Logiciel de simulation "MATLAB"

Logiciel de simulation

Université de Tlemcen

Département GEE

Dr Saidi Farah

Janvier 2024

Table des matières

Objectifs

L'objectif du Chapitre 3 est de familiariser les apprenants avec l'écriture de scripts **MATLAB** et l'utilisation de fonctions pour automatiser des tâches et résoudre des problèmes. Ce chapitre vise également à introduire les concepts de base du graphisme en **MATLAB**, permettant aux apprenants de créer des graphiques et des visualisations pour représenter leurs données de manière efficace.

Les principaux objectifs du Chapitre 3 incluent :

- Compréhension des scripts **MATLAB**
- Manipulation des variables et des opérations arithmétiques
- Utilisation des structures de contrôle
- Définition et utilisation de fonctions
- Introduction au graphisme

Programmer sous MATLAB

I

Dans ce chapitre, nous explorerons les concepts fondamentaux des scripts et des fonctions en MATLAB. Les scripts et les fonctions constituent les briques de base de la programmation MATLAB, permettant aux utilisateurs de créer et d'organiser des ensembles d'instructions pour effectuer des tâches spécifiques. Nous découvrirons comment écrire des scripts pour automatiser des tâches répétitives, ainsi que comment définir des fonctions pour encapsuler des blocs de code réutilisables. En comprenant ces concepts, vous serez en mesure de développer des programmes plus complexes et plus efficaces dans MATLAB.

1. Scripts MATLAB

Introduction aux scripts MATLAB

Un script **MATLAB** est un fichier contenant une série d'instructions **MATLAB.** Il est utilisé pour exécuter des tâches répétitives ou pour automatiser des processus.

Pour créer un script **MATLAB**, vous pouvez ouvrir l'éditeur de texte intégré à **MATLAB** en cliquant sur l'icône appropriée dans l'interface utilisateur, puis écrire vos instructions **MATLAB** dans l'éditeur et les enregistrer dans un fichier avec l'extension **".m"**.

Types de fichiers-M

- **Script M-Files** : Ces fichiers contiennent une séquence d'instructions **MATLAB** pour effectuer une tâche ou une série de tâches spécifiques. Vous pouvez les exécuter en les ouvrant et en cliquant sur le bouton d'exécution, ou en utilisant la commande **run** ou simplement en tapant leur nom dans la console.
- **Fonction M-Files** : Les fonctions M-Files sont utilisées pour encapsuler un ensemble d'instructions dans une fonction réutilisable. Vous pouvez les appeler avec des arguments pour effectuer des calculs et renvoyer des résultats.

Exemple

```
a = 5;

b = 3;

sum = a + b;

disp(['La somme de ', num2str(a), ' et ', num2str(b), ' est ', num2str(sum)]);
```

Structure d'un script

Un script **MATLAB** est généralement organisé de manière logique, avec des sections distinctes pour différentes parties du code.

Les commentaires sont utilisés pour rendre le code plus lisible et pour expliquer ce que fait chaque partie du script. Les commentaires commencent par le symbole "%" et sont ignorés par **MATLAB** lors de l'exécution du script.

Exemple

```
% Script pour calculer la moyenne de trois nombres

% Déclaration des variables

a = 10;

b = 15;

c = 20;

% Calcul de la moyenne

moyenne = (a + b + c) / 3;

% Affichage du résultat

disp(['La moyenne de ', num2str(a), ', ', num2str(b), ' et ', num2str(c), ' est ', num2str(moyenne)]);
```

Manipulation des variables

Les variables dans **MATLAB** sont des conteneurs pour stocker des données. Vous pouvez déclarer une variable en lui attribuant une valeur.

Les opérations arithmétiques de base telles que l'addition, la soustraction, la multiplication et la division peuvent être effectuées sur des variables et des constantes dans un script **MATLAB**.

Exemple

```
% Script pour calculer l'aire d'un rectangle

longueur = 10;

largeur = 5;

aire = longueur * largeur;

disp(['L\'aire du rectangle est ', num2str(aire)]);
```

Structures de contrôle

Les boucles (comme **for** et **while**) sont utilisées pour répéter des blocs de code un certain nombre de fois ou jusqu'à ce qu'une condition soit remplie.

Les instructions conditionnelles (comme **if-else**) sont utilisées pour exécuter un bloc de code si une condition est vraie, sinon un autre bloc de code est exécuté.

Ces structures de contrôle permettent de contrôler le flux d'exécution du programme et d'exécuter des instructions en fonction de certaines conditions ou de manière répétitive.

Exemple

```
% Script pour afficher les carrés des nombres de 1 à 5

for i = 1:5

carre = i^2;

disp(['Le carré de ', num2str(i), ' est ', num2str(carre)]);

end
```

Conseil

1- Le script doit avoir une extension de la forme **« .m »**.

- Dans l'appellation des scripts, Il faut éviter : D'utiliser les caractères désignant un opérateur spécifique à Matlab tel que « -», « ; », « * »,...etc.
- D'utiliser le nom d'une fonction **MATLAB** telle que : sqrt, sin, cos,...etc.
- D'utiliser deux mots séparées (par exemple : program st, program 1)

2- Divisez votre script en sections logiques avec des commentaires descriptifs pour rendre le code plus facile à comprendre et à maintenir.

3- Utilisez des noms de variables significatifs et évitez les noms génériques comme "x" ou "y" pour rendre votre code plus clair et plus compréhensible.

4- Assurez-vous de comprendre comment fonctionnent les structures de contrôle et testez votre code avec différentes valeurs pour vous assurer qu'il produit les résultats attendus.

Remarque : Gestion de la mémoire

Les variables créées dans un fichier-M sont stockées dans l'environnement **MATLAB** global. Veillez à gérer correctement la portée des variables pour éviter les conflits et les fuites de mémoire.

2. Fonctions MATLAB

Définition : Définition de fonctions MATLAB

Les fonctions MATLAB sont des blocs de code autonomes qui acceptent des entrées, effectuent des calculs ou des opérations, et renvoient des résultats.

Les fonctions sont utilisées pour organiser le code de manière modulaire, permettant la réutilisation du code et l'encapsulation de la logique de calcul. Elles sont essentielles pour la création de bibliothèques de fonctions personnalisées.

- Utilisez le mot-clé **function** suivi du nom de la fonction et de ses arguments entre parenthèses.
- Ajoutez des commentaires descriptifs à l'intérieur de la fonction pour expliquer son fonctionnement.

Exemple

```
function resultat = addition(a, b)
% Cette fonction calcule la somme de deux nombres
resultat = a + b;
end
% exemple
x = 3;
y = 7;
z = addition(x, y);
disp(['La somme de x et y est : ' num2str(z)]);
```

Conseil

- Nommez vos fonctions de manière descriptive pour indiquer clairement leur fonction.
- Évitez d'utiliser des noms de fonctions **MATLAB** existantes pour éviter les conflits de noms.
- Organisez vos fonctions dans des fichiers séparés pour une meilleure gestion et réutilisation du code.
- Testez vos fonctions avec différents jeux de données pour vous assurer qu'elles produisent les résultats attendus.

3. Instructions de contrôle

Définition

Les instructions de contrôle en **MATLAB** sont des structures qui permettent de contrôler le flux d'exécution d'un programme en fonction de certaines conditions ou de répéter des blocs de code un certain nombre de fois.

3.1. Boucle for

Une boucle **for** est une structure de contrôle qui permet de répéter un ensemble d'instructions un certain nombre de fois, en fonction d'une valeur de compteur.

Syntaxe

```
for variable = valeurs
instructions
end
```

Exemple

```
%Cette boucle affiche la valeur de la variable i de 1 à 5
for i = 1:5
disp(['Valeur de i : ', num2str(i)]);
end
```

Boucle for emboîtés

Les boucles for emboîtées en **MATLAB** sont utilisées lorsque vous avez besoin d'itérer sur plusieurs ensembles de données ou de réaliser des opérations imbriquées.

Syntaxe

for variable1 = valeurs1

for variable2 = valeurs2

instructions

end

end

Exemple

Supposons que vous ayez une matrice 2D (par exemple, une matrice de taille 3x3) et que vous vouliez parcourir chaque élément de la matrice et afficher ses coordonnées ainsi que sa valeur. Vous pouvez le faire en utilisant des boucles for emboîtées comme suit :

matrice = [1, 2, 3; 4, 5, 6; 7, 8, 9];

% Boucle pour parcourir les lignes de la matrice

for i = 1:size(matrice, 1)

% Boucle pour parcourir les colonnes de la matrice

for j = 1:size(matrice, 2)

% Afficher les coordonnées et la valeur de chaque élément

disp(['Coordonnées : (', num2str(i), ',', num2str(j), ') - Valeur : ', num2str(matrice(i, j))]);

end

end

Dans cet exemple, la boucle externe parcourt les lignes de la matrice (i va de 1 à 3), tandis que la boucle interne parcourt les colonnes de la matrice (j va également de 1 à 3). À chaque itération des boucles, les coordonnées (i, j) de l'élément actuel sont affichées, ainsi que sa valeur.

3.2. Boucle while

- Une boucle **while** est une structure de contrôle qui répète un ensemble d'instructions tant qu'une condition donnée est vraie.
- La boucle **while** continue à s'exécuter tant que la condition est vraie. Si la condition devient fausse, la boucle s'arrête.
- Les boucles **while** sont utilisées lorsque le nombre d'itérations n'est pas connu à l'avance, mais dépend d'une condition.

Syntaxe

while condition

```
instructions
end
```

Exemple

```
x = 0;
while x < 5
disp(['Valeur de x : ', num2str(x)]);
x = x + 1;
end
```

Cette boucle affiche la valeur de la variable x tant qu'elle est inférieure à 5.

3.3. Instruction conditionnelle if-else

L'instruction **if-else** est utilisée pour exécuter un bloc de code si une condition spécifiée est vraie, sinon un autre bloc de code est exécuté.

L'instruction conditionnée if

L'instruction conditionnée **if** permet de prendre des décisions dans un programme en fonction de l'évaluation d'une condition.

Syntaxe

```
if condition
% Instructions à exécuter si la condition est vraie
else
% Instructions à exécuter si la condition est fausse
end
```

Si la condition est vraie, les instructions sous le bloc **if** sont exécutées. Sinon, les instructions sous le bloc **else** (optionnel) sont exécutées.

Exemple

```
x = 10;
if x > 5
disp('x est supérieur à 5.')
else
disp('x n\'est pas supérieur à 5.')
end
```

if-elseif

Dans certains cas, vous pouvez avoir plusieurs conditions à évaluer en utilisant l'instruction **elseif** (ou **else if**) pour gérer différentes branches conditionnelles

Syntaxe

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

Exemple

```
x = 10;
if x < 5
disp('x est inférieur à 5');
elseif x == 5
disp('x est égal à 5');
else
disp('x est supérieur à 5');
end
```

3.4. Instructions break et continue

- L'instruction break est utilisée pour sortir d'une boucle prématurément.
- L'instruction continue est utilisée pour passer à l'itération suivante d'une boucle.

Exemple

```
for i = 1:10
if i == 5
continue; % Passe à l'itération suivante si i est égal à 5
end
disp(['Valeur de i : ', num2str(i)]);
if i == 8
break; % Sort de la boucle si i est égal à 8
end
```

```
end
```

Cette boucle affiche les valeurs de i de 1 à 10, en ignorant 5 et en sortant de la boucle lorsque i est égal à 8.

* *

*

Dans ce chapitre, nous avons exploré en détail les concepts des scripts et des fonctions en MATLAB. Nous avons appris que les scripts sont des fichiers contenant un ensemble d'instructions MATLAB à exécuter séquentiellement, tandis que les fonctions sont des blocs de code encapsulés pouvant accepter des entrées et retourner des sorties. Nous avons également étudié les différentes façons d'appeler des fonctions, de passer des arguments et de gérer les sorties. En maîtrisant ces concepts, vous êtes maintenant mieux équipé pour écrire des programmes MATLAB plus robustes et plus modulaires, améliorant ainsi votre efficacité et votre productivité en tant que développeur MATLAB.

Activité d'apprentissage

II

1. Exercice

Quelle est la principale différence entre un script et une fonction en MATLAB ?

- ○ Une fonction peut accepter des arguments en entrée, tandis qu'un script ne le peut pas.
- ○ Les scripts et les fonctions sont identiques en **MATLAB**.
- ○ Un script peut retourner une valeur, tandis qu'une fonction ne le peut pas

2. Exercice

Comment appelle-t-on un fichier MATLAB contenant un ensemble d'instructions à exécuter ?

- ☐ Fonction
- ☐ Script
- ☐ programme

3. Exercice

Combien de fois une boucle for s'exécutera-t-elle si elle est définie comme suit : for i = 1:5 ?

4. Exercice

Quel est le mot-clé utilisé pour définir une fonction en MATLAB ?

5. Écriture de Scripts

1. Écrire un script **MATLAB** qui calcule la somme des entiers de 1 à 100 et affiche le résultat.
2. Écrire un script qui génère les premiers termes d'une suite mathématique.
3. Créer une fonction **MATLAB** qui calcule la factorielle d'un nombre donné.

6.
Exercice :

Série de TD

Exercice 1 :

Créez un script **MATLAB** qui affiche "Bonjour, MATLAB !" à l'aide de la fonction disp.

Exercice 2 :

Écrivez une fonction **MATLAB** appelée **calculer_moyenne** qui prend un vecteur de nombres en entrée et retourne leur moyenne. Testez cette fonction avec différents vecteurs.

Exercice 3 :

Écrivez une fonction **MATLAB** appelée **calculer_factorielle** qui prend un entier positif **n** en entrée et retourne sa factorielle. Utilisez une boucle **while** pour effectuer le calcul.

Exercice 4 :

Écrivez un script MATLAB qui crée une matrice 3x3 avec des valeurs aléatoires comprises entre 1 et 10, puis utilisez une double boucle **for** pour afficher la matrice ligne par ligne.