1. Graphiques

1.1 Les graphiques et la visualisation des données en 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.

1.2 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 :

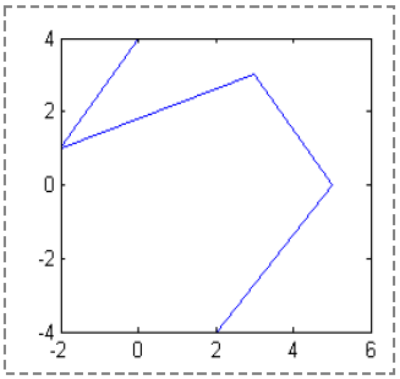
- **Deux vecteurs de la même taille comme arguments**
- **Un seul vecteur comme argument**
- **Une seule matrice comme argument**
- **Deux matrices comme arguments**

1.2.1 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];
>> B = [-4, 0, 3, 1,4];
>> plot(A,B)
```



1.2.2 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).

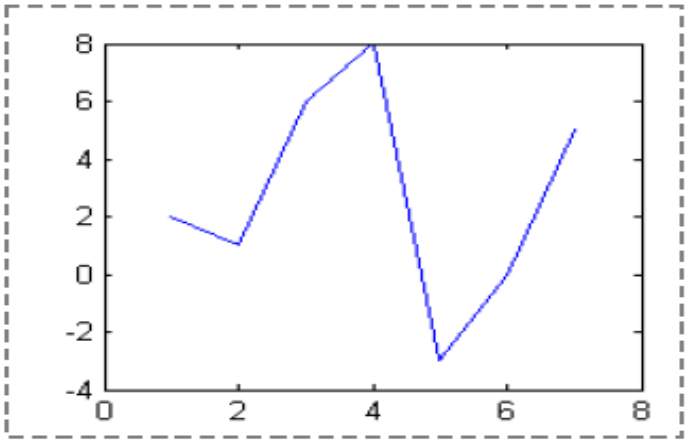
Exemple :

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

V =

2 1 6 8 -3 0 5

>> plot(V)



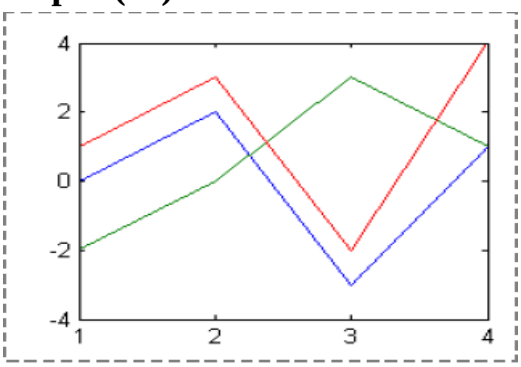
1.2.3 Une seule matrice comme argument

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]

>> plot(M)

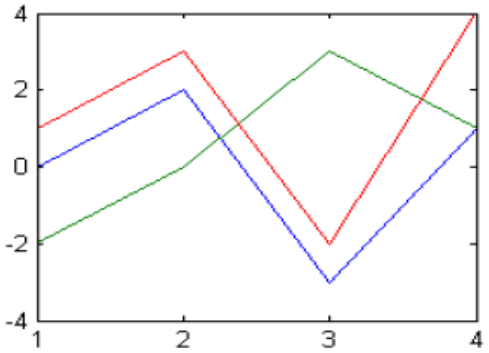


1.2.4 Deux matrices comme arguments

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];
>> M = [0 -2 1;
        2 0 3;
        -3 3 -2;
        1 1 4];
>> plot(K,M)
```



1.3 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 :

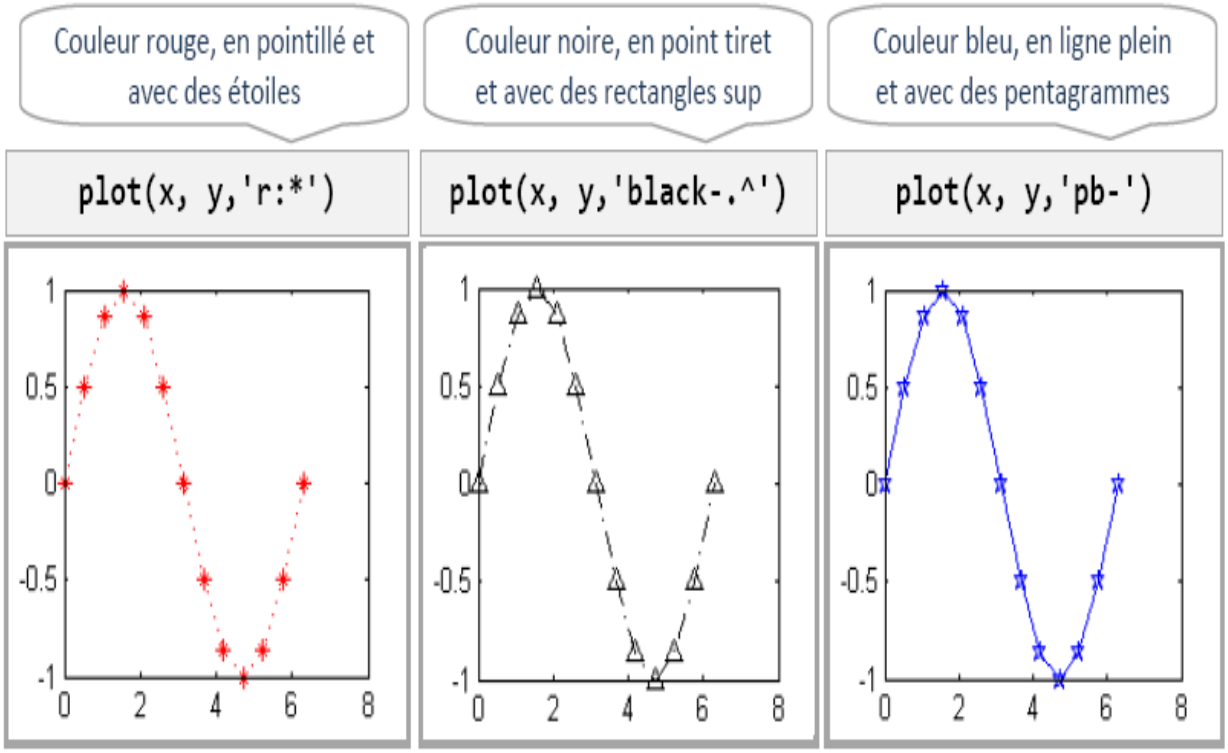
Essayons de dessiner la fonction :

$y = \sin(x)$ pour $x = [0 \ldots 2\pi]$ avec un pas $= \pi/6$.

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

>> y = sin(x);

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



1.4 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 : $y = -2x^3+x^2-2x+4$ 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;

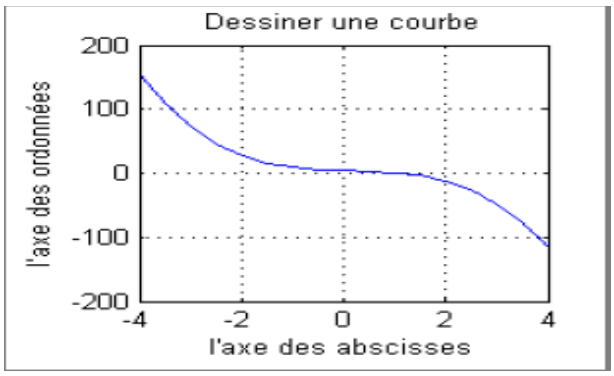
>> plot(x,y)

>> grid

>> title('Dessiner une courbe')

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

>> ylabel('l''axe des ordonnées')



1.5 Dessiner plusieurs courbes dans la même figure

Par défaut en MATLAB, chaque nouveau dessin avec la commande plot efface le précédent.

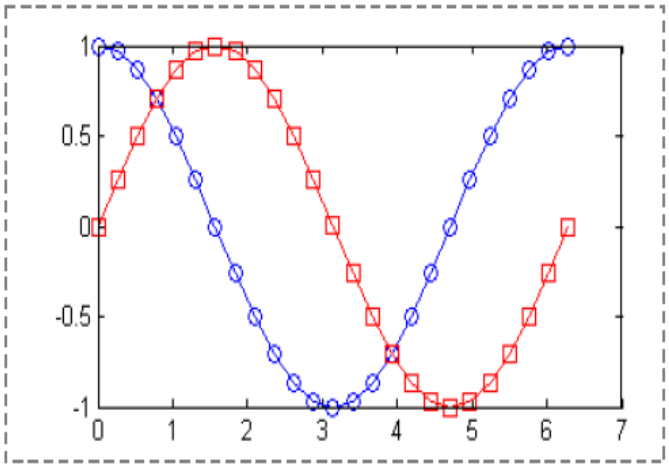
Pour forcer une nouvelle courbe à coexister avec les précédentes courbes:

1.5.1 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).

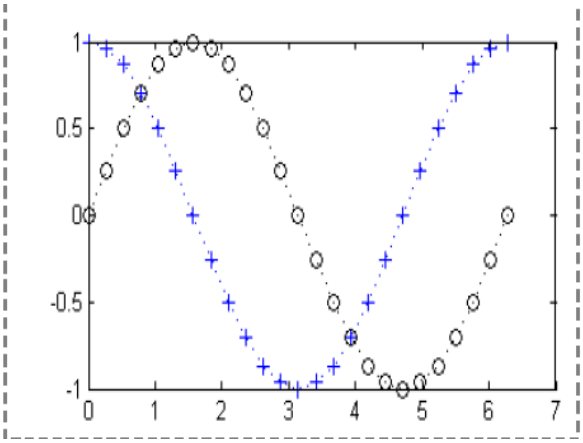
Par exemple pour dessiner la courbe des deux fonctions cos(x) et sin(x) dans la même figure, on peut écrire :

```
>> x=0:pi/12:2*pi;
>> y1=cos(x);
>> y2=sin(x);
>> plot(x,y1,'b-o')
>> hold on
>> plot(x,y2,'r-s')
```



1.5.2 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')
```



1.5.3 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 souris avec un glisser-déposer.

1.6 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. **ymin** 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**

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.**

1.7 D'autres types de graphiques

Le langage MATLAB ne permet pas uniquement l'affichage des points pour tracer des courbes, mais il offre aussi la possibilité de tracer des graphes à bâtons et des histogrammes.

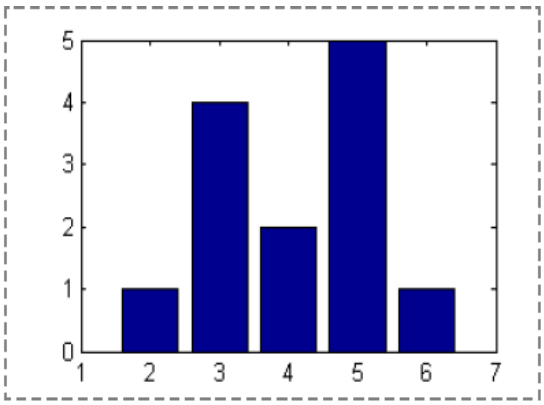
Pour tracer un graphe à bâtons nous utilisons la fonction **bar** qui a le même principe de fonctionnement que la fonction **plot.**

Exemple :

>> X=[2,3,5,4,6];

>> Y=[1,4,5,2,1];

>> bar(X,Y)



Il est possible de modifier l'apparence des bâtons, et il existe la fonction **barh** qui dessine les bâtons horizontalement, et la fonction **bar3** qui ajoute un effet 3D.

Parmi les fonctions de dessins très intéressantes on peut trouver : **hist, stairs, stem, pie, pie3, ...etc.**

Nous signalons aussi que MATLAB permet l'utilisation d'un système de coordonnées autre que le système cartésien comme le système de coordonnées polaire (les fonctions **compass, polar et rose).**

1.8 Tracer dans une ou plusieurs fenêtres

Suivant le contexte, on peut souhaiter que :

1.8.1 Les tracés apparaissent dans des fenêtres différentes : on crée autant de fenêtres que de tracés en utilisant la fonction **figure** :

- **Fenêtres graphiques**

  Création d'une fenêtre - fonctions figure et gcf

  La fonction **figure** crée une fenêtre graphique vide

  Exemple:

  >> h = figure

  h =

  1

Une fenêtre appelée Figure 1 apparaît. La valeur retournée par la fonction figure est le numéro de la fenêtre. Un second appel à la fonction figure crée une seconde fenêtre appelée, on s'en doute, Figure 2.

>> h = figure

h =

2

La dernière fenêtre crée est la fenêtre active (située au premier plan). Pour faire passer une fenêtre au premier plan, on utilise la fonction figure avec pour argument le numéro de la fenêtre que l'on souhaite activer.

Si aucune fenêtre portant ce numéro n'existe elle sera créée.

Exemple :

>> figure(1) ; x=[0:.1:2*pi] ; c=cos(x) ; plot(x,c)

>> figure(2) ; s=sin(x) ; plot(x,s)

Ces deux séquences construisent deux fenêtres, la première contenant le graphe de cos(*x*), la seconde le graphe de sin(*x*).

Réciproquement, la fonction **gcf** *(**g**et **c**urrant **f**igure)* retourne le numéro (ou référence) de la fenêtre active.

Exemple :

>> h = gcf

h =

2

>> figure(1)

>> h = gcf

h =

1

La fenêtre active est la Figure 2, puis on rend active Figure 1. Elle passe au premier plan (il est possible de faire la même chose en cliquant dans Figure 1).

- **Attributs d'une fenêtre - get**

- Les fenêtres possèdent un grand nombre d'attributs, par exemple un nom (Name), une couleur de fond (Color), . . .

- On obtient la liste complète des **attributs de la fenêtre active** et de leur valeur, par **get(n)** où **n** est le numéro de cette fenêtre.

- **Exemple :**

  >> h = gcf ;

  >> get(h)

  BackingStore = on

  CloseRequestFcn = closereq

  Color = [0.8 0.8 0.8]

  Colormap = [(64 by 3) double array]

  CurrentAxes = []

  . . .

La fonction **gcf** est utilisée pour obtenir le numéro de la fenêtre active, et **get** pour obtenir la liste des attributs de la fenêtre et de leur valeur sous la forme : *Nom de l'attribut = Valeur de l'attribut*.

On modifie la valeur des attributs d'une fenêtre avec la fonction **set** :

set(*'Nom de l'attribut', 'Valeur de l'attribut', . . . , . . .*)

On peut créer directement une fenêtre dont les attributs ont une valeur particulière :

figure(*'Nom de l'attribut', 'Valeur de l'attribut', . . . , . . .*)

Exemple :

La séquence :

>> figure('Name', 'essai', 'NumberTitle', 'off'), crée une fenêtre dont le nom est essai.

1.8.2 Afficher plusieurs graphiques (subplot)

- Voilà une fonctionnalité très utile pour présenter sur une même page graphique plusieurs résultats.
- L'idée générale est de découper la fenêtre graphique en pavés de même taille, et d'afficher un graphe dans chaque pavé.
- On utilise l'instruction **subplot** en lui spécifiant le nombre de pavés sur la hauteur, le nombre de pavés sur la largeur, et le numéro du pavé dans lequel on va tracer :

subplot (Nbre pavés sur hauteur, Nbre pavés sur largeur, Numéro pavé)

- La virgule peut être omise. Les pavés sont numérotés dans le sens de la lecture d'un texte de gauche à droite et de haut en bas.
- Une fois que l'on a tapé une commande **subplot**, toutes les commandes graphiques suivantes seront exécutées dans le pavé spécifié.

- **Exemple:**

x=-2*pi:2*pi/100:2*pi;

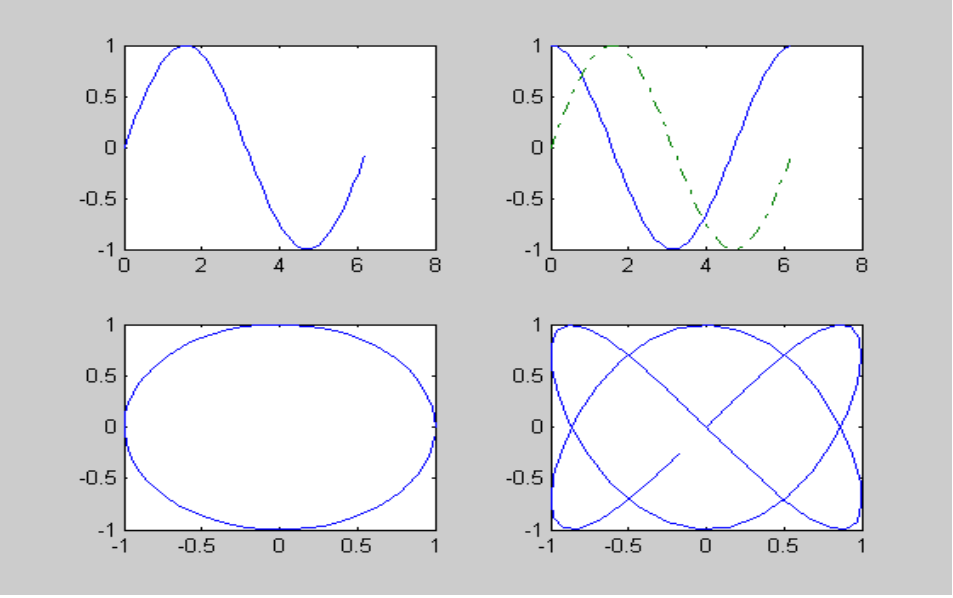
subplot(221) % sélectionner la première fenêtre

plot(x,sin(x))

title ('Représentation de sin(x)')

xlabel ('axe des x')

ylabel ('axe des y')
subplot(222) % sélectionner la deuxième fenêtre
plot(x,cos(x),x,sin(x),'-.')
subplot(223) % sélectionner la troisième fenêtre
plot(cos(x),sin(x))
subplot(224) % sélectionner la quatrième fenêtre
plot(sin(2*x),sin(3*x))



2. Polynômes

Pour MATLAB, un polynôme est une liste : la **liste des coefficients ordonnés par ordre décroissant** :

Exemple :

Le polynôme $p(x) = 1 - 2x + 4x^3$ est représenté par le liste :

```
>> p = [4 0 -2 1]
p =
4 0 -2 1
```

Les fonctions usuelles du calcul polynomial sont les suivantes :

Fonction	Arguments	Résultat
polyval	un polynôme p et un nombre a	calcul de $p(a)$
roots	un polynôme p	la liste des racines de p
conv	deux polynômes p et q	le polynôme produit $p \times q$
deconv	deux polynômes p et q	le quotient et le reste de la division euclidienne de p par q
polyder	un polynôme p	le polynôme-dérivée de p

3. CRÉER UNE INTERFACE GRAPHIQUE AVEC MATLAB

Pour rendre l'utilisation d'une application conviviale et intuitive, bien souvent, on lui ajoute une interface graphique, en d'autres termes une interface homme machine (IHM). Les domaines d'utilisations sont innombrables. Il est possible de le faire avec Matlab.

Comment créer une interface graphique?

Dans Matlab il existe un outil de création pour ce type de projet appelé: **GUIDE**.

Pour créer une interface graphique avec GUIDE, il existe deux méthodes:

- **Graphiquement** : méthode plus facile pour dessiner les objets, mais assez complexe à coder;
- **En créant un code sous forme d'une suite d'instructions**: méthode création difficile, mais facile à coder car on a la maitrise totale sur les objets, le débogage s'en trouve aisé, ainsi que la maintenance.

On peut aussi combiner les deux méthodes.

3.1 Graphiquement:

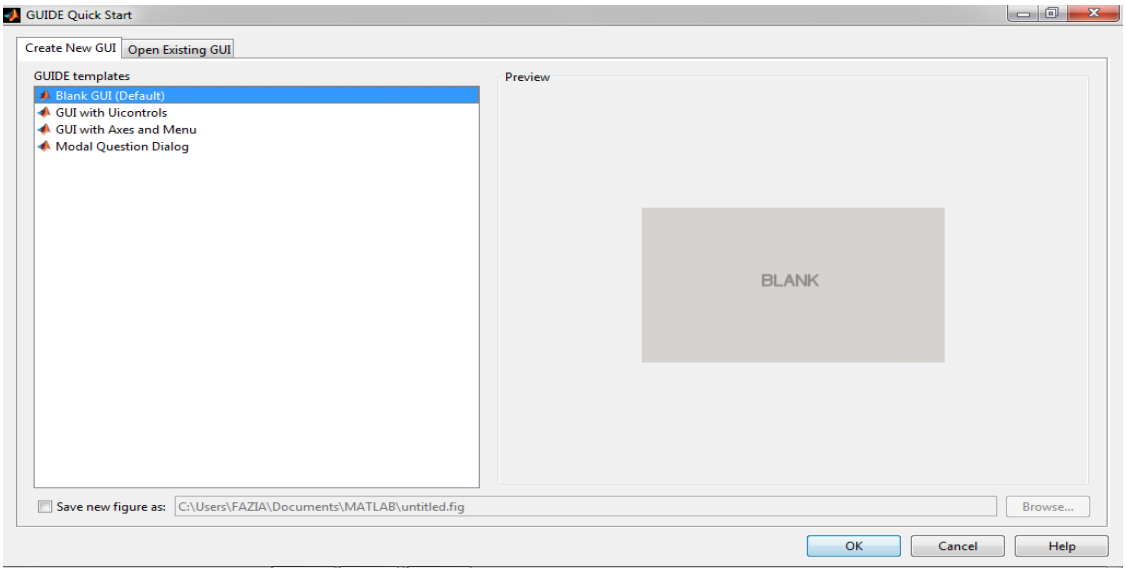
Exemple:

Création d'une interface graphique pour dessiner la fonction sinus,

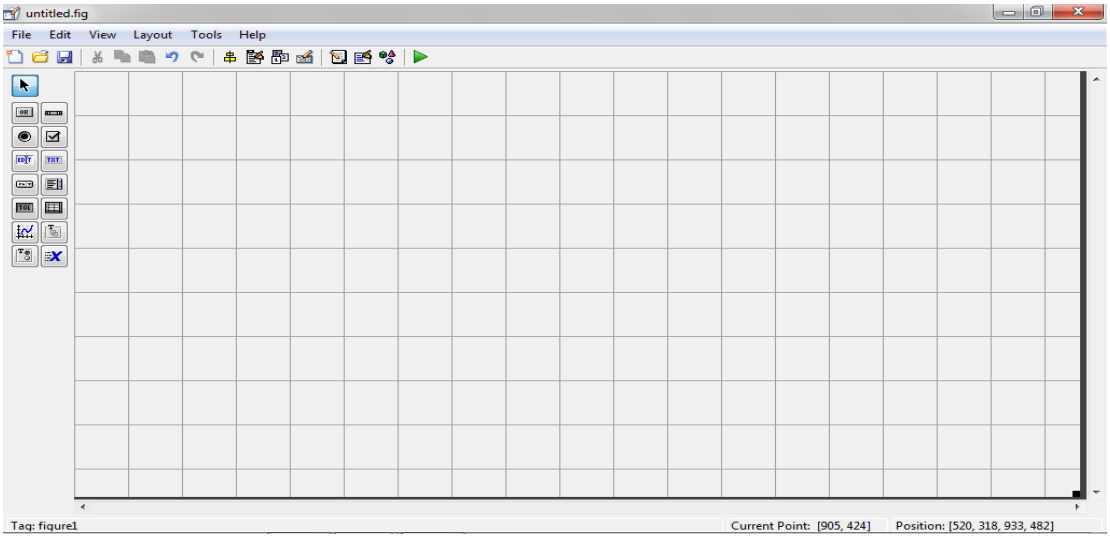
Dans la fenêtre de commandes, taper:

```
>> guide
```

Une fenêtre s'ouvre:



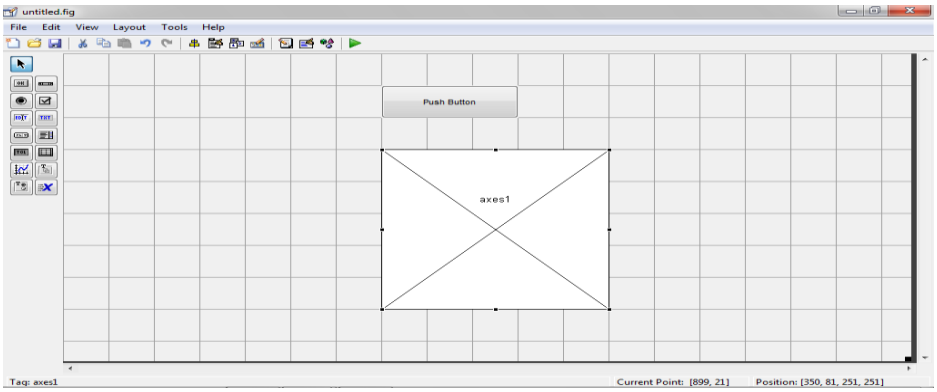
Quatre modèles d'interfaces sont proposés, on va choisir pour cet exemple un modèle vide: choisir "Blank GUI (Default)" puis cliquer sur OK.



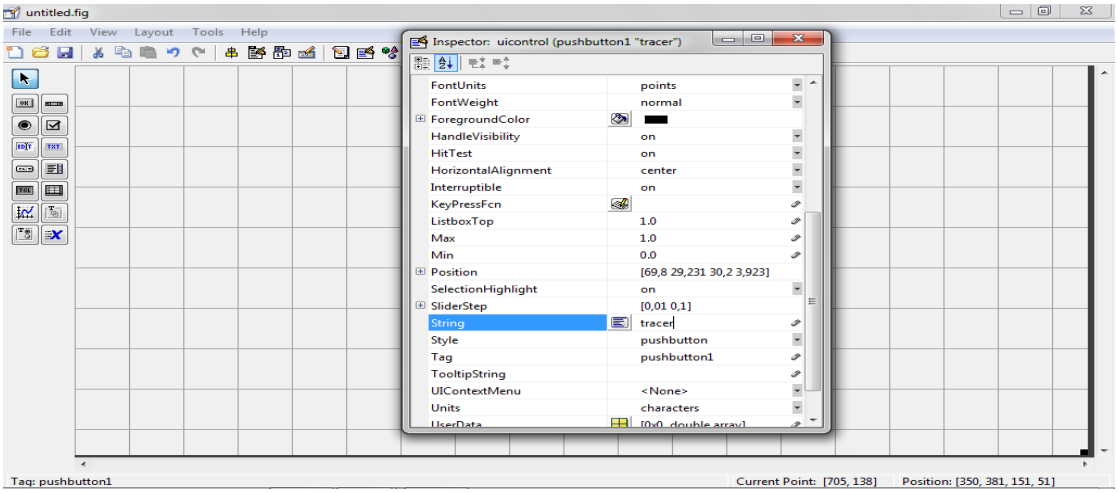
A gauche de l'interface plusieurs objets sont proposés: des **boutons**, des **cases à cocher**, des **menus déroulants**, **des graphes**...

Cliquer sur "**Push Button**", puis dessiner le bouton sur l'espace quadrillé à droite. Puis choisir "**Axes**", pour dessiner des axes.

Votre figure devra ressembler à ceci:



Pour changer le nom du bouton poussoir, cliquer deux fois dessus, puis dans la fenêtre "Inspecteur des propriétés" qui s'ouvre changé le nom dans devant la propriété "String": Taper « Tracer », puis cliquer de nouveau sur "String":



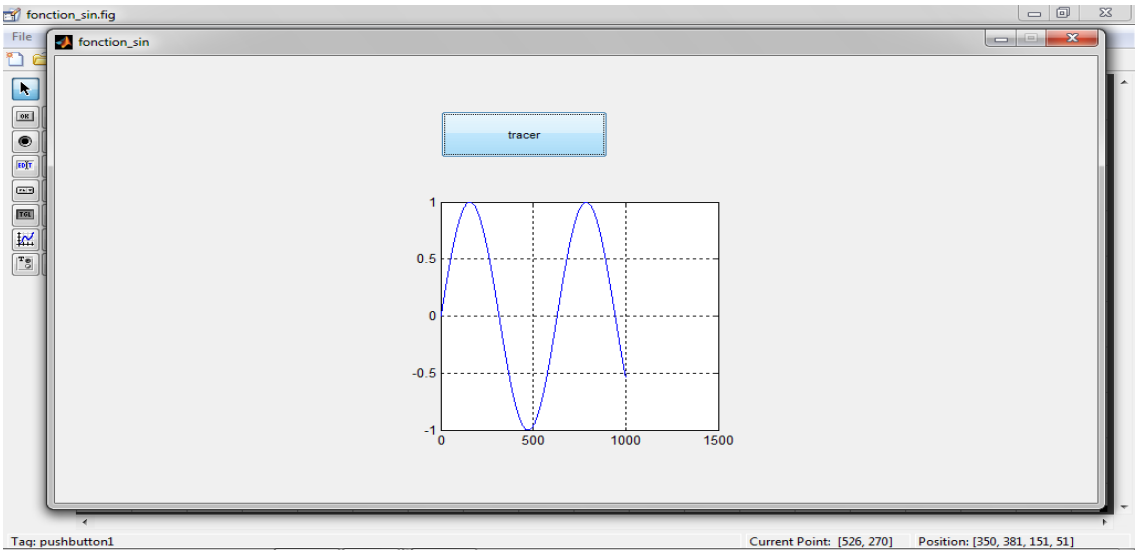
Enregistrer l'interface obtenue (en lui donnant le nom "fonction_sin" par exemple), ou en utilisant le raccourci clavier " Ctrl+S".

Dans, Matlab les instructions qui permettent de représenter l'interface graphique sont codées sous forme d'une fonction. Le code obtenu est donc affiché sous forme d'un script Matlab.

Sur l'interface graphique qui vient d'être crée, faire un clic droit sur le bouton "Tracer", choisir « **View Callbacks** », puis cliquer "**Callback**": Le curseur de la souris vous amène directement sur le Callback qui correspond au bouton Tester.

- **Qu'est-ce qu'un Callback**? C'est une action qui devra être exécutées lorsque l'utilisateur de l'interface va cliquer sur le bouton. Pour ce qui est de l'exemple proposé ici, on veut simplement dessiner la fonction sin(x). Coller les instructions suivantes à la suite de ce Callback:

- Enregistrer le script, puis l'exécuter soit en cliquant sur la flèche verte, soit en tapant dans la fenêtre de commande "**>>fonction_sin**", ou alors CTRL+T dans l'interface si elle encore ouverte. Exécuter, puis cliquer sur le bouton Tracer.



Remarque:

Lors de l'enregistrement, le GUIDE génère deux fichiers :

- un fichier .fig (non éditable) contenant les objets graphiques Figure, Axes et Pushbutton ;
- un fichier .m contenant le code du fonctionnement de l'interface graphique.

3.2 Programmation à la main

Il est possible de programmer une interface graphique entièrement à la main sous MATLAB. Bien que cette seconde méthode semble beaucoup moins intuitive que celle utilisant le GUIDE.

Le code peut être écrit dans un ou plusieurs fichiers .m (conception modulaire) et l'ouverture de l'interface graphique se fait en lançant simplement le fichier .m principal comme une fonction MATLAB.

```
function exemple_main

%EXEMPLE_MAIN Exemple d'une interface graphique codée à la main

 % Création de l'objet Figure

fig = figure('units', 'pixels', ... 'position', [520 380 300 200], ... 'name', 'example_main');

% Création de l'objet Uicontrol Pushbutton

 uicontrol('style', 'pushbutton', ... 'units', 'pixels', ... 'position',[75 10 150 20], ... 'callback', @cb);
```

```
% Création de l'objet Axes
axes('units', 'pixels', ... 'position', [25 40 250 150], ... 'tag','axes1');
% Stockage des identifiants utiles
handles = guihandles(fig); guidata(fig,handles) function cb(obj,event)
% Fonction associée au Callback de l'objet Pushbutton
% obj : identifiant de l'objet Pushbutton
% event : événement liés à l'objet Pushbutton
% Récupération des identifiants utiles fig = get(obj,'parent'); handles = guidata(fig);
% Modification de la couleur de l'objet Axes set(handles.axes1, 'color', rand(1,3));
```

Références :

[1] Cours Outils de programmation pour les mathématiques, Y.BAZIZ, Centre Universitaire RELIZANE, Institut des Sciences et Technologies, Spécialité : Mathématique et Informatique, LMD 1ére année 2013/2014.
[2] Tutoriel MATLAB, Introduction aux Méthodes Numériques 2013-2014.
[3] poly introduction Matlab, Introduction à Matlab
[4] Introduction à MATLAB, André Casadevall, Université Paris-Dauphine, Département MIDO, mars 2013
[5] Outils Mathématiques et utilisation de Matlab, Quentin Glorieux , Cours 2013-2014, Université Pierre et Marie Curie - Paris VI, Licence Professionnelle (L3) Instrumentation Optique et Visualisation,.