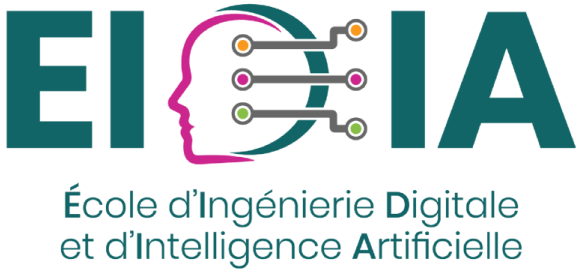
Université Euro Méditerranéenne de Fès
Ecole d'Ingénierie Digitale et d'Intelligence Artificielle (EIDIA)



Module: Calcul Formel sous Matlab

Chapitre 2:

Syntaxe de Condition et Boucle en Matlab

Pr : A. SLIMANI

2021/2022

Plan

1- Introduction

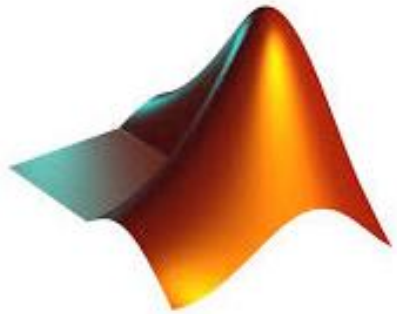
2- Entrées – Sorties

3- Instructions de contrôle : Les conditions

4- Instructions de contrôle : Les boucles

5- Boucles et conditions encapsulées

1. Introduction



Les instructions de contrôle permettent de gérer le flux d'exécution d'un programme, en fonction de certaines conditions.

Pour les instructions de conditions, une expression conditionnelle est déclarée : si l'expression est vraie, le groupe de commandes qui suit la condition est exécutée, sinon cette séquence de commande est ignorée.

Une boucle permet de répéter l'exécution d'une commande ou groupe de commandes plusieurs fois consécutivement.

2. Entrées – Sorties

Les entrées-sorties sont des commandes utilisées soit pour lire ou afficher des données dans la fenêtre de commande Matlab. Ces données qui peuvent être sous forme de différents formats.

Le **format** est une chaine de caractère qui précise comment les données doivent être affichées à l'écran. On utilise une syntaxe similaire à celle de langage C: $\%7.3f$

- 7 est le nombre total de chiffre, le point compris,
- 3 est le nombre de chiffre après la virgule,
- f spécificateur de type

Le tableau ci-dessous nous donne une indication sur les différents formats en Matlab:

Désignation	Description
$\%c$	Un seul caractère
$\%d$	Décimal signé
$\%e$	Notation exponentielle. (ex : 3.1415e+00)
$\%f$	Notation en virgule fixe
$\%g$	Format libre (zéros non significatifs ne s'affichent pas)
$\%o$	Représentation octale
$\%s$	Chaîne de caractères
$\%u$	Décimal non signé
$\%x$	Hexadécimal (lettres a-f)
$\%X$	Hexadécimal (lettres A-F)

A. Lecture

- **input** : c'est une commande qui demande à l'utilisateur de fournir des données. Sa syntaxe est sous forme :

variable = **input (** ' *Instruction demandée* ' **)**

L'instruction demandée s'affiche dans la fenêtre de commande, MATLAB attend que l'utilisateur saisisse une donnée au clavier, qui sera une valeur numérique ou une instruction. Une valeur numérique est directement affectée à la variable après un retour chariot qui provoque la fin de la saisie.

Si on souhaite de saisir une réponse de type chaîne de caractères on utilise la syntaxe :

variable = **input (** ' *Instruction demandée* ' , ' **s** ')

Exemple:

1. Exécuter le programme ci-dessous sous forme d'un nouveau script:

```
x=input('Donnez la valeur de x : ');

y=1+sqrt(x)
```

2. Aussi le programme ci-dessous:

```
Nom=input(' Votre nom et prénom ? \n','s');

Nom
```

Remarque: `\n` est utilisée pour retourner à une nouvelle ligne

B. Écriture

➢ Affichage simple

- **disp** : c'est une commande qui permet d'afficher un message sous forme de valeurs numériques ou de caractères. Sa syntaxe est sous forme:

disp (' *message* ')

On utilise fréquemment cette commande avec un tableau qui est une chaîne de caractères pour afficher un message.

Par exemple : **disp(' *Le résulta final de y est :* ').**

Exemple:

1. Donner le script qui demande de donner la valeur d'un nombre x, puis elle affiche le résultat $y = 1 + \frac{\sqrt{x.e^x}}{x+1}$ avec le message: **la valeur de y est :**

```
x=input('Donnez la valeur de x : ') ;
disp('la valeur de y est: ')
y=1+sqrt(x*exp(x))/(x+1)
```

➢ Impression dirigée par format

- **sprintf** est une commande qui permet l'impression de variables selon un modèle donné.

Un modèle d'édition se présente sous la forme du symbole pourcent **(%)** suivi d'indications permettant de composer le contenu du champ à imprimer, en particulier sa longueur en nombre de caractères.

Sa syntaxe est sous forme:

sprintf (**format** , variables)

- variables est le nom des variables à imprimer.
- format s'agit d'une chaîne de caractères contenant les modèles des variables à imprimer.

Exemple:

- Exécuter la commande ci-dessous soit sur la zone de commande ou sous forme d'un nouveau script:

```
sprintf('pi=%d  pi=%e  pi=%f  pi=%4.2f pi=%g  ', pi, pi,
pi, pi,pi)
```

1. Qu'est ce qu'il fait ce programme ?
2. Qu'est ce que vous remarquez ?

Plan

3. Instructions de contrôle : les conditions

A. Les structures en **if** (condition logique)

- L'instruction **if**, permet d'exécuter du code si une condition est vraie. Cette instruction se termine toujours par un **end**.

Le sens de **if** reste le même pour les différents langages.

La structure if – end :

```
if  condition
    instructions
end
```

Exemple:

Exécuter le programme ci-dessous sous forme d'un nouveau script.

```
a=input('Donnez moi une valeur de a : ');
b=6;
   if  a<b
      c=a+b
   end
```

La structure if – else – end :

```
if condition
    instructions
else
    instructions
end
```

Exemple:

Exécuter le programme ci-dessous sous forme d'un nouveau script.

```
a=input('Donnez moi une valeur de a : ') ;
b=6;
    if  a<b
        c=a+b
    else
        c=a*b
    end
```

La structure if – elseif – else – end :

```
if condition
    instructions
elseif
    instructions
else
    instructions
end
```

Le nombre de conditions **elseif** n'est pas limité.

Exemple:

Exécuter le programme ci-dessous dans un nouveau script.

```
a=input('Donnez moi la valeur de a: ') ;
b=6;
    if  a<b
        c=a+b
    elseif a>b
        c=a*b
    else
        c=a
    end
```

Exercice d'application:

Soit l'équation de second degré désignée par:

$$\boldsymbol{ax^2 + bx + c = 0}$$

- Créer un script qui calcule les racines de cette équation en demandant d'insérer les valeurs de $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{c}$.

Remarque: Il faut afficher un message qui exprime s'il y a une seule ou deux solutions réelles ou complexes avant d'afficher le résultat.

Solution : Méthode 1

```
a=input('Donner la valeur de a: ');
b=input('Donner la valeur de b: ');
c=input('Donner la valeur de c: ');
delta=b^2-4*a*c;
if delta<0
        disp('Les solutions sont des nombres complexes')
     x1=(-b+sqrt(delta))/(2*a)
     x2=(-b-sqrt(delta))/(2*a)
    elseif delta==0
         disp('La solution est:')
         x=-b/(2*a)
    else
        disp('Il y a deux solutions x1 et x2: ')
     x1=(-b+sqrt(delta))/(2*a)
     x2=(-b-sqrt(delta))/(2*a)
end
```

Solution : Méthode 2

```
a=input('Donner la valeur de a: ');
b=input('Donner la valeur de b: ');
c=input('Donner la valeur de c: ');
delta=(b^2)-4*a*c;
if delta<0
      x1=(-b+sqrt(delta))/(2*a);
      x2=(-b-sqrt(delta))/(2*a);
    Solution=sprintf(' Il y a deux solutions complexes: x1=%f+%fi et
x2=%f+%fi',real(x1), imag(x1), real(x2), imag(x2))
    elseif delta==0
         x=-b/(2*a);
         Solution=sprintf('La solution est x1=%f',x)
    else
     x3=(-b+sqrt(delta))/(2*a);
     x4=(-b-sqrt(delta))/(2*a);
     Solution=sprintf('Il y a deux solutions réelles x3=%f et x4=%f',x3,x4)
end
```

B. Les structures en switch - case (jonction conditionnelle)

C'est une instruction de branchement conditionnel.

Sa syntaxe est :

```
switch  variable d'entrée
    case    choix 1
            Commande
    case    choix 2
            Commande
      ⋮
    otherwise
            Commande
end
```

switch – case:

- C’est une instruction conditionnelle à choix multiples.
- Elle sert à comparer la valeur d’une **variable d’entrée** par rapport à plusieurs cas, où la **variable d’entrée** doit être **un scalaire** ou une **chaîne de caractère**.
- Dès que le choix correspond à l’un des cas, le traitement correspondant est exécuté, et les cas suivants sont ignorés.
- Si aucun choix n’est validé, le bloc qui suit **otherwise** est exécuté.

Remarque:
Pour le cas des choix, si le choix est sous forme d’une chaine de caractère, il doit être entre deux apostrophes après **case** : case ' choix '

Exemple d'application :

Exécuter le programme ci-dessous sous Matlab:

```
disp('1: Doctorat ')
disp('2: Master ')
disp('3: Licence ')
msg=input('Donnez moi un choix (1,2 ou 3): ');
switch msg
    case 1
        disp('Passez vers le batiment A')
    case 2
         disp('Passez vers le batiment B')
    case 3
        disp('Passez vers le batiment C')
    otherwise
         disp('Désolé: Accès refusé')
end
```

Exercice d’application :

Sous un nouveau script:

- Exécuter le programme qui vous demande si vous avez le code ou non (il faut répondre par Oui ou Non).
- Si vous répondez par **Oui**, il vous demande de le tapez, après il affiche le code avec le message: Votre code (***le code tapé***) est bien vérifié,
- Si vous répondez par **Non**, il vous demande de le demander,
- Si vous n’avez pas répondu par Oui ou Non, il vous demande de répondre par Oui ou Non.

Solution:

```
M=input(' Avez-vous le code ? Oui/Non:  ','s');
switch M
    case 'Oui'
        disp('Tapez-le')
        y=input('le code:','s');
    code=sprintf('Ton code (%s) est bien vérifié',y)
    case 'Non'
        disp('Demandez-le !')
    otherwise
        disp('Prière de répondre par Oui ou Non')
end
```

Exemple de la fonction modulo :

- C’est une fonction qui nous permet de calculer le reste de la division euclidienne d’un nombre par un autre.
- C’est une fonction qu’on peut l’appeler sous la forme **mod** en matlab,
- Le modulo est très pratique pour savoir si un nombre est divisible par un autre ou non:

  Si le nombre est divisible: le reste de la division euclidienne est égal à 0.

- C’est une fonction qu’on l’utilise souvent dans une condition pour tester la divisibilité d’un nombre par un autre.

Exemple d'application 1:

Exécuter le programme ci-dessous sous Matlab:

```
x = input('Entrez une valeur : ');
if mod(x,5) == 0
 disp('x est divisible par 5')
else
 disp('x est indivisible par 5')
end
```

Exercice d’application :

Créer un script qui teste la parité d’un nombre.

- ❑ Si le nombre est pair, il va afficher le message: **le nombre est pair**.
- ❑ Si le nombre est impair, il va afficher le message: **le nombre est impair**.

Solution:

```
x = input('Entrez une valeur : ');
  if mod(x,2) == 0
     disp('le nombre x est pair')
  else
     disp(' le nombre x est impair')
  end
```

Plan

4. Instructions de contrôle : les boucles

Une boucle permet de répéter l'exécution d'une commande ou un groupe de commandes plusieurs fois consécutivement.

Il y a deux types de boucle dans Matlab :

La boucle for – end :

```
for    i = debut : pas : fin
       instruction
end
```

La boucle while – end :

```
while     condition
instruction exécutée si la condition est vraie
end
```

Exemple d'application : La boucle for – end

Soit une suite de Fibonacci : $\boldsymbol{f_i = f_{i-1} + f_{i-2}}$, avec $f_1 = 0$ et $f_2 = 1$.

Pour calculer les 5 premiers termes de cette suite, on peut utiliser le programme suivant basé sur la boucle **for-end**:

```
f(1) = 0;
f(2) = 1;
 for i = 3:5
f(i) = f(i-1) + f(i-2);
end
f
```

Exercice d'application :

Soit la suite : $\boldsymbol{U_n = 3n - 2}$

avec, $\boldsymbol{U_1 = 1}$ et sa raison $\boldsymbol{r_1 = 3}$.

- En se basant sur la boucle **for-end,** donner dans un seul programme :
  - ✓ L'instruction qui calcule et affiche les 5 premiers termes de cette suite sous le message: ***Les 5 premiers termes de la suite Un sont :***
  - ✓ et l'instruction qui vérifie et affiche si le type de cette suite est une suite arithmétique ou non.

Solution:

```
U(1) = 1;
r(1)=3;
 for i = 2:5
U(i) = 3*i -2;
r(i)=U(i)-U(i-1);
 end
disp('Les 5 premiers termes de la suite Un sont: ')
U
disp('La raison de la suite Un est: ')
r
if r==3
    disp('Cette suite est arithmétique')
else
    disp('Suite non arithmétique')
end
```

Exemple d'application : La boucle while – end

Soit l'équation suivante: $a_n = n^{(n-1)}$

- Pour calculer les 4 premiers termes de cette équation, on peut utiliser le programme suivant basé sur la boucle **while-end**:

```
n = 1;
while n<5

 a(n) = n^(n-1);
 n = n+1;
end
a
```

Le compteur (**n**) est ajouté ici, qui doit être initialisé et incrémenté pour assurer la condition d'arrêt de la boucle **while**.

Exercice d’application :

Soit l’équation suivante: $$y_n = a^n + b^{(n-1)} + c^{(n-2)}$$

- En se basant sur la boucle **while-end,** donner le programme qui calcule les 6 premiers valeurs de cette équation, en demandant d’insérer les valeurs de a, b et c.

Solution:

```
a=input('Donner la valeur de a: ');
b=input('Donner la valeur de b: ');
c=input('Donner la valeur de c: ');
n = 1;
while n<7

 y(n) = a^n+b^(n-1)+c^(n-2);
 n = n+1;
End
disp('Les termes de y sont: ')
y
```

Plan

1- Introduction

2- Entrées – Sorties

3- Instructions de contrôle : les conditions

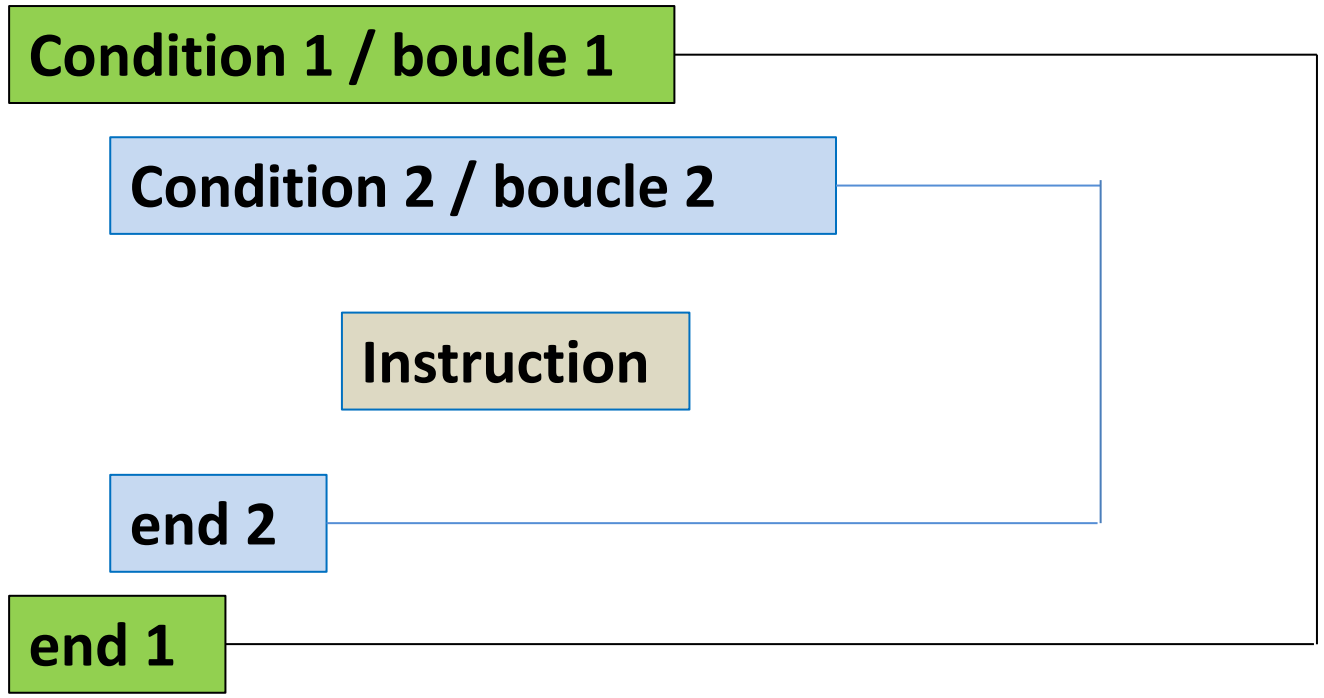
4- Instructions de contrôle : les boucles

5- Boucles et conditions encapsulées

5. Boucles et conditions encapsulées

Les conditions et boucles peuvent être encapsulées. C'est-à-dire qu'une boucle et/ou une condition peut se situer à l'intérieur d'une autre boucle et/ou une condition (aucune limite de nombre).

La structure encapsulée :



Exemple d'application :

Création d'une matrice de dimension : $m \; x \; n$

```
m = input('Donner le nombre de lignes m: ');
n = input('Donner le nombre de colonnes n:');


for i=1:m
        for j=1:n
            M(i,j) = 4;
        end
    end
M
```

Chaque fois que (**i**) augmente de 1 (boucle), la boucle encapsulée (**j**) est exécutée n fois.

Exercice d’application :

- Créer un script qui affiche une matrice de dimension $\boldsymbol{m\ x\ n}$ avec tous les éléments égalent à la somme du numéro de la ligne et de la colonne, sachant que sa diagonale égale à une valeur $\boldsymbol{x}$ demandée au début.

Remarque:

Il faut demander de taper le nombre de ligne $\boldsymbol{m}$, de colonnes $\boldsymbol{n}$ et la valeur $\boldsymbol{x}$. Si $\boldsymbol{m}$ ou $\boldsymbol{n}$ est inférieur ou égale à zéro, un message s’affiche qui demande d’insérer des valeurs entières non nulles avec les commandes pour insérer $\boldsymbol{m}$ et $\boldsymbol{n}$.

Solution:

```
 m = input('Donner le nombre de lignes m: ');
while m<=0
    m = input('Prière de donner une valeur positive de m: ');
end
n = input('Donner le nombre de colonnes n:');
while  n<=0
    n = input('Prière de donner une valeur positive de n: ');
end
x = input('Donner la valeur de x: ');
for i=1:m
   for j=1:n
       if i==j
          M(i,j)=x;
       else
            M(i,j) = i+j;
       end
   end
end
disp('La matrice proposée est :'), M
```

FIN