

TP Analyse Numérique 02



Dr. Nour El-houda
BOUCHERCHEM

Université de 20 Août 1955.
SKIKDA.

Faculté des Sciences.

Département de
mathématique.

Email : n.boucherchem@univ-skikda.dz

1.0
Mai 2025

Table des matières

I - Chapitre 01 : Résolution des systèmes linéaires	3
1. Rappel sur l'algèbre linéaire dans Matlab	4
1.1. Les vecteurs	4
1.2. Exercice :	4
1.3. Les matrices	5
1.4. Exercice :	5
2. Les méthodes directes	8
2.1. Méthode de Gauss	8
2.2. Méthode de décomposition LU	9
2.3. Méthode de Cholesky	10
3. Les méthodes itératives	12
3.1. La méthode de Jaccobi	12
3.2. La méthode de Gauss Seidel	13
3.3. La méthode de Relaxation	14

I Chapitre 01 : Résolution des systèmes linéaires

1. Rappel sur l'algèbre linéaire dans Matlab

1.1. Les vecteurs

💡 Fondamental

$n:m$	nombres de n à m par pas de 1
$n:p:m$	nombres de n à m par pas de p
$\text{linspace}(n,m,p)$	p nombres de n à m
$\text{length}(x)$	longueur de x
$\text{transpose}(x)$ ou x'	transposer un vecteur x
$x(i)$	ième coordonnée de x
$x(i_1:i_2)$	coordonnées i_1 à i_2 de x
$[x,y]$	concaténer les vecteurs x et y

1.2. Exercice :

1. Créer un vecteur ligne et un vecteur colonne.
2. Créer un vecteur contenant des entiers de 1 à 90 et un vecteur contenant des entiers de 10 à 100 de pas 5.
3. Comment générer un vecteur ligne contenant 4 valeurs également espacées entre 13 et 40 ?

Solution :

```

1 >> %1
2 >> v1 = [ 1 3 0 -1 5 ] %vecteur ligne 1*5
3 v2 = [ 2 ;4 ;8 ;-3 ;7 ;-2 ;1] %vecteur colonne
4
5 v1 =
6
7     1     3     0    -1     5
8
9
10 v2 =
11
12     2
13     4
14     8
15    -3
16     7
17    -2
18     1
19 >> %2
20 >> v3 = 5 : 9 % incrément 1 par défaut
21 v4 = 1 : 0.5 : 4 % le pas est 0.5
22
23 v3 =
24
25     5     6     7     8     9

```

```

26
27
28 v4 =
29
30    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
31
32 >> %3
33 >> v5=linspace(13,40 ,4) % un vecteur de 4 éléments de 13 à
34
35 v5 =
36
37    13    22    31    40

```

1.3. Les matrices

💡 Fondamental

<i>size(A)</i>	nombre de lignes et de colonnes de A
<i>diag(A)</i>	coefficient diagonaux de A
<i>A(i,j)</i>	élément ligne i et colonne j de A
<i>A(p,:)</i>	accéder à la p ième ligne
<i>A(:,n)</i>	accéder à la n ième colonnes
<i>zeros(m,n)</i>	matrice nulle de taille m,n
<i>ones(m,n)</i>	matrice de taille m,n dont tous les coefficients valent 1
<i>eye(n)</i>	matrice identité de taille n
<i>diag(x)</i>	matrice diagonale dont la diagonale est le vecteur x
<i>A'</i>	transposée de A
<i>inv(A)</i>	inverse de A
<i>det(A)</i>	déterminant de A
<i>A\b</i>	solution de $Ax = b$

1.4. Exercice :

Soient le vecteur colonne et la matrice suivants : $A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ -1 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 1 \end{pmatrix}, b = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$

1. Entrer ces données sous Matlab.
2. Calculer le déterminant et l'inverse de A.
3. Extraire l'élément (4, 4) de A ainsi la troisième colonne de A .
4. Résoudre le problème $Ax = b$.

Solution :

```

1 >> %1
2 >> A=[1 0 1 0; -1 1 0 -1; 1 0 -1 0; 0 1 -1 1]
3 b=[1 ; -1 ; 1 ; -1]
4
5 A =
6
7     1     0     1     0
8    -1     1     0    -1
9     1     0    -1     0
10    0     1    -1     1
11
12
13 b =
14
15     1
16    -1
17     1
18    -1
19
20 >> %2
21 >> D=det(A)
22 I=inv(A)
23
24 D =
25
26    -4
27
28
29 I =
30
31    0.5000         0    0.5000         0
32    0.5000    0.5000         0    0.5000
33    0.5000         0   -0.5000         0
34         0   -0.5000   -0.5000    0.5000
35
36 >> %3
37 >> A(4,4)
38 A(:,3)
39
40 ans =
41
42     1
43
44
45 ans =
46
47     1
48     0
49    -1
50    -1
51 >> %4
52 >> x=A\b
53
54 x =
55
56    1.0000
57   -0.5000
58         0
59   -0.5000

```


2. Les méthodes directes

2.1. Méthode de Gauss

Définition

La méthode de Gauss consiste à transformer le système $AX = b$ où A est une matrice carrée d'ordre n en un système linéaire de la forme $A'X = b'$ ayant la même solution que le système initial et tel que A' soit triangulaire supérieure.

L'algorithme de Gauss qui va triangulaires la matrice A se déroule de la façon suivante :

Pour k allant de 1 à $n-1$:

$$L_i = L_i - \frac{a_{ik}}{a_{kk}} L_k \quad \text{avec } i = k + 1 : n.$$

La solution du système suivant l'algorithme est :

$$\begin{cases} x_n = \frac{b_n}{a_{nn}} \\ x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}} \end{cases} \quad \text{avec } i = (n-1) : 1.$$

Code Matlab :

```
1 function [x,a]=gauss(a,b)
2 n=length(b);
3 for k=1:n-1
4     for i=k+1:n
5         m=a(i,k)/a(k,k);
6         a(i,:)=a(i,:)-m*a(k,:);
7         b(i)=b(i)-m*b(k);
8     end
9 end
10 x(n)=b(n)/a(n,n);
11 for i=n-1:-1:1
12     somme=sum(a(i,i+1:n).*x(i+1:n));
13     x(i)=(b(i)-somme)/a(i,i);
14 end
15 x=x';
16 end
```

Exemple

Soit le système linéaire $AX = b$, où

$$A = \begin{pmatrix} 2 & 3 & -1 \\ 4 & 4 & -3 \\ -2 & 3 & -1 \end{pmatrix}, b = \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix}.$$

- La résolution de système par la méthode de Gauss :

```
1 >> a=[2 3 -1; 4 4 -3; -2 3 -1]; b=[5; 3; 1];
```

```

2 >> [x,a]=gauss(a,b)
3
4 x =
5
6     1
7     2
8     3
9
10
11 a =
12
13     2     3    -1
14     0    -2    -1
15     0     0    -5

```

2.2. Méthode de décomposition LU

Définition

Soit le système linéaire $Ax = b$, la méthode consiste à décomposer la Matrice A en un produit de deux matrices L, U tel que

- L : une matrice triangulaire inférieure unitaire.
- U : une matrice triangulaire supérieure.

Donc la résolution du système $Ax = b$ revient à la résolution de deux systèmes triangulaire :

$$Ax = b \iff LUx = b \iff \begin{cases} Ly = b \\ Ux = y \end{cases}$$

La détermination des éléments de L et U cherchés se fait suivant l'algorithme général :

Pour $k = 1 : n$

$$\begin{cases} u_{kj} = a_{kj} - \sum_{m=1}^{k-1} l_{km}u_{mj} & \text{avec } k \leq j \leq n. \\ l_{ik} = \frac{a_{ik} - \sum_{m=1}^{k-1} l_{im}u_{mk}}{u_{kk}} & \text{avec } k+1 \leq i \leq n. \end{cases}$$

La solution du système suivant l'algorithme est :

$$\begin{cases} y_i = b_i - \sum_{j=1}^{i-1} l_{ij}y_j & \text{avec } 1 \leq i \leq n. \\ x_k = \frac{y_k - \sum_{j=i+1}^n u_{kj}u_{mk}x_j}{u_{kk}} & \text{avec } n \leq k \leq 1. \end{cases}$$

Code Matlab :

```

1 function [l,u,x]=decomposition(a,b)
2 n=length(b);
3 u=zeros(n);
4 l=zeros(n);
5 for i=1:n

```

```

6     u(i,i)=1;
7 end
8 for k=1:n
9     for i=k:n
10        s1=sum(l(i,1:k-1)*u(1:k-1,k));
11        l(i,k)=a(i,k)-s1;
12    end
13    for j=k+1:n
14        s2=sum(l(k,1:k-1)*u(1:k-1,j));
15        u(k,j)=(a(k,j)-s2)/l(k,k);
16    end
17 end
18 y=zeros(n,1);
19 x=zeros(n,1);
20 for i=1:n
21     s3=sum(l(i,1:i-1)*y(1:i-1));
22     y(i)=1/l(i,i)*(b(i)-s3);
23 end
24 for i=n:-1:1
25     s4=sum(u(i,i+1:n)*x(i+1:n));
26     x(i)=y(i)-s4;
27 end
28 end

```

Exemple

On va résoudre le même système par la méthode de la décomposition LU :

```

1 >> a=[2 3 -1; 4 4 -3; -2 3 -1]; b=[5; 3 ; 1];
2 >> [l,u,x]=decomposition(a,b)
3
4 l =
5
6     2     0     0
7     4    -2     0
8    -2     6    -5
9
10
11 u =
12
13    1.0000    1.5000   -0.5000
14         0    1.0000    0.5000
15         0         0    1.0000
16
17
18 x =
19
20     1
21     2
22     3

```

2.3. Méthode de Cholesky

Définition

La méthode de Cholesky consiste à décomposer la matrice symétrique définie positive A en un produit de deux matrices tel que

- R : une matrice triangulaire inférieure .

- R^t : la matrice transposée de la matrice R.

La résolution du système $Ax = b$ revient à la résolution de deux systèmes triangulaire :

$$Ax = b \iff RR^t x = b \iff \begin{cases} Ry &= b \\ R^t x &= y \end{cases}$$

Code Matlab :

```

1 >> A = [4, 1; 1, 3];
2 b = [1; 2];
3
4 % Vérification de la symétrie et positive-définie
5 if isequal(A, A') && all(eig(A) > 0)
6     % Décomposition de Cholesky
7     R = chol(A); % A = R'*R
8
9     % Résolution de R'*y = b
10    y = R' \ b;
11
12    % Résolution de R*x = y
13    x = R \ y;
14
15    % Affichage
16    disp('Solution par Cholesky :');
17    disp(x);
18 else
19     disp('A est pas symétrique définie positive. La décomposition de Cholesky ne s applique
    pas. ');
20 end
21 Solution par Cholesky :
22     0.0909
23     0.6364

```

```

1 >> A=[2 3 -1; 4 4 -3; -2 3 -1]; b=[5; 3 ; 1];
2 >>
3 % Vérification de la symétrie et positive-définie
4 if isequal(A, A') && all(eig(A) > 0)
5     % Décomposition de Cholesky
6     R = chol(A); % A = R'*R
7
8     % Résolution de R'*y = b
9     y = R' \ b;
10
11    % Résolution de R*x = y
12    x = R \ y;
13
14    % Affichage
15    disp('Solution par Cholesky :');
16    disp(x);
17 else
18     disp('A est pas symétrique définie positive. La décomposition de Cholesky ne s applique
    pas. ');
19 end
20 A est pas symétrique définie positive. La décomposition de Cholesky ne s applique pas.

```

3. Les méthodes itératives

3.1. La méthode de Jaccobi

Définition

Dans la méthode de Jaccobi, on va décomposer la matrice A sous la forme $A = M - N$, avec

- M une matrice facile à inverser (on va prendre dans cette méthode $M = D = (\text{diagonal de } A)$).
- $N = E + F$,
tel que
- E : c'est la partie inférieure avec des 0 sur la diagonal.
- F : c'est la partie supérieure avec des 0 sur la diagonal.

Alors

$$\begin{aligned} Ax = b &\iff (D - (E + F))x = b \\ &\iff Dx = (E + F)x + b \\ &\iff x = D^{-1}(E + F)x + D^{-1}b, \end{aligned}$$

donc

$$Ax = b \iff x^{(k+1)} = D^{-1}(E + F)x^{(k)} + D^{-1}b.$$

On a alors, pour $i = 1 : n$

$$\begin{cases} x^{(0)} & \text{connu,} \\ x_i^{(k+1)} & = \frac{-1}{a_{ii}} \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} + \frac{b_i}{a_{ii}}. \end{cases}$$

Code Matlab :

```
1 function [x, niter]=jaccobi(a, b, x0,nmax, tol)
2 n=length(b);
3 x=zeros(n,1);
4 for niter=1:nmax
5     for i=1:n
6         j=[1:i-1, i+1:n];
7         s=a(i,j)*x0(j);
8         x(i)=(b(i)-s)/a(i,i);
9     end
10 end
11 if norm(x-x0)<tol
12     return
13 end
14 end
15 x0=x;
16 end
17 end
```

Exemple

Résoudre le système $Ax=b$ par la méthode de Jaccobi où

$$A = \begin{pmatrix} 10 & -1 & 2 & -3 \\ 1 & 10 & -1 & 2 \\ 2 & 3 & 20 & -1 \\ 3 & 2 & 1 & 20 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 5 \\ -10 \\ 15 \end{pmatrix}$$

```

1 >> a=[10 -1 2 -3; 1 10 -1 2; 2 3 20 -1; 3 2 1 20]; b=[0; 5 ; -10; 15];
2 >> x0 = [0;0.5;-0.5; 3/4];
3 >> nmax=200; tol=0.001;
4 >> [x, niter]=jaccobi(a, b, x0,nmax, tol)
5
6 x =
7
8     0.3448
9     0.2718
10    -0.5404
11     0.6981
12
13
14 niter =
15
16     5
17

```

3.2. La méthode de Gauss Seidel

Définition

La méthode de Gauss Seidel est une modification de la méthode de Jaccobi qui consiste à utiliser pour chaque équation les composantes de $x^{(k+1)}$ déjà calculées, ceci conduit aux formules :

Pour $i = 1 : n$

$$\begin{cases} x^{(0)} & \text{connu,} \\ x_i^{(k+1)} & = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right). \end{cases}$$

Code Matlab :

```

1 function [x, niter]=gaussseidel(a, b, x0,nmax, tol)
2 n=length(b);
3 x=x0;
4 for niter=1:nmax
5     for i=1:n
6         s1=sum(a(i,1:i-1)*x0(1:i-1));
7         s2=sum(a(i,i+1:n)*x0(i+1:n));
8         x(i)=(b(i)-s1-s2)/a(i,i);
9     end
10 end
11 if norm(x-x0)<tol
12     return
13 end
14 x0=x;
15 end
16 end

```

Exemple

Résoudre le système $Ax = b$ par la méthode de Gauss-Seidel où

$$A = \begin{pmatrix} 10 & 1 & 1 \\ 2 & 10 & 1 \\ 2 & 2 & 10 \end{pmatrix}, b = \begin{pmatrix} 12 \\ 13 \\ 14 \end{pmatrix}.$$

```
1 >> a=[10 1 1; 2 10 1; 2 2 10]; b=[12 ; 13; 14];
2 >> x0 = [1.2;0;0];
3 >> nmax=80; tol=0.001;
4 >> [x, niter]=gaussseidel(a, b, x0,nmax, tol)
5
6 x =
7
8     1.0001
9     1.0001
10    1.0001
11
12
13 niter =
14
15     7
```

3.3. La méthode de Relaxation

Définition

La méthode de relaxation consiste à écrire le système sous la forme :

$$\begin{aligned} Ax = b &\iff \left(\frac{1}{w}D - E\right)x + \left(\frac{w-1}{w}D - F\right)x = b \\ &\iff \left(\frac{1}{w}D - E\right)x = \left(\frac{w-1}{w}D - F\right)x + b \end{aligned}$$

cela donne

Pour $i = 1 : n$

$$\begin{cases} x^{(0)} & \text{connu,} \\ x_i^{(k+1)} &= \frac{w}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} \right) + (1-w)x_i^{(k)}. \end{cases}$$

Où w un nombre réel non nul, appelé paramètre de la méthode de relaxation.

Code Matlab :

```
1 function [x, niter]=relaxation(a, b, x0,nmax, tol,w)
2 n=length(b);
3 x=x0;
4 for niter=1:nmax
5     for i=1:n
6         s1=sum(a(i,1:i-1)*x0(1:i-1));
7         s2=sum(a(i,i+1:n)*x0(i+1:n));
8         x(i)=(1-w)*x0(i)+w*(b(i)-s1-s2)/a(i,i);
9     end
```

```

10     end
11     if norm(x-x0)<tol
12         return
13     end
14     x0=x;
15 end
16 end

```

Exemple

Résoudre le système $Ax = b$ dans l'exemple de la méthode de Gauss-Seidel par la méthode de Relaxation.

```

1 >> a=[10 1 1; 2 10 1; 2 2 10]; b=[12 ; 13; 14];
2 >> x0 = [1.2;0;0];
3 >> nmax=80; tol=0.001;
4 >> w=0.5;
5 >> [x, niter]=relaxation(a, b, x0,nmax, tol,w)
6
7 x =
8
9     1.0007
10    0.9992
11    0.9999
12
13
14 niter =
15
16     12

```