Chapitre 4
Représentations graphiques

1. Introduction

Nous avons vu dans les chapitres précédents que les tableaux pouvaient contenir des données et qu'il est possible de les manipuler, d'effectuer des opérations arithmétiques et logiques dessus, ou de les extraire.

L'analyse et l'interprétation des données sont une tâche très difficile, surtout pour les grands tableaux. Les ingénieurs et les techniciens utilisent les graphiques pour rendre l'information plus facile à comprendre.

Avec un graphique, il est aisé d'identifier les tendances, d'observer l'évolution des données, selon un axe ou un critère, et d'isoler les points particuliers qui peuvent être des erreurs de mesure ou de calcul.

MATLAB possède une bibliothèque de fonctions et de commandes pour élaborer toutes sortes de représentations graphiques, de la plus simple à la plus sophistiquée. Il permet de tracer très facilement n'importe quelle donnée à tout moment.

2. Graphisme 2D

2.1 Introduction

Le graphique linéaire est le type de courbe le plus utilisé. Il nécessite deux vecteurs de longueur égale, l'un qui précise les abscisses de chaque point, et l'autre qui précise les ordonnées. Si l'on dispose d'une plage de données stockées dans deux vecteurs ou dans un tableau à deux dimensions, on peut représenter ces données sous forme d'un graphique 2D.

Un graphique ou une courbe 2D est une représentation des données. Elle est composée d'un ensemble de points, chacun défini par une abscisse et une ordonnée. Ces points peuvent être reliés ou non par des segments de droite, pour donner l'illusion d'une courbe continue.

Sous MATLAB, la liste des fonctions relatives aux graphiques 2D est accessible via `help graph2d` et `help specgraph`.

2.2 Courbe simple

2.2.1 La commande plot

Une fois que les vecteurs abscisse et ordonnée (les valeurs de x et y) ont été définis, MATLAB permet de créer facilement des courbes, en utilisant la commande suivante :

Fonction de traçage `plot` - Syntaxe

```
plot(Vx,Vy)
```

où :

– `Vx` : le vecteur des abscisses.

– `Vy` : le vecteur des ordonnées.

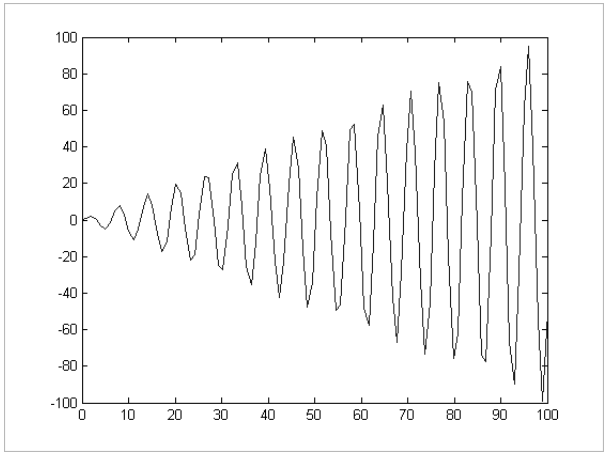
Les paramètres de la fonction `plot` sont des vecteurs de même longueur. Une fenêtre nommée `figure` s'affiche automatiquement lorsque la fonction `plot` est exécutée.

Courbe simple `plot` - Exemple n° 1

Écrire un script MATLAB qui permet de tracer la fonction `x sin(x)` sur l'intervalle [0, 100].

```
% courbe_simple_plot_1
close all; clear all; clc;

Vx = linspace(0, 100);
Vy = Vx .* sin(Vx);
plot(Vx, Vy);
```

Résultat :



Courbe simple `plot` - Exemple n° 2

Supposons que l'on a un ensemble de données contenant le temps et la distance parcourue pour un véhicule. On a mesuré les distances dans un intervalle entre 0 et 18 minutes avec un pas de 2 minutes. Écrire le script MATLAB qui permet de saisir les données et de représenter graphiquement cette série de valeurs.

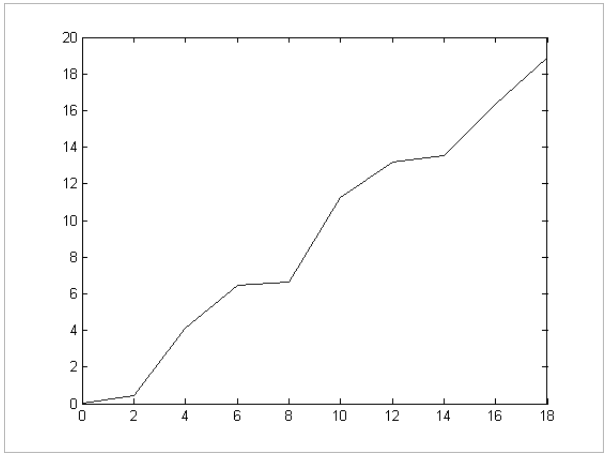
```
% courbe_simple_plot_2
close all; clear all; clc;

Vx = [0:2:18];
Vy = input('Donner les valeurs des distances : \n');

disp('=== Représentation graphique des données ===')
plot(Vx, Vy);
```

Résultat :

Donner les valeurs des distances

[0 0.44 4.15 6.49 6.65 11.23 13.19 13.56 16.33 18.84]

=== Représentation graphique des données ===



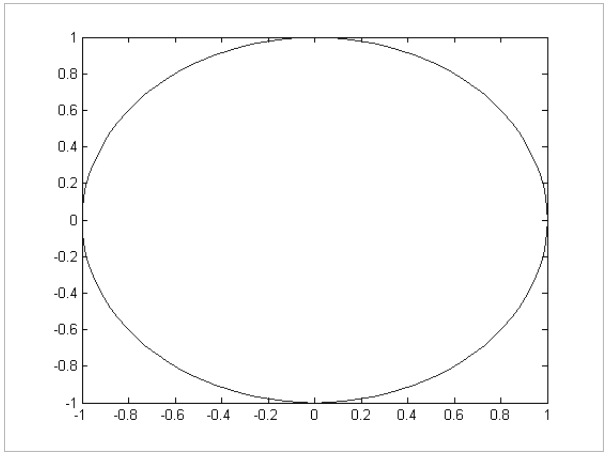
La fonction `plot` nécessite les valeurs d'abscisse et d'ordonnée. Logiquement, les abscisses (x) sont toujours un ensemble des valeurs. Grâce à MATLAB, ces abscisses peuvent être une fonction de x plutôt que x lui-même. En d'autres termes, il est donc tout aussi facile de tracer des graphiques paramétrés.

Courbe simple `plot` - Exemple n° 3

Écrire un script MATLAB qui permet de tracer une courbe, dont les abscisses sont `cos(x)` et les ordonnées sont `sin(x)` sur l'intervalle [0,2π].

```
% courbe_simple_plot_3
close all; clear all; clc;

x = 0 : 2*pi/100 : 2*pi;
Vx = cos(x);
Vy = sin(x);
plot(Vx, Vy);
```

Résultat :



Remarque

Il faut que les deux vecteurs des valeurs (abscisses et ordonnées) aient la même longueur ; sans cela, la commande `plot` entraînera une erreur explicite.

Plusieurs courbes sur le même graphique

Il est possible de tracer plusieurs courbes sur le même graphique. La syntaxe est la suivante :

Fonction de traçage plusieurs courbes `plot` - Syntaxe

```
plot(Vx1,Vy1, Vx2,Vy2,..., Vxn,Vyn)
```

où :

– `Vxi` : le vecteur d'abscisses de la ième fonction.

– `Vyi` : le vecteur d'ordonnées de la ième fonction.

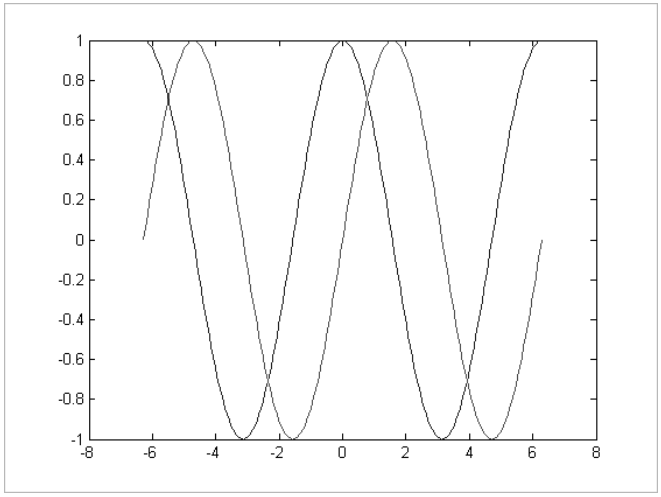
Si nous voulons que les graphiques aient une apparence distincte, nous emploierons des options de couleurs et/ou de styles de ligne distincts après chaque paire de vecteurs x, y.

Courbe simple `plot` - Exemple n° 4

Écrire un script MATLAB qui permet un graphique, contenant les courbes des fonctions `cos(x)` et `sin(x)` sur l'intervalle [-2π,2π].

```
% courbe_simple_plot_4
close all; clear all; clc;

x = -2*pi : 2*pi/100 : 2*pi;
plot(x, cos(x), x, sin(x));
```

Résultat :



2.2.2 La commande fplot

MATLAB propose une autre variante de traçage des courbes 2D, c'est la fonction `fplot`. Cette commande permet de tracer le graphe d'une fonction mathématique (fonction prédéfinie par MATLAB ou fonction définie par l'utilisateur) sur un intervalle donné. La syntaxe est :

Fonction de traçage `fplot` - Syntaxe

```
fplot('nom_fonction', [x_min, x_max])
```

où :

– `nom_fonction` : le nom de la fonction

– `x_min` : valeur initiale de l'intervalle

– `x_max` : valeur finale de l'intervalle

La commande `fplot` trace la fonction entre les limites indiquées. Cette fonction est de la forme `y = f(x)`, où `x` est un vecteur contenant les abscisses et `y` est un vecteur de même format (dimension et nombre d'éléments) que `x`. Le vecteur `y` contient les valeurs de la fonction pour toutes valeurs de `x`.