

TP N° 6 : Résolution des systèmes linéaires

Dans ce TP, nous allons voir comment résoudre un système des équations linéaires de type :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \end{cases}$$

où : n est le nombre d'inconnues $x_1, \dots, x_n$ et m le nombre d'équations.

Représentation matricielle du système

Ce système peut s'exprimer sous forme matricielle : $\mathbf{AX} = \mathbf{B}$, Avec :

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{ et } B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

A est appelé matrice de coefficients et X le vecteur solution.

Résolution du système

Si $m=n$, A est une matrice carrée et si le déterminant de A est non nul, alors A est inversible :

La solution donc est : $\mathbf{X} = \mathbf{A}^{-1} \mathbf{AX} = \mathbf{A}^{-1} \mathbf{B}$.

Exemple

En utilisant Matlab, nous allons résoudre le système linéaire suivant : $\begin{cases} 2x_1 - x_2 = 2 \\ x_1 + x_2 = 5 \end{cases}$

```
>> A = [2 -1 ; 1 1]
A =
     2     -1
     1      1

>> B = [2 ; 5]
B =
     2
     5

>> X = inv(A)*B
X =
     2.3333
     2.6667

>> rats(X)           % rats : approximation rationnelle
ans =
     7/3
     8/3
```

Avec MATLAB, les systèmes d'équations linéaires peuvent se résoudre à l'aide de l'opérateur '\', comme suit:

```
>> A = [2 -1 ; 1 1] ;
>> B = [2 ; 5] ;
>> X =
    2.3333
    2.6667

>> rats(X)
ans =
    7/3
    8/3
```

L'opérateur anti-slash, '\', donne systématiquement le résultat.

Remarque

- Dans le cas où le système d'équations n'admet pas de solution : le résultat obtenu par $A \setminus B$ sera faux.
- Dans le cas d'une infinité de solutions, l'opérateur \ donnera une solution parmi les solutions possibles.

Une manière de savoir si un système possède réellement au moins une solution, il faut vérifier le théorème de Kronecker-Capelli :

Théorème de Kronecker-Capelli

Un système d'équations linéaires à n variables, de la forme $\mathbf{AX} = \mathbf{b}$, possède une solution *si et seulement si* le *rang de la matrice des coefficients A* est égal à celui de la matrice augmentée ($A_{\mathbf{b}}$).

S'il existe une solution X_0 , alors, l'ensemble des solutions forme un sous-espace affine de $\mathbb{R}^n$: $X_0 + \ker(A)$, sa dimension est $n - \text{rang}(A)$ (car $n = \dim(\ker(A)) + \dim(\text{Im}(A))$ et $\text{rang}(A) = \dim(\text{Im}(A))$). Dans ce cas :

- si $n = \text{rang}(A)$, la solution est unique ;
- sinon, il existe une infinité de solutions.

Exemples en Matlab

1. Exemple d'un système avec une solution unique

$$\begin{cases} 2x_1 - x_2 = 2 \\ x_1 + x_2 = 5 \end{cases}$$

```
>> A = [2 -1 ; 1 1] ;
>> B = [2 ; 5] ;
>> rank(A), rank([ A B ])
ans =
    2
ans =
    2
%on constate que : rang(A)= rang(A|b)= dim(R2)=2 --> solution unique
X = rats(A \ B)
X =
    7/3
    8/3
% pour s'assurer que la solution est vraie
>> A*(A \ B)
```

```

ans =
    2.0000
    5.0000
% attention A*X ne peut pas faire la faire parce que X est un char
% X est rats(A\B) c'est une chaîne de caractère qui représente la valeur
% de X sous forme rationnelle
>> A*X
ans =
    Columns 1 through 11
    32    32    32    32    32    32    54    47    51    32    32
    64    64    64    64    64    64   111    94   102    64    64
    Columns 12 through 14
    32    32    32
    64    64    64
%
```

2. Exemple d'un système avec ensemble infini de solutions

$$\begin{cases} 2x_1 + 3x_2 + 4x_3 = 4 \\ x_1 + x_2 + x_3 = 5 \end{cases}$$

```

>> A = [2 3 4 ; 1 1 1] ;
>> B = [4 ; 5] ;
>> r1 = rank(A), r2 = rank([ A B ])
r1 =
    2
r2 =
    2
%on constate que: rang(A)= rang(A|b)= 2= dim(R3)-1 --> dim(ker(A))=1
% --> solution : espace affine de dimension 1 (une droite)
>> X = rats(A\B)
X =
     8
     0
    -3
% le système donne une solution parmi l'ensemble infini de solution

%vérification de la solution
A*(A\B)
ans =
    4.0000
    5.0000
```

3. Exemple d'un système qui ne possède aucune solution

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 12 \\ 3x_1 + 2x_2 + x_3 = 15 \\ 3x_1 + 4x_2 + 7x_3 = 13 \\ 10x_1 + 9x_2 + 8x_3 = 17 \end{cases}$$

```

A = [1 2 3 ; 3 2 1; 3 4 7; 10 9 8] ;
B = [12 ; 15 ; 13; 17] ;
>> rank(A), rank([ A B ])
ans =
```

```

3
ans =
4
% rang(A)≠rang(A|B) --> d'après le théorème : pas de solution
>> X = A\B
X =

1.0887
-0.2527
1.5349
% le X ici n'est pas une solution vous pouvez vérifier que A*x ≠ B

A*X
ans =

5.1882
4.2957
13.0000
20.8925

```

Conclusion :

Avant d'appliquer la syntaxe $\text{inv}(A)*B$ ou bien $A \setminus B$, il faut étudier l'ensemble de solution du système linéaire en appliquant le théorème, sinon Matlab peut vous donner une solution erronée !!!.

Voici un script qui permet de vérifier l'existence et l'unicité de la solution

```

%vérification de l'existence et l'unicité de la solution

if rank(A)== rank([ A B ])
    s=size(A) ; %s contient le nombre d'equations et le nombre de variables
    if rank(A) == s(2) % s(2) c'est la dim de l'espace vectoriel
        disp('solution unique')
    else
        disp(['une infinite de solution : espace affine de dim : '
num2str(s(2)-rank(A))])
    end
else
    disp('pas de solution')
end

```

Fonction linsolve()

Soit le système linéaire (S) suivant :

$$(S): \begin{cases} x + y + 2z = 3 \\ x + 2y + z = 1 \\ 2x + y + z = 0 \end{cases}$$

La fonction linsolve peut résoudre le système :

```

>> A=[ 1 1 2; 1 2 1; 2 1 1];
>> b= [ 3 1 0]';
>> linsolve(A,b)
ans =

-1
0
2

```

Remarque

La fonction `linsolve` rencontre les mêmes problèmes que l'opérateur `\`, en effet ; la fonction renvoie une solution pour un système qui n'a pas de solution

```
%Exemple d'un système sans solution :
% x1 + 2x2 + 3x3 = 12
% 3x1 + 2x2 + x3 = 15
% 3x1 + 4x2 + 7x3 = 13
% 10x1 + 9x2 + 8x3 = 17
A = [1 2 3 ; 3 2 1; 3 4 7;10 9 8] ;
B = [12 ; 15 ; 13; 17] ;
>> rank(A), rank([ A B ])
ans =
     3
ans =
     4
>> linsolve(A,B)
ans =
    1.0887
   -0.2527
    1.5349
```

Function `mldivide()`

La fonction `mldivide` peut également résoudre ce système :

```
>> A=[ 1 1 2; 1 2 1; 2 1 1];
>> b= [ 3 1 0]';
>> mldivide(A,b)
ans =
   -1.0000
    0.0000
    2.0000
```

Remarque

La même remarque précédente pour la fonction `mldivide`

```
%Exemple d'un système sans solution :
% x1 + 2x2 + 3x3 = 12
% 3x1 + 2x2 + x3 = 15
% 3x1 + 4x2 + 7x3 = 13
% 10x1 + 9x2 + 8x3 = 17
A = [1 2 3 ; 3 2 1; 3 4 7;10 9 8] ;
B = [12 ; 15 ; 13; 17] ;
>> rank(A), rank([ A B ])
ans =
     3
ans =
     4
>> mldivide(A,B)

ans =
    1.0887
   -0.2527
    1.5349
```

Références

Ref1 : <https://www.mathworks.com/help/matlab/>

Ref2 : http://www.isir.upmc.fr/UserFiles/File/clady_homepage/L2/TP2.pdf

Ref3 : http://fractale.gecif.net/si/logiciels/matlab/matlab_ingenieur_oct2010.pdf

Annexe : Application en électricité

Résolution d'un circuit électrique

Étant donné le circuit de la figure 1, on souhaite calculer les valeurs efficaces et phases des courants lorsque :

$$U_g = 220 [V], \quad f_g = 50 [Hz], \quad R_1 = 10 [\Omega]$$

$$R_2 = 3 [\Omega], \quad C = 10 [\mu F], \quad L = 100 [mH]$$

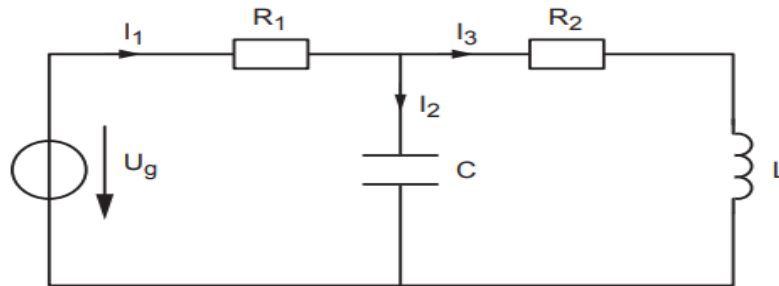


FIG. 1: Circuit électrique

Travail à faire :

1- Description du circuit

Pour résoudre ce circuit, il faut commencer par écrire les équations qui le décrivent. Nous appelons cette étape la modélisation mathématique du problème.

Essayer de modéliser ce problème

Description du circuit

Pour résoudre ce circuit, il faut bien entendu commencer par écrire les équations qui le décrivent. Considérant les courants I_1 , I_2 , I_3 circulant dans les 3 branches du circuit, celui-ci est complètement décrit par les équations suivantes :

$$\begin{aligned} U_g &= R_1 I_1 + \frac{1}{j\omega C} I_2 \\ 0 &= -\frac{1}{j\omega C} I_2 + (R_2 + j\omega L) I_3 \\ 0 &= I_1 - I_2 - I_3 \end{aligned}$$

équations que l'on peut récrire sous forme matricielle :

$$\begin{pmatrix} R_1 & \frac{1}{j\omega C} & 0 \\ 0 & -\frac{1}{j\omega C} & R_2 + j\omega L \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} U_g \\ 0 \\ 0 \end{pmatrix}$$

La solution s'obtient en multipliant à gauche les 2 membres de l'équation par l'inverse de la matrice décrivant le circuit :

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} R_1 & \frac{1}{j\omega C} & 0 \\ 0 & -\frac{1}{j\omega C} & R_2 + j\omega L \\ 1 & -1 & -1 \end{pmatrix}^{-1} \begin{pmatrix} U_g \\ 0 \\ 0 \end{pmatrix}$$

2- Implémentation en Matlab

Le calcul des courants par Matlab en interprétant les lignes de commande d'un script Matlab (dit aussi MFile ou bien fichier .m).

Les étapes principales pour écrire le code Matlab :

- Définir et initialiser les données
- Description matricielle du circuit
- Résolution du circuit
- Affichage des valeurs

Essayer d'écrire le code Matlab dans un fichier circuit.m en respectant les étapes citées, de le compiler

Implémentation en Matlab

Code matlab

```
% Calcul d'un circuit electrique
format compact ; format short g ;
% donnees
fg = 50 ; w = 2*pi*fg ; Ug = 220 ; % Ug = 220 Veff
R1 = 10 ; R2 = 3 ; L = 100e-3 ; C = 10e-6 ;

% rappel des equations du circuit
% R1 I1 + 1/jwC I2 + 0 I3 = Ug
% 0 I1 - 1/jwC I2 + (R2+jwL) I3 = 0
% I1 - I2 - I3 = 0

% description matricielle du circuit
Z = [ R1      , +1/(j*w*C) , 0
      0      , -1/(j*w*C) , R2+j*w*L
      1      , -1       , -1] ;
U = [Ug ; 0 ; 0] ; % vecteur colonne

% resolution du circuit
I = inv(Z) * U ;
Ieff = abs(I) ;
PhaseI = angle(I) * 180 / pi;

% affichage des valeurs
I
Ieff
PhaseI
% affichage sous forme matricielle
Courants = [I Ieff PhaseI]
```

Exemple d'exécution du script circuit.m

```
>> circuit
I =
2.1521 - 5.4723i
-0.1719 + 0.6235i
2.3240 - 6.0959i

Ieff =
5.8803
0.6468
6.5238

PhaseI =
-68.5319
```

3- Convertir le script en une fonction Matlab : fcircuit()

Avant de commencer, il faut tout d'abord préciser les arguments de la fonction fcircuit() et sa valeur de retour