIQUES APPLIQUÉES

Factorisation LU

Bara DIOUKHANE
Quentin DUCASSE

Semestre 2
Promotion 2022

2019

Table des matières

Introduction

La factorisation LU est une méthode de décomposition d'une matrice comme produit d'une matrice triangulaire inférieure L (comme lower, inférieure en anglais) par une matrice triangulaire supérieure U (comme upper, supérieure). Cette décomposition est utilisée en analyse numérique pour résoudre des systèmes d'équations linéaires de la forme $AX = B$ avec A une matrice carré d'ordre n inversible. Au cours de ce travail pratique, l'objectif sera de chercher à résoudre un système linéaire quelconque à l'aide de la décompositon LU.

1 Présentation

1.1 Méthode du pivot de Gauss

Présentation et intêret

La méthode du Pivot de Gauss d'un système linéaire permet de ramener un système quelconque $Ax = b$ à un sytème triangulaire supérieur $Ux = b'$ plus facile à résoudre.

Soit un système linéaire de la forme :

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 & (L_1) \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 & (L_2) \\ \vdots \\ \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n = b_m & (L_m) \end{cases}$$

Celui ci peut également s'écrire sous forme matricielle :

$$\text{A=}\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} ; \quad \text{x=}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \text{et} \quad \text{b=}\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

tel que $A \times x = b$

La méthode de résolution d'un système linéaire par l'utilisation du pivot de Gauss utilise les 3 opérations élémentaires sur les matrices :
- Échange de deux lignes ;
-Multiplication d'une ligne par un scalaire non nul ;
-Ajout du multiple d'une ligne à une autre ligne.

Le but de la méthode du pivot de Gauss est de se sérvir de ces 3 opérations pour parvenir à un systéme linéaire plus facile à étudier. Dans notre cas, nous cherchons à construire un système triangulaire supérieur $U \times x = bb$ avec U une matrice triangulaire supérieure et bb un vecteur colonne.

Application à un système linéaire

Soit un système linéaire :

$$\begin{pmatrix} 1 & 2 & 1 \\ 1 & 3 & -2 \\ 3 & 5 & 8 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 8 \end{pmatrix} \Longleftrightarrow \begin{cases} x + 2y + z = 2 & (L_1) \\ x + 3y - 2z = -1 & (L_2) \\ 3x + 5y + 8z = 8 & (L_3) \end{cases}$$

On cherche à isoler une des trois variables.
On conserve la ligne (L_1) dont on va se servir comme pivot pour éliminer la variable x des autres lignes ;
On effectue donc :

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & -4 \\ 0 & -1 & 2 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ 2 \end{pmatrix} \Longleftrightarrow \begin{cases} x + 2y + z = 2 & (L_1) \\ y - 4z = -3 & (L_2 \leftarrow L_2 - L_1) \\ -y + 2z = 2 & (L_3 \leftarrow L_3 - 3L_1) \end{cases}$$

On conserve cette fois ci la ligne (L_2) qu'on utilise comme pivot pour éliminer la variable y de la troisième ligne. On trouve :

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & -4 \\ 0 & 0 & -2 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ -1 \end{pmatrix} \Longleftrightarrow \begin{cases} x + 2y + z = 2 & (L_1) \\ y - 4z = -3 & (L_2) \\ -2z = -1 & (L_3 \leftarrow L_3 + L_2) \end{cases}$$

Ce dernier système triangulaire supérieur, est plus facile à résoudre.

Choix algorithmique

On part d'un système $A \times x = b$ qui pourrait s'avérer difficile à résoudre. Le principe de l'algorithme $gaussPivot$ est de se ramener à un système où la matrice A devient triangulaire et le vecteur b s'y adapte. Concernant la signature de la matrice, on prend donc en paramètre la matrice A carré et le vecteur b colonne tel que $A \times x = b$, et on renvoie la matrice U carrée triangulaire supérieure et le vecteur colonne bb tel que $U \times x = bb$
La complexité de cet algorithme est de l'ordre de n^3

```
function [U,bb]=gauss_pivot(A,b)
     [n] =length(b)
    [m1,m2]=size(A);
    //on initialise les matrices U et bb aux matrices A et b en paramètre
    bb=b;
    U=A;
    for i = 1 : n-1
    num=i
     for k = 1 : n-1
         if U(k,i)>U(num,i)  then
             num=k
         end
         //effectue un échange des lignes si la coordonnée num est infèrieur à la coordonnée k
         if num<>i then
             for j = i : n
                 tmp=A(num,j)
                 A(num,j)=A(i,j)
                 A(i,j)=tmp
             end
         end
     end
     pivot = U(i,i)
     for k = (i+1):n
         //on calcule le premier inconnue grace au pivot
        fact=(U(k,i)/pivot)
      for j = i : n
       //on determine les éléments de la matrice en remontant en se servant des valeurs U calculé
         U(k,j)=U(k,j)-fact*U(i,j)
     end
     //on détermine le vecteur bb en utilisant le fact obtenue dans le calcul ci dessus
     bb(k)=bb(k)-fact*bb(i)
 end

 end
endfunction
```

1.2 Définition et propriétés

En supposant que la méthode du pivot de GAUSS puisse être réalisée sans aucun échange de lignes (autrement dit, si il n'y a pas de pivot nul) alors AU^{-1} est une matrice triangulaire inférieure L et $A = LU$ Dans le cas contraire alors cela signifie que un pivot est nul (c'est a dire une valeur de la diagonale de la matrice A est nul). Pour eviter que le pivot soit nul on permute la ligne qui contient le pivot avec une autre jusqu'à ce que le pivot ne soit plus nul Alors dans ce cas la on note P la nouvelle matrice obtenue en permutant les lignes. Alors $A = PLU$

Définition : [Décomposition LU]
Soit n un entier naturel non nul et A une matrice carrée d'ordre n à coefficients réels. On dit que A admet une décomposition LU si et seulement si il existe une matrice L triangulaire inférieure d'ordre n dont les coefficients diagonaux sont tous égaux à un et une matrice U triangulaire supérieure d'ordre n telles que A = LU

Théorème : [Condition nécessaire et suffisante d'existence de décomposition LU]
Soit n un entier naturel non nul et A une matrice carrée d'ordre n à coefficients réels. Pour tout entier $p \in [1; n]$, on note $A_p = (A_{ij})_{1 \geq i \geq p, 1 \geq j \geq p}$. Alors A admet une décomposition LU si et seulement si pour tout entier $p \in [1; n-1]$, la matrice A_p est inversible.

2 Décomposition d'une matrice sous une forme LU

2.1 Exemple

Soit une matrice $A = \begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix}$

On cherche une matrice carrée inférieure L et une matrice supérieure U tel que : $A = LU$.

C'est a dire $\begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$

Etape 1 : Verification de l'existence de la decomposition LU

D'après la définition la décomposition LU de A existe si et seulement si A est inversible. C'est a dire si le determinant de A est non nul.

$$\begin{aligned}
det(A) &= \begin{bmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{bmatrix} \begin{matrix} L_1 -> L_1 - 2L_3 \\ L_2 -> L_2 - L_3 \end{matrix} \\
&= \begin{bmatrix} 0 & -6 & -8 \\ 0 & -2 & -5 \\ 1 & 5 & 6 \end{bmatrix} \\
&= +1 \begin{bmatrix} -6 & -8 \\ -5 & -2 \end{bmatrix} \\
&= -6 \times (-2) - (-8) \times (-5) \\
det(A) &= 14
\end{aligned}$$

Donc $det(A) = 14 \neq 0$

Donc il existe une matrice trianglaire inferieure L et une matrice triangulaire supérieure U tel quel $A = LU$

Etape 2 : Résolution du système

$$A = LU \iff \begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$\iff \begin{cases} u_{11} = 2 \\ u_{12} = 4 \\ u_{13} = 4 \\ l_{21}u_{11} = 1 \\ l_{31}u_{11} = 1 \\ l_{21}u_{12} + u_{22} = 3 \\ l_{21}u_{13} + u_{23} = 1 \\ l_{31}u_{12} + l_{32}u_{22} = 5 \\ l_{31}u_{32} + l_{23}u_{33} = 6 \end{cases} \iff \begin{cases} u_{11} = 2 \\ u_{12} = 4 \\ u_{13} = 4 \\ l_{21} = \frac{1}{2} \\ l_{31} = \frac{1}{2} \\ u_{22} = 3 - l_{21}u_{12} = 1 \\ u_{23} = 1 - l_{21}u_{13} = -1 \\ u_{32} = \frac{5 - l_{31}u_{12}}{u_{22}} = 3 \\ u_{33} = 6 - l_{31}u_{13} - l_{32}u_{23} = 7 \end{cases}$$

Etape 3 : Conclusion

Donc $A = \begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 3 & 1 \end{pmatrix}}_{L} \times \underbrace{\begin{pmatrix} 2 & 4 & 4 \\ 0 & 1 & -1 \\ 0 & 0 & 7 \end{pmatrix}}_{U}$

On peut retrouver la valeur du determinant de A a l'aide de la nouvelle decomposition LU :

$$\begin{aligned} det(A) &= det(LU) \\ &= det(L) \times det(U) \end{aligned}$$

Comme L et U sont des matrices triangulaires alors :

$det(L) = 1 \times 1 \times 1$
$det(U) = 2 \times 1 \times 7$

Donc $det(A) = det(L) \times det(U) = 1 \times 7 = 14$

On retrouve le même resultat que le résultat trouvé lors de l'étape 1

De manière générale pour toute matrice A carré d'ordre n decomposable en LU, $det(A) = \prod_{i=1}^{n} U_{ii}$

2.2 Choix algorithmiques

myLU

La foncion $myLU$ permet de calculer la décomposition LU d'une matrice carré A d'ordre n et de la stocker à la place de A. Si un pivot est nul , la décomposition LU n'existe pas.
Cette fonction prend donc en paramètre une matrice carré A et renvoit la decomposition LU stockée dans A

```
function [LU]=my_LU(A)

    [n,n]=size(A);
    j=1;
    // on suppose que le pivot existe (pivot different de 0)
    bool=%t;
    // tant que le pivot est different de 0 et qu'on n'arrive pas au boout de la diagonale
    while ((j<=n-1) & (bool)) do
                // Si le pivot est nul
              if  (A(j,j)==0) then
                   // la mtrice est = nan (pas de valeur)
                   LU =%nan
                   // on stope l'algorithme et on renvoie A=nan
                   bool=%f;

              // si le pivot est non nul
        else
                    //on effectue les opérations décrites dans la méthode générale
                    // Donc on divise les elements de la matrice par le pivot
                    //      on effectue des opérations sur les lignes sans les permuter
                    for i=j+1:n
                        A(i,j)=A(i,j)/A(j,j)
                        for k=j+1:n
                            A(i,k)=A(i,k)-A(i,j)*A(j,k)
                        end

                        // on renvoie la nouvelle matrice calculée après cette succesions d'opérati
                        LU=A
                    end
                end
        j=j+1;
        end
endfunction
```

Pour améliorer l'algorithme ci dessus, dans le cas ou le pivot est nul, il faudrait inverser la ligne contenant le pivot avec une autre jusqu'à ce que le pivot ne soit plus nul. Il suffit de creer une nouvelle fonction qui calcule la nouvelle matrice P suite aux permutations de lignes. Dans ce cas là, on calcul PLU tel que A=PLU.

extractLU

La fonction $extractLU$ permet d'extraire les matrices L et U de LU qui sont respectivement triangulaires inférieures et supérieures.
Cette fonction prend en paramètre la matrice carré décomposée LU et renvoit les deux matrices L et U.

```
function [L,U]=extract_LU (LU)
    [n,n]=size(LU);
    
    // on initialise les  2 matrices a 0 
    L=zeros(n,n);
    U=zeros(n,n);
    
    // si la matrice LU a extraire n'existe pas alors les matrices L et U n'existent pas non plus
    if (LU==%nan) then
        L=%nan
        U=%nan
    // si LU existe 
    else
      
      // On parcour la partie supérieure de la matrice 
      
       for i=1:n
           for j=1:(i-1)
               // on remplit la partie supéreure de L par la partie supérieure de LU
               L(i,j)=LU(i,j);
           end
       end
       
       // on parcourt la diagonale
       
       for i=1: n
           // D'apres la definition, les elemnts de la diagonales sont égales a 1
           L(i,i)=1;
       end
       
       
       // on parcourt toute la matrice U (initialement nulle)
       for i=1:n
           for j=1:n
               // si l'elemnt de la matrice L est egal a 0
               if (L(i,j)==0) then
                   // alors cette partie inferieur de U prend les valeure de LU
                   U (i,j)=LU(i,j);
               end
           end
               
       end
       
       // on remplit la diagonale par les valeur de la diagonale de LU
        for i=1: n
           U(i,i)=LU(i,i);
       end
   end
          
endfunction
```

Par exemple, les matrices L et U extraites de $LU = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ sont :

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 8 & 1 \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{pmatrix}$$

calcul du determinant

Pour faciliter la compréhension on note $L \times U$ le produit de L et U.
On note LU la matrice qui stocke la decomposition LU.
La fonction $myDet$ permet de calculer le determiant d'une matrice A a l'aide de sa décomposition LU.
Il suffit de calculer le determinant de $L \times U$. Comme le determinant de L est égal à 1 cela revient a calculer le determiant de U.
U et la matrice LU (qui stocke la decomposition LU) ont la même diagonale. Donc il suffit de calculer le determiant de LU.
Ainsi la fonction prend en paramètre d'entrée une matrice A carré et renvoit son determiant.
La complexité de cet algorithme est de l'orde de n. Calculer le determiant de la matrice triangulaire supéreiure U est beaucoup plus efficace que de calculer le determiant d'une matrice carré quelconque d'ordre n . En effet la complexité d'un tel algorithme est égale a $n!$.

```
function [d]=my_det(A)
    [n,n]=size(A);
    // on initialise le determinant a 1
    d=1;
    
    // on calcul la decomposition LU de A
    A=my_LU(A);
    
    // On parcourt la diagonale de LU
    for i=1:n
        // on multiplie les valeurs de la diagonale
        d=d*A(i,i);
    end
    
endfunction
```

Par exemple on veut calculer le determinant de $A = \begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix}$:

On sait que : $L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 3 & 1 \end{pmatrix}$ et $U = \begin{pmatrix} 2 & 4 & 4 \\ 0 & 1 & -1 \\ 0 & 0 & 7 \end{pmatrix}$ et $LU = \begin{pmatrix} 2 & 4 & 4 \\ \frac{1}{2} & 1 & -1 \\ \frac{1}{2} & 3 & 7 \end{pmatrix}$

ALors on remarque que U et LU ont la même diagonale.
Donc $det(A) = \prod_{i=1}^{3} LU_{ii} = 2 \times 1 \times 7 = 14$

3 Résolution d'un système linéaire de forme AX=B

3.1 Methode générale

Soit A une matrice triangulaire supérieur d'ordre n :

$$A^{i+1} = \begin{pmatrix} a_{1,1}^{(i)} & \cdots & \cdots & \cdots & a_{1,n}^{(i)} \\ 0 & \ddots & \cdots & \cdots & \vdots \\ \vdots & \vdots & a_{i+1,i+1}^{(i+1)} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & a_{n,n}^{(i+1)} \end{pmatrix};$$

Le résultat général que l'on peut démontrer, est que si la matrice A est inversible, alors il existe une matrice de permutation P, une matrice triangulaire inférieure L et une matrice triangulaire supérieure U telles que P$\times A = LU$

Le cas général d'une matrice n $\times$ n

De manière plus générale, pour une matrice A carrée d'ordre n, la méthode de Gauss s'écrit :
On pose $A^{(1)}$ = A et $b^{(1)}$= b. Pour i = 1,...,n 1, on cherche à calculer $A^{(i+1)}$ et $b^{(i+1)}$ tels que les systèmes $A^{(i)}x$ = $b^{(i)}$ et $A^{(i+1)}x$ = $b^{(i+1)}$ soient équivalents, où $A^{(i+1)}$ est une matrice dont les coefficients sous-diagonaux des colonnes 1 à i sont tous nuls.
Une fois la matrice $A^{(n)}$ (triangulaire supérieure) et le vecteur $b^{(n)}$ calculés, il sera facile de résoutre le système $A^{(n)}$ x = $b^{(n)}$. Le calcul de $A^{(n)}$ est l'étape de "factorisation", le calcul de $b^{(n)}$ l'étape de "descente", et le calcul de x l'étape de "remontée". Donnons les détails de ces trois étapes.

Etape de factorisation et descente

Pour passer de la matrice $A^{(i)}$ à la matrice $A^{(i+1)}$, on va effectuer des combinaisons linéaires entre lignes qui permettront d'annuler les coefficients de la i-ème colonne situés en dessous de la ligne i (dans le but de se rapprocher d'une matrice triangulaire supérieure). Evidemment, lorsqu'on fait ceci, il faut également modifier le second membre b en conséquence. L'étape de factorisation et descente s'écrit donc :

1. Pour k $\leq$ i et pour j = 1,...,n, on pose $a_{k,j}^{(i+1)}$ =$a_{k,j}^{(i)}$ et $b_k^{(i+1)} = b_k^{(i)}$
2. Pour k > i, si $a_{i,i}^{(i)} \neq 0$, on pose :
$a_{k,j}^{(i+1)}$ =$a_{k,j}^{(i)}$-$\frac{a_{k,i}^{(i)}}{a_{i,i}^{(i)}}a_{i,j}^{(i)}$ pour k=j,...,n
Remarquons que le système $A^{(i+1)}$ x = $b^{(i+1)}$ est bien équivalent au système $A^{(i)}$ x = $b^{(i)}$. Si la condition $a_{i,i}^{(i)} \neq 0$ est vérifiée pour i = 1 à n, on obtient par le procédé de calcul ci-dessus un système linéaire $A^{(n)}$ x = $b^{(n)}$. équivalent au système Ax = b, avec une matrice $A^{(n)}$ triangulaire supérieure facile à inverser.
Dans le cas où $a_{i,i}^{(i)} = 0$ c'est à dire si le pivot est nul alors on peut résoudre ce problème en utlisant la

technique du "pivot partiel" qui revient a choisir une matrice de permutation P. Pour cela on échange la ligne qui contient le pivot nul avec une autre ligne jusqu'à ce que le pivot ne soit plus nul puis on continue la procédure de Gauss décrite plus haut.
L'interet de cette stratégie de "pivot partiel" est qu'on aboutit toujours à la résolution du système (seulement si A est inversible)

Etape de remontée

Il reste à résoutre le système $A^{(n)}\ x = b^{(n)}$. Ceci est une étape facile. Comme $A^{(n)}$ est une matrice inversible, on a $a_{i,i}^{(i)} \neq 0$ pour tout i=1,...,n, et comme $A^{(n)}$ est une matrice triangulaire supérieure, on peut donc calculer les composantes de x en "remontant", c'est–à–dire de la composante x_n à la composante x_1 :
$x_n = \frac{b_n^{(n)}}{a_{n,n}^{(i)}}$
$x_i = \frac{1}{a_{i,i}^{(n)}}\ (b^{(i)}\ \text{-}\ \sum_{j=i+1}^{n} a_{i,j}^{(n)} x_j)$ avec i= n-1,...,1

3.2 Exemple de résolution d'un système linéaire de la forme AX=B

Soit $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ et $B = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$
On cherche a résoudre le système linéaire $AX = B$

AX=B $\iff \begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

$$\iff \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 3 & 1 \end{pmatrix}}_{L} \times \underbrace{\begin{pmatrix} 2 & 4 & 4 \\ 0 & 1 & -1 \\ 0 & 0 & 7 \end{pmatrix}}_{U} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Donc $X = U^{-1}(L^{-1}B)$

Etape 1 : calcul de L^{-1} et U^{-1}

Par définition , $L \times L^{-1} = Id_3$

On pose $L^{-1} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$

On a donc :

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 3 & 1 \end{pmatrix} \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\iff \begin{cases} a = 1 \\ b = 0 \\ c = 0 \\ \frac{1}{2}a + d = 0 \\ \frac{1}{2}b + e = 1 \\ \frac{1}{2}c + f = 0 \\ \frac{1}{2}a + 3d + g = 0 \\ \frac{1}{2}b + 3e + h = 0 \\ \frac{1}{2}c + 3f + i = 0 \end{cases} \iff \begin{cases} a = 1 \\ b = 0 \\ c = 0 \\ d = -\frac{1}{2}a = -\frac{1}{2} \\ e = 1 - \frac{1}{2}b = 1 \\ f = -\frac{1}{2}c = 0 \\ g = -\frac{1}{2}a - 3d = 1 \\ h = -\frac{1}{2}b - 3e = -3 \\ i = -\frac{1}{2}c - 3f = 1 \end{cases}$$

Donc $L^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 1 & -3 & 1 \end{pmatrix}$

Pour determiner U^{-1} nous allons utiliser la formule :

$U^{-1} = \frac{1}{det(U)} \times com(U)^t$

$det(U) = 14$ (d'apres ci dessus)

$$\mathrm{U}^{-1} = \frac{1}{14} \begin{pmatrix} 7 & -28 & -8 \\ -0 & 14 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

$$U^{-1} = \begin{pmatrix} \frac{1}{2} & -2 & -\frac{4}{7} \\ 0 & 1 & \frac{1}{7} \\ 0 & 0 & \frac{1}{7} \end{pmatrix}$$

Etape 2 : Résolution du système

On cherche X tel que $AX = B$ avec :
D'après l'étape 1, $X = U^{-1}(L^{-1}B)$

On pose $Y = L^{-1}B$

$$Y = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 1 & -3 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{3}{2} \\ -2 \end{pmatrix}$$

X=U^{-1}Y

$= \begin{pmatrix} \frac{1}{2} & -2 & -\frac{4}{7} \\ 0 & 1 & \frac{1}{7} \\ 0 & 0 & \frac{1}{7} \end{pmatrix} \begin{pmatrix} 1 \\ \frac{3}{2} \\ -2 \end{pmatrix}$

$= \begin{pmatrix} \frac{-19}{14} \\ \frac{17}{14} \\ \frac{-2}{7} \end{pmatrix}$

Etape 3 : Conclusion

Donc la solution du système d'équation linéaire est donc
$X = \begin{pmatrix} \frac{-19}{14} \\ \frac{17}{14} \\ \frac{-2}{7} \end{pmatrix}$

On verifie que $AX = B$:

$$
\begin{aligned}
AX &= \begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \\
&= \begin{pmatrix} 2 & 4 & 4 \\ 1 & 3 & 1 \\ 1 & 5 & 6 \end{pmatrix} \begin{pmatrix} \frac{-19}{14} \\ \frac{17}{14} \\ \frac{-2}{7} \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \\
&= B
\end{aligned}
$$

3.3 Choix algortihmiques

UpperSolve

Pour résoudre un système triangulaire de la forme $UX = B$ avec U une matrice triangulaire supérieure, il suffit de calculer la solution a partir de la dernière ligne de la matrice car l'inconnue y est déjà isolé donc facile à déterminer. On effectue par la suite une remontée en se servant des valeurs des inconnues obtenues dans les calculs précédents . Cette fonction prend en paramètre une matrice U triangulaire supérieure et un vecteur colonne B et renvoie le vecteur X, contenant les inconnues permettant de résoudre le système. De manière générale, les calculs que l'on effectuent est celui ci :

$$
\begin{cases}
U_{11}X_1 + U_{12}X_2 + U_{13}X_3 + ... + U_{1n}X_n & = B_1 \\
U_{22}X_2 + U_{23}X_3 + ... + U_{2n}X_n & = B_2 \\
U_{33}X_3 + ... + U_{3n}X_n & = B_3 \\
... + ... & = ... \\
U_{nn}X_n & = B_n
\end{cases}
$$

```
function x = upper_solve(U, b)
    [n,n]=size(U);
    x(n)= b(n)/U(n,n)
    // on récupère la valeur de l'inconnue en (n,n) de la matrice qui est déjà isolé
    for i= (n-1):-1:1
         // on effectue la somme des éléments d'une ligne fois les éléments du vecteur x déjà cal
           som=0;
        for j= (i+1):n
            som=som + U(i,j)*x(j)
        end
       //on redescends la matrice en utilisant l'inconnue calculé à la ligne précèdente pour déte
        x(i)=(1/U(i,i))*(b(i)-som)
       end
endfunction
```

LowerSolve

De même, que pour la fonction précèdente , on résout un système triangulaire inférieure. Pour cela, on part cette fois ci de la première ligne de la matrice car l'inconnue est isolée. Puis on redescends la matrice en calculant les différents inconnues à partir de ceux calculés précédemment. La fonction prend donc en paramètre la matrice triangulaire infèrieure L et le vecteur colonne B et renvoie le vecteur colonne X, solution du système. Egalement, le calcul éffectué correspond au système ci dessous :

$$\begin{cases} L_{11}X_1 & = B_1 \\ ... + ... & = ... \\ L_{22}X_2 + L_{23}X_3 + ... + L_{2n}X_n & = B_{n-2} \\ L_{33}X_3 + ... + L_{3n}X_n & = B_{n-1} \\ L_{11}X_1 + L_{12}X_2 + L_{13}X_3 + ... + L_{1n}X_n & = B_n \end{cases}$$

```
function x = lower_solve(L, b)
    [m,n]=size(L);
    // on récupère la valeur de l'inconnue en (1,1) de la matrice qui est déjà isolé
    x(1)= b(1)/L(1,1)
    for i= 1 : m
        // on effectue la somme des éléments d'une ligne fois les éléments du vecteur x déjà calcu
        som=0;
        for j= 1: (i-1)
            som=som + L(i,j)*x(j)
        end
       //on redescends la matrice en utilisant l'inconnue calculé à la ligne précèdente pour déte
        x(i)=(1/L(i,i))*(b(i)-som)
    end
endfunction
```

résolution du système final

La fonction $mySolve$ permet de résoudre un sytème linéaire quelconque de la forme $AX = B$ avec A une matrice carré d'ordre n.
Elle utilise les fonctions $MyLU$, $extractLU$, $upperSolve$ et $lowerSolve$ implémentées précdemment.
La fonction prend en paramètre d'entrée la matrice A et un vecteur colonne B et renvoit le vecteur colonne X solution de l'équation.

```
function x= my_linsolve(A,b)

    // on stocke la decomosition Lu de A dans A
    A=my_LU(A);

    //On extrait L et U de A
    [L,U]=extract_LU(A);

    // on calcule y = L^(-1) x b
    y=lower_solve(L,b);
    // on calcule x = U^(-1) x y
    // renvoit x
    x=upper_solve(U,y);



endfunction
```

Conclusion

La méthode de la factorisation LU utilise la méthode du pivot de Gauss pour calculer les matrices triangulaires L et U. Cependant cette méthode est plus complète que la méthode du pivot de gauss qui ne traite que les résolutions des systèmes linéaire avec des matrices triangulaires supérieure. Ainsi la méthode de factorisation LU permet de résoudre n'importe quel système linéaire de la forme $AX = B$ avec A une matrice carré inversible.
Cette méthode est utile pour résoudre tout types de systèmes linéaires complexes de la forme $AX = B$ a condition que la matrice A soit carré et inversible.

Bibliographie

https ://old.i2m.univ-amu.fr/lic-maths/lib/exe/fetch.php ?media=ensmi5u4 :cours2.pdf
https ://fr.wikipedia.org/wiki/D