UNIVERSITE IBN ZOHR Année Universitaire : 2019-2020

Ecole Nationale des Sciences Appliquées – Agadir Filières : ENSA 1

**Graphique des fonctions**

**Avec Matlab**

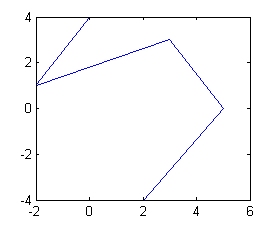
Matlab offre un puissant système de visualisation qui permet la présentation et l’affichage graphique des données d’une manière à la fois efficace et facile.

Dans cette partie du cours, nous allons présenter les principes de base indispensables pour dessiner des courbes en Matlab.

1. **La fonction plot :**

La fonction ***plot*** est utilisable avec des vecteurs ou des matrices. Elle trace des lignes en reliant des points de coordonnées définies dans ses arguments, et elle à plusieurs formes :

* ***Si elle contient deux vecteurs de la même taille comme arguments*** : elle considère les valeurs du premier vecteur comme les éléments de l’axe X (les abscisses), et les valeurs du deuxième vecteur comme les éléments de l’axe Y (les ordonnées).



Exemple :

**>> A = [2, 5, 3,-2,0]**

A =

2 5 3 -2 0

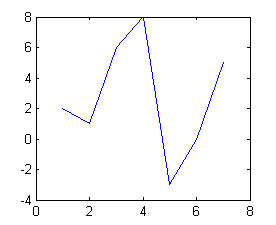
**>> B = [-4, 0, 3, 1,4]**

B =

-4 0 3 1 4

**>> plot(A,B)**

* ***Si elle contient un seul vecteur comme argument :*** elle considère les valeurs du vecteur comme les éléments de l’axe Y (les ordonnées), et leurs positions relatives définirons l’axe X (les abscisses).



Exemple :

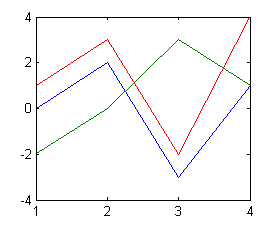
**>> V = [2, 1, 6, 8,-3, 0,5]**

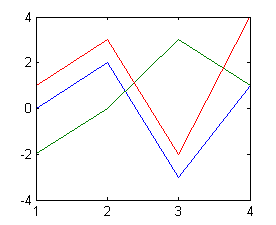
V =

2 1 6 8 -3 0 5

**>> plot(V)**

* ***Si elle contient une seule matrice comme argument :*** elle considère les valeurs de chaque colonne comme les éléments de l’axe Y, et leurs positions relatives (le numéro de ligne) comme les valeurs de l’axe X. Donc, elle donnera plusieurs courbes (une pour chaque colonne).





Exemple :

**>> M = [0 -2 1;2 0 3;-3 3 -2;1 1 4]**

M =

0 -2 1

2 0 3

-3 3 -2

1 1 4

**>> plot(M)**

* ***Si elle contient deux matrices comme arguments :*** elle considère les valeurs de chaque colonne de la première matrice comme les éléments de l’axe X, et les valeurs de chaque colonne de la deuxième matrice comme les valeurs de l’axe Y.

Exemple :

**>> K = [1 1 1;2 2 2; 3 3 3;4 4 4]**

K =

1 1 1

2 2 2

3 3 3

4 4 4

**>> M = [0 -2 1;2 0 3;-3 3 -2;1 1 4]**

M =

0 -2 1

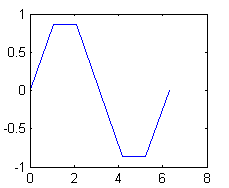
2 0 3

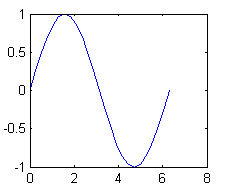
-3 3 -2

1 1 4

**>> plot(K,M**)

Il est évident que plus le nombre de coordonnées augmente plus la courbe devienne précise. Par exemple pour dessiner la courbe de la fonction *y = sin(x)* sur [0, 2] on peut écrire :





Pas =

Pas =

La première figure

**>> x = 0:pi/3:2\*pi;**

**>> y = sin(x);**

**>> plot(x,y)**

La deuxième figure

**>> x = 0:pi/12:2\*pi;**

**>> y = sin(x);**

**>> plot(x,y)**

1. **Modifier l’apparence d’une courbe :**

Il est possible de manipuler l’apparence d’une courbe en modifiant la couleur de la courbe, la forme des points de coordonnées et le type de ligne reliant les points.

Pour cela, on ajoute un nouveau argument (qu’on peut appeler un marqueur) de type chaine de caractère à la fonction ***plot*** comme ceci :

**plot (x, y, ’marqueur’)**

Le contenu du marqueur est une combinaison d’un ensemble de caractères spéciaux rassemblés dans le tableau suivant :

|  |  |  |  |
| --- | --- | --- | --- |
| Couleur de la courbe | | Représentation des points | |
| ***le caractère*** | ***son effet*** | ***le caractère*** | ***son effet*** |
| **b** ou **blue** | courbe en bleu | **.** | un point **.** |
| **g** ou **green** | courbe en vert | **o** | un cercle ⏺ |
| **r** ou **red** | courbe en rouge | **x** | le symbole **x** |
| **c** ou **cyan** | entre le vert et le bleu | **+** | le symbole **+** |
| **m** ou **magenta** | en rouge violacé vif | **\*** | une étoile **\*** |
| **y** ou **yellow** | courbe en jaune | **s** | un carré ⏹ |
| **k** ou **black** | courbe en noir | **d** | un losange ⧫ |
| Style de la courbe | | **v** | triangle inferieur ⏷ |
| ***le caractère*** | ***son effet*** | **^** | triangle supérieur ⏶ |
| **-** | en ligne plein | **<** | triangle gauche ⏴ |
| **:** | en pointillé | **>** | triangle droit ⏵ |
| **-.** | en point tiret | **p** | pentagramme ★ |
| **--** | en tiret | **h** | hexagramme 🟋 |

**Exemple :**

Essayons de dessiner la fonction *y = sin(x)* pour x = [0 ... 2π] avec un pas = π/6.

**>> x = 0:pi/6:2\*pi;**

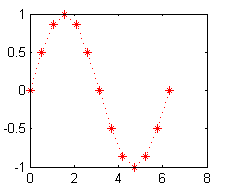
**>> y = sin(x);**

En changeant le marqueur on obtient des résultats différents, et voici quelques exemples :

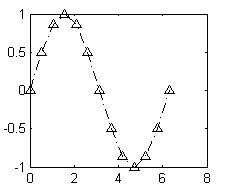
Couleur rouge, en pointillé et avec des étoiles

Couleur noire, en point tiret et avec des rectangles sup

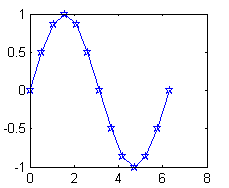
Couleur bleu, en ligne plein et avec des pentagrammes



**plot(x, y,'r:\*')**



**plot(x, y,'black-.^')**



**plot(x, y,'pb-')**

1. **Annotation d’une figure :**

Dans une figure, il est préférable de mettre une description textuelle aidant l’utilisateur à comprendre la signification des axes et de connaitre le but ou l’intérêt de la visualisation concernée.

Il est très intéressant également de pouvoir signaler des emplacements ou des points significatifs dans une figure par un commentaire signalant leurs importances.

* Pour donner un titre à une figure contenant une courbe nous utilisons la fonction ***title*** comme ceci :

**>> title('titre de la figure')**

* Pour donner un titre pour l’axe vertical des ordonnées y, nous utilisons la fonction ***ylabel*** comme ceci :

**>> ylabel('Ceci est l''axe des ordonnées Y')**

* Pour donner un titre pour l’axe horizontal des abscisses x, nous utilisons la fonction ***xlabel*** comme ceci :

**>> xlabel('Ceci est l''axe des abscisses X')**

* Pour écrire un texte (un message) sur la fenêtre graphique à une position indiquée par les coordonnées ***x*** et ***y***, nous utilisons la fonction ***text*** comme ceci :

**>> text(x, y, 'Ce point est important')**

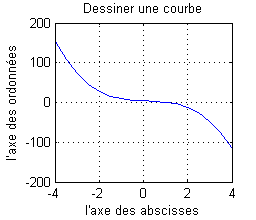
* Pour mettre un texte sur une position choisie manuellement par la souris, nous utilisons la fonction ***gtext***, qui a la syntaxe suivante :

**>> gtext('Ce point est choisi manuellement')**

* Pour mettre un quadrillage (une grille), utilisez la commande ***grid*** (ou ***grid on***). Pour l’enlever réutiliser la même commande **grid** (ou ***grid off***).

**Exemple :**

Dessinons la fonction : *y = -2x3+x2-2x+4* pour x varie de -4 jusqu’à 4, avec un pas = 0.5.



**>> x = -4:0.5:4;**

**>> y = -2\*x.^3+x.^2-2\*x+4;**

**>> plot(x,y)**

**>> grid**

**>> title('Dessiner une courbe')**

**>> xlabel('l''axe des abscisses')**

**>> ylabel('l''axe des ordonnées')**

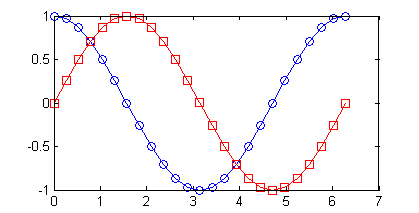
1. **Dessiner plusieurs courbes dans la même figure :**

Par défaut en Matlab, chaque nouveau dessin avec la commande ***plot*** efface le précédent. Pour forcer une nouvelle courbe à coexister avec les précédentes courbes, Il existe au moins trois méthodes :

**4.1** **La commande *hold***

La commande ***hold*** (ou ***hold*** ***on***) active le mode ‘préservation des anciennes courbes’ ce qui permette l’affichage de plusieurs courbes dans la même figure. Pour annuler son effet il suffit de réécrire ***hold*** (ou ***hold off***).

Par exemple pour dessiner la courbe des deux fonctions *cos(x)* et *sin(x)* dans la même figure, on peut écrire :



**>> x=0:pi/12:2\*pi;**

**>> y1=cos(x);**

**>> y2=sin(x);**

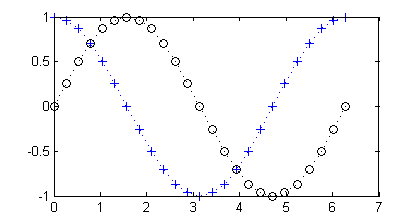
**>> plot(x,y1,'b-o')**

**>> hold on**

**>> plot(x,y2,'r-s')**

**4.2** **Utiliser plot avec plusieurs arguments**

On peut utiliser plot avec plusieurs couples (x,y) ou triples (x ,y, ‘marqueur’) comme arguments. Par exemple pour dessiner les mêmes fonctions précédentes on écrit :



**>> x=0:pi/12:2\*pi;**

**>> y1=cos(x);**

**>> y2=sin(x);**

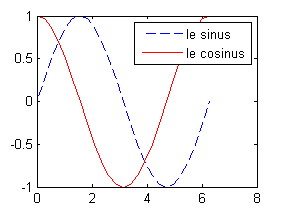
**>> plot(x,y1,'b:+',x,y2,'k:o')**

**4.3** **Utiliser des matrices comme argument de la fonction plot**

Dans ce cas on obtient plusieurs courbes automatiquement pour chaque colonne (ou parfois les lignes) de la matrice. On a déjà présenté ce cas plus auparavant.

Après pouvoir parvenir à mettre plusieurs courbes dans la même figure, il est possible de les distinguer en mettant une légende indiquant les noms des ces courbes.

Pour cela, on utilise la fonction ***legend***, comme illustre l’exemple suivant qui dessine les courbes des deux fonctions *sin(x)* et *cos(x)* :



**>> x=0:pi/12:2\*pi;**

**>> y1=sin(x);**

**>> y2=cos(x);**

**>> plot(x,y1,'b--',x,y2,'-r')**

**>> legend('le sinus','le cosinus')**

Il est possible de déplacer la légende (qui se situe par défaut dans le coin supérieur droit) en utilisant la sourie avec un glisser-déposer.

1. **Manipulation des axes d’une figure:**

Matlab calcule par défaut les limites (le minimum et le maximum) des axes X et Y et choisie automatiquement le partitionnement adéquat. Mais il est possible de contrôler l’aspect des axes via la commande ***axis***.

Pour définir les limites des axes il est possible d’utiliser cette commande avec la syntaxe suivante :

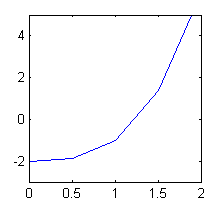
***axis ( [ xmin xmax ymin ymax ] )*** Ou ***axis ( [ xmin,xmax,ymin,ymax ] )***

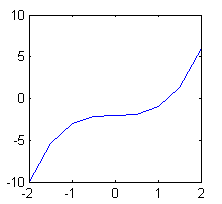
Avec : ***xmin*** et ***xmax*** définissent le minimum et le maximum pour l’axe des abscisses.

***ymain*** et ***ymax*** définissent le minimum et le maximum pour l’axe des ordonnées.

Pour revenir au mode d’affichage par défaut, nous écrivons la commande : ***axis auto***

**Exemple :**





f(x) = x3-2

**>> x = -2:0.5:2;**

**>> y = x.^3-2;**

**>> plot(x,y)**

**>> axis auto**

**>> axis([0,2,-3,5])**

Parmi les autres options de la commande ***axis***, nous présentons les suivantes :

* Pour rendre la taille des deux axes identique (la taille et non le partitionnement), nous utilisons la commande ***axis square***. Elle est nommée ainsi car elle rend l’aspect des axes tel un carré.
* Pour rendre le partitionnement des deux axes identique nous utilisons la commande ***axis equal***.
* Pour revenir à l’affichage par défaut et annuler les modifications nous utilisons la commande ***axis auto***.
* Pour rendre les axes invisibles nous utilisons la commande ***axis off***. Pour les rendre visibles à nouveau nous utilisons la commande ***axis on***.