Cours
Outils de programmation pour les mathématiques
-Partie 02-

Nous avons vu jusqu'à présent comment utiliser MATLAB pour effectuer des commandes ou pour évaluer des expressions en les écrivant dans la ligne de commande (Après le prompt >>), par conséquent les commandes utilisées s'écrivent généralement sous forme d'une seule instruction (éventuellement sur une seule ligne).

Cependant, il existe des problèmes dont la description de leurs solutions nécessite plusieurs instructions, ce qui réclame l'utilisation de plusieurs lignes.

Une collection d'instructions bien structurées visant à résoudre un problème donnée s'appelle un programme. Dans cette partie du cours, on va présenter les mécanismes d'écriture et d'exécution des programmes en MATLAB.

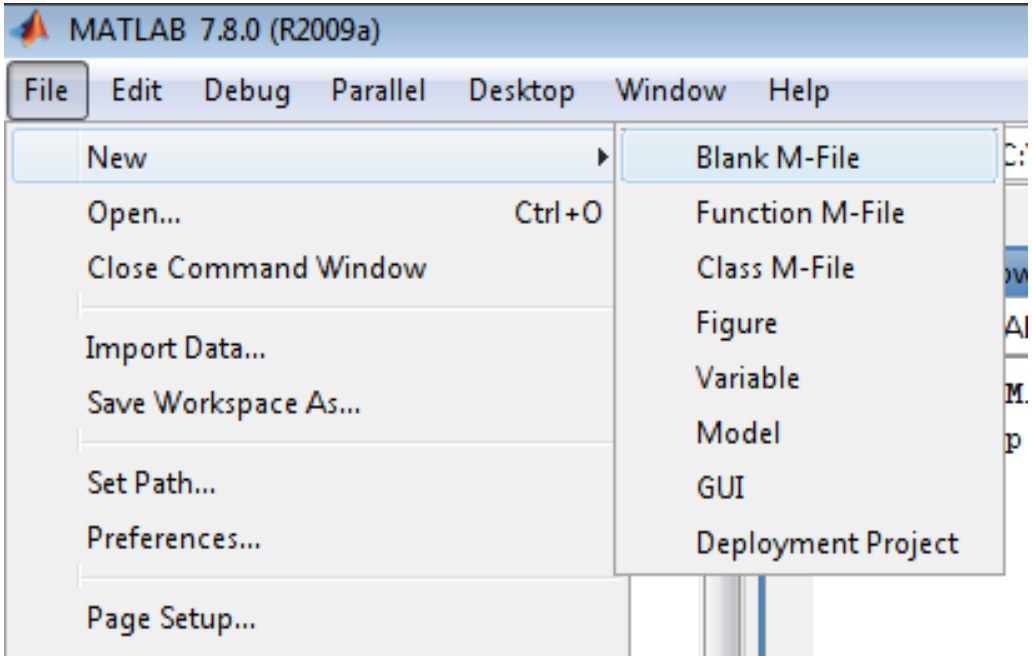
MATLAB est un langage de programmation puissant aussi bien qu'un environnement de calcul interactif. Les fichiers qui contiennent un programme dans le langage MATLAB ont l'extension ".m" (en anglais, on les appellent les M-files). On crée des M-files en utilisant un éditeur de texte.

Matlab possède son propre éditeur de texte qu'on ouvre :

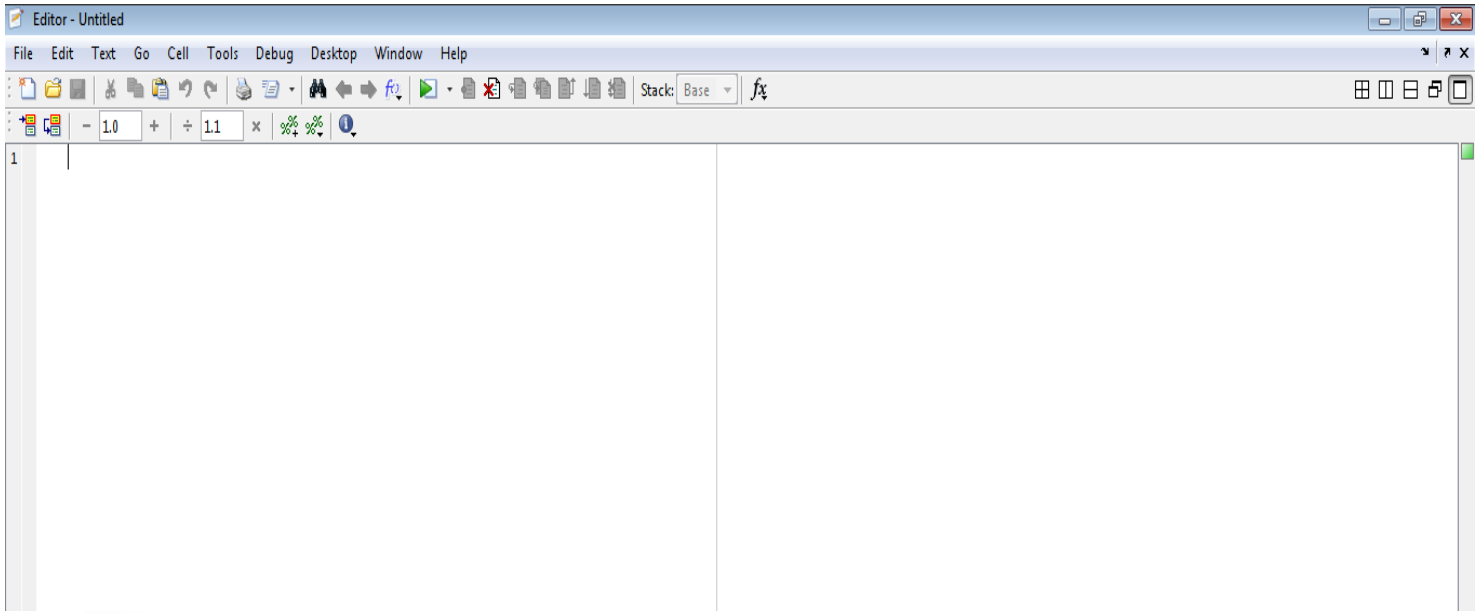
- à partir de la barre d'outils en cliquant sur l'icône  , ou
- en tapant la commande **>> edit**, ou
- à partir du menu fichier en cliquant sur

File→New→ Blank M-file.

MATLAB/ Programmation Matlab



Dans tous les cas une fenêtre d'édition comme celle-ci va apparaitre :



Il y a deux types de **M-File** : les scripts et les fonctions.

Scripts

Un script est une suite d'instructions MATLAB. Si le script est écrit dans un fichier de nom **myscript.m** on l'exécute dans la fenêtre de commande en tapant son nom (>> myscript) puis valider. son exécution est équivalente à la saisie des instructions une par une dans la fenêtre de commande.

Il est en effet beaucoup plus simple de modifier des instructions dans un fichier à l'aide d'un éditeur de texte que de retaper un ensemble d'instructions MATLAB dans la fenêtre de commande.

Il est important de commenter abondamment un programme. En MATLAB un commentaire commence par le symbole "**%**" et occupe le reste de la ligne.

Lors de l'écriture de script la communication avec l'utilisateur est importante.

MATLAB offre la possibilité de communication avec l'utilisateur grâce aux entrées/sorties de données qui peuvent se faire à travers plusieurs manières différentes, dans cette introduction on ne s'intéressera qu'à la communication par clavier.

Pour lire une valeur donnée par l'utilisateur, il est possible d'utiliser la commande input, qui a la syntaxe suivante :

```
variable = input ('une phrase indicative')
```

Quand MATLAB exécute une telle instruction, La phrase indicative sera affichée à l'utilisateur en attendant que ce dernier entre une valeur.

```
>> A = input ('Entrez un nombre entier : ')
   Entrez un nombre entier : 5
   A =
      5
```

```
>> B = input ('Entrez un vecteur ligne : ')
   Entrez un vecteur ligne : [1:2:8,3:-1:0]
   B =
      1     3     5     7     3     2     1     0
```

Pour l'affichage, on a déjà vu que MATLAB peut afficher la valeur d'une variable en tapant seulement le nom de cette dernière, mais avec cette méthode, MATLAB écrit le nom de la variable puis le signe (=) suivie de la valeur de variable.

On peut utiliser la fonction **disp**, qui a la syntaxe : ***disp(objet)***, pour afficher uniquement la valeur de l'objet. La valeur de l'objet peut être un nombre, un vecteur, une matrice, une chaine de caractères ou une expression.

On signale qu'avec un vecteur ou une matrice vide, **disp** n'affiche rien.

```
>> disp(A)      % Afficher la valeur de A sans 'A = '
     5
>> disp(A);     % Le point virgule n'a pas d'effet
     5
>> disp(B)      % Afficher le vecteur B
     1     3     5     7     3     2     1     0
```

Fonction

Une fonction Matlab est un fichier d'extension '.m' qui calcule des variables de sorties (arguments) en utilisant des arguments d'entrée. Une fonction commence par une ligne de définition ayant la syntaxe suivante :

```
function [ output_arguments ] = function_name( input_arguments )
```

Il faut faire attention à ce que le nom de la fonction 'function_name' soit le même que le nom du fichier dans lequel la fonction est enregistrée.

```
function [ output_arguments ] = function_name( input_arguments )
```

Le fichier doit impérativement commencer par la déclaration function. Suit entre crochets les variables de sortie de la fonction, puis le symbole =, suivie par le nom de la fonction et enfin les variables d'entrée entre parenthèses. Si la fonction ne possède qu'une seule variable de sortie, les crochets sont inutiles.

L'appel de la fonction peut se faire à travers la saisie de son nom et des arguments correspondants.

Pour utiliser une fonction, il n'est pas nécessaire d'utiliser les mêmes noms de variables d'entrée et de sortie que ceux utilisés dans le fichier FUNCTION lui-même.

À la différence des fichiers SCRIPT, les variables à l'intérieur d'un fichier FUNCTION ne sont pas disponibles à l'extérieur. On dit qu'elles sont locales.

```
function y = fonc(x)
% Création de la fonction f(x)=x*x
% x =  scalaire ou vecteur
% y = scalaire ou vecteur de sortie

y = x.^2;
```

```
» fonc(2)
» b = fonc(5)
» x
??? Undefined function or variable x.
```

Mentionnons l'existence de commandes permettant d'évaluer une chaîne de caractères comme une commande Matlab. La commande **eval** s'utilise ainsi :

```
>> comd='s1=sin(pi/2)'

>> eval(comd)
```

MATLAB/ Programmation/ Contrôle

Les structures de contrôle de flux sont des instructions permettant de définir et de manipuler l'ordre d'exécution des tâches dans un programme. Elles offrent la possibilité de réaliser des traitements différents selon l'état des données du programme, ou de réaliser des boucles répétitives pour un processus donnée.

MATLAB compte huit structures de contrôle de flux à

Savoir:

- if
- switch
- for
- while
- continue
- break
- try - catch
- return

L'instruction if

L'instruction **if** est la plus simple et la plus utilisée des structures de contrôle de flux. Elle permette d'orienter l'exécution du programme en fonction de la valeur logique d'une condition

Si la condition est évaluée à ***vrai***, les instructions entre le **if** et le **end** seront exécutées, sinon elles ne seront pas (ou si un **else** existe les instructions entre le **else** et le **end** seront exécutées).

MATLAB/ Programmation/ Contrôle

```
if condition
     %%bloc d'instructions
End
```

```
if (condition)
      ensemble d'instructions 1
else
      ensemble d'instructions 2
end
```

S'il est nécessaire de vérifier plusieurs conditions au lieu d'une seule, on peut utiliser des clauses **elseif** pour chaque nouvelle condition, et à la fin on peut mettre un **else** dans le cas ou aucune condition n'a été évaluée à ***vrai***. Voici donc la syntaxe générale :

```
if condition1
    %bloc d'instructions
elseif condtion2
    %bloc d'instruction
elseif condition3
    %bloc d'instructions
else
    %bloc d'instructions
end
```

L'instruction switch

L'instruction **switch** exécute des groupes d'instructions selon la valeur d'une variable ou d'une expression. Chaque groupe est associé a une clause **case** qui définie si ce groupe doit être exécuté ou pas selon l'égalité de la valeur de ce **case** avec le résultat d'évaluation de l'expression de **switch**. Si tous les case n'ont pas étés acceptés, il est possible d'ajouter une clause **otherwise** qui sera exécutée seulement si aucun ca se n'est exécuté.

MATLAB/ Programmation/ Contrôle

```
switch expression
    case cas1, %bloc d'instructions
    case cas2, %bloc d'instructions
    case cas3, %bloc d'instructions
    otherwise %bloc d'instructions
end
```

L'instruction *for*

Boucle : ***for indice = i_début : pas : i_fin***

end

La boucle '**for**' permet une exécution répétitive d'un bloc d'instruction pour un nombre prédéfini de fois, elle possède la syntaxe suivante :

```
for i = indice_début : pas : indice_fin
    %%bloc d'instructions
end
```

Dans MATLAB il est d'usage d'initier l'indice dans les boucles à '1' et non pas à '0' pour éviter les erreurs lors de la manipulation des vecteurs et matrices puisque l'appel des premiers éléments de ces derniers se fait à travers l'indice '1'.

L'instruction while

Boucle : ***while condition***

end

La boucle 'while' permet de répéter un bloc d'instruction tant qu'une condition prédéfinie est vraie, sa syntaxe est la suivante :

```
while condition
    %%bloc d'instructions
End
```

MATLAB/ Programmation/ Contrôle

Un programme	Une fonction

`a = input('Entrez un nombre positif: ');`
`x = a/2;`
`precision = 6;`
`for i = 1:precision`
`  x = (x + a ./ x) / 2;`
`end`
`disp(x)`
	`function r = racine(nombre)`
	`  r = nombre/2;`
	`  precision = 6;`
	`  for i = 1:precision`
	`  r = (r + nombre ./ r) / 2;`
	`  end`

L'exécution :
`>> racine` ↵
`Entrez un nombre positif: 16` ↵
`4`
	L'exécution :
	`>> racine(16)` ↵
	`ans =`
	`4`

on ne peut pas écrire des expressions tel que `>> 2 * racine + 4` ✗	on peut écrire sans problème des expressions comme : `>> 2 * racine(x) + 4` ✓

MATLAB/ Graphique

Partant du principe qu'une image vaut mieux qu'un long discours, MTLAB offre un puissant système de visualisation qui permet la présentation et l'affichage grphique des données d'une manière à la fois efficace et facile.

La commande de base pour un graphique bidimensionnel (Matrice ou Vecteur) sur MATLAB est la commande 'plot', elle s'écrit :

```
plot(valeur_X, valeur_Y, 'option_de_style')
```

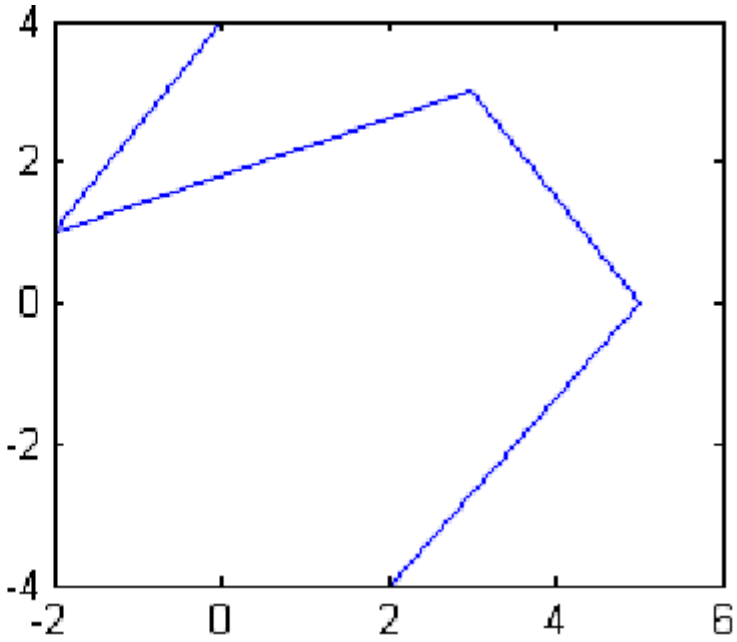
Elle trace des lignes en relient des points de coordonnées définies dans ces arguments

La fonction **Plot** a plusieurs formes:

1- 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 :

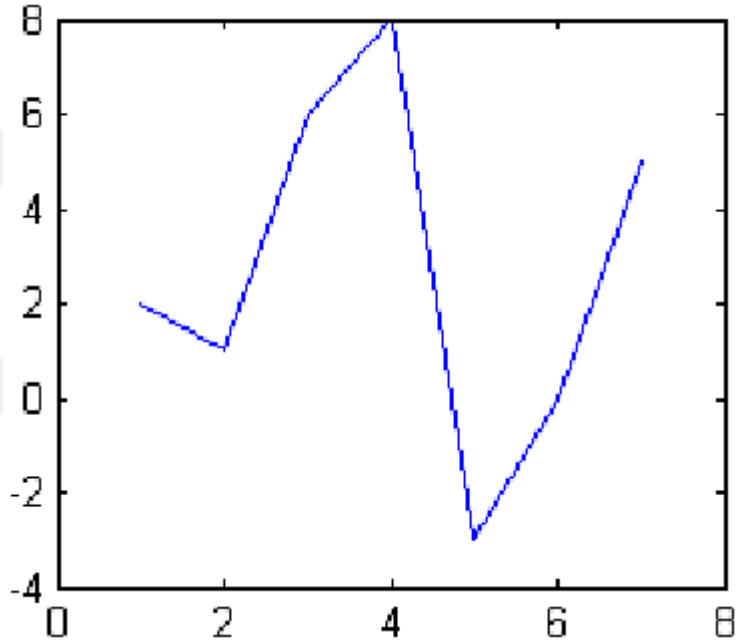
```
>> A = [2, 5, 3,-2,0]
A =
   2     5     3    -2     0

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

>> plot(A,B)
```



MATLAB/ Graphique

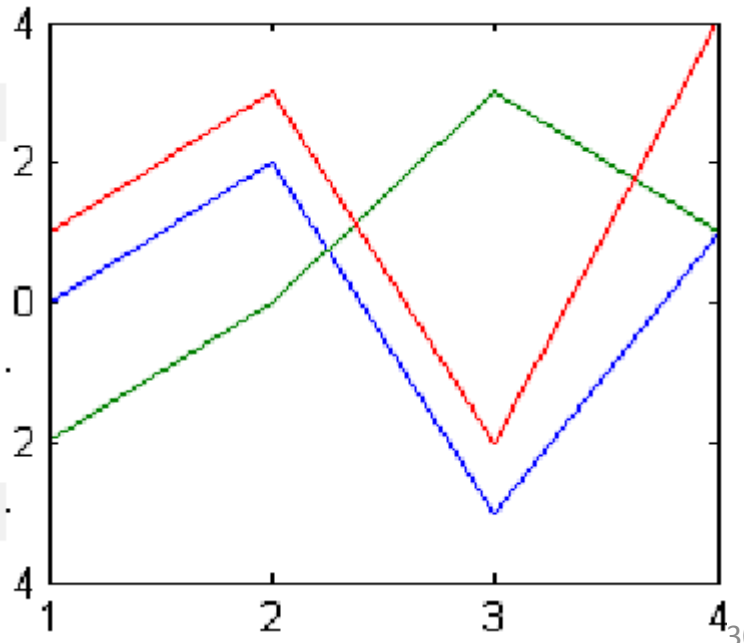
2- Si elle contient un seul vecteur comme argument : elle considère les valeurs du vecteur comme les éléments de l'axe Y (les ordonnées), et leurs positions relatives définirons l'axe X (les abscisses).

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

>> plot(V)
```



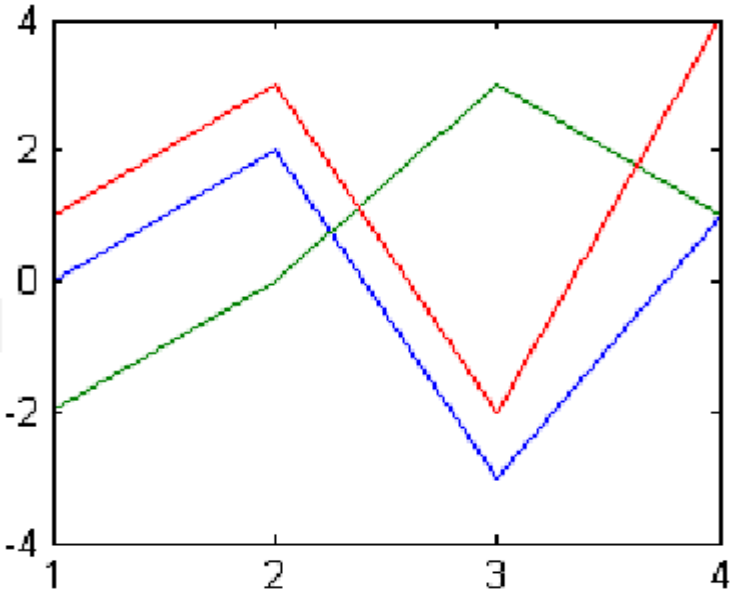
3- Si elle contient une seule matrice comme argument : elle considère les valeurs de chaque colonne comme les éléments de l'axe Y, et leurs positions relatives (le numéro de ligne) comme les valeurs de l'axe X. Donc, elle donnera plusieurs courbes (une pour chaque colonne).

```
>> 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)
```



4- 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.

```
>> 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)
```



Les options de style sont résumées dans le tableau suivant :

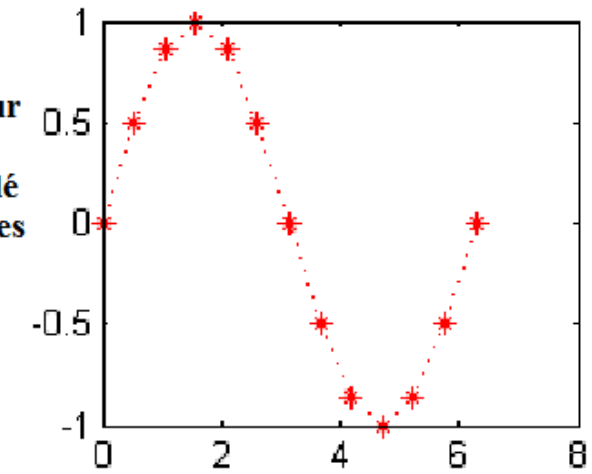
Couleur de la courbe	
le caractère	**son effet**
b ou **blue**	courbe en bleu
g ou **green**	courbe en vert
r ou **red**	courbe en rouge
c ou **cyan**	entre le vert et le bleu
m ou **magenta**	en rouge violacé vif
y ou **yellow**	courbe en jaune
k ou **black**	courbe en noir

Style de la courbe	
le caractère	**son effet**
-	en ligne plein ———
:	en pointillé ·····
-.	en point tiret – · – ·
--	en tiret – – –

Représentation des points	
le caractère	**son effet**
.	un point .
o	un cercle ●
x	le symbole x
+	le symbole +
*	une étoile *
s	un carré ■
d	un losange ♦
v	triangle inferieur ▼
^	triangle supérieur ▲
<	triangle gauche ◀
>	triangle droit ▶
p	pentagramme ★
h	hexagramme ✶

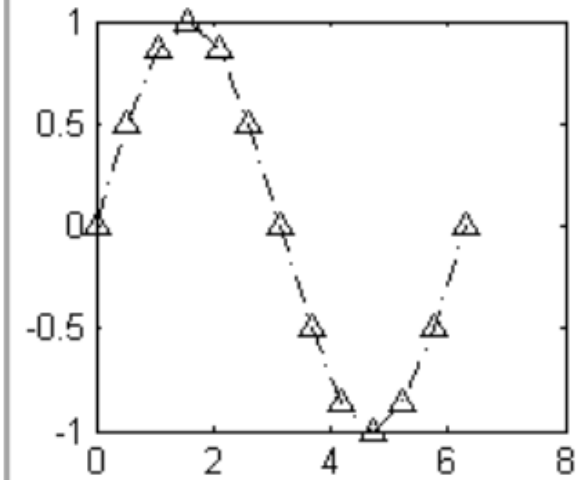
MATLAB/ Graphique

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



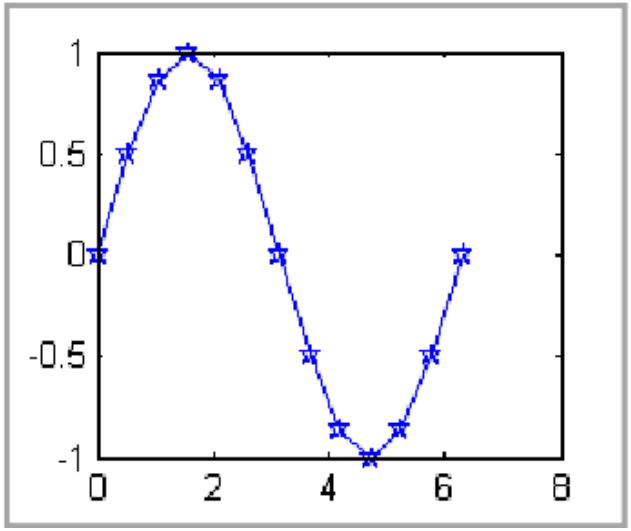
Couleur en pointillé avec des étoiles

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



Couleur noir en point tiret et avec des rectangles sup

plot(x, y,'pb-')



Couleur bleu, en ligne plein et avec des pentagrammes

MATLAB/ Graphique

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**).

MATLAB/ Graphique

```
>> 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')
```