# Tutoriel Matlab

## 1. Installation

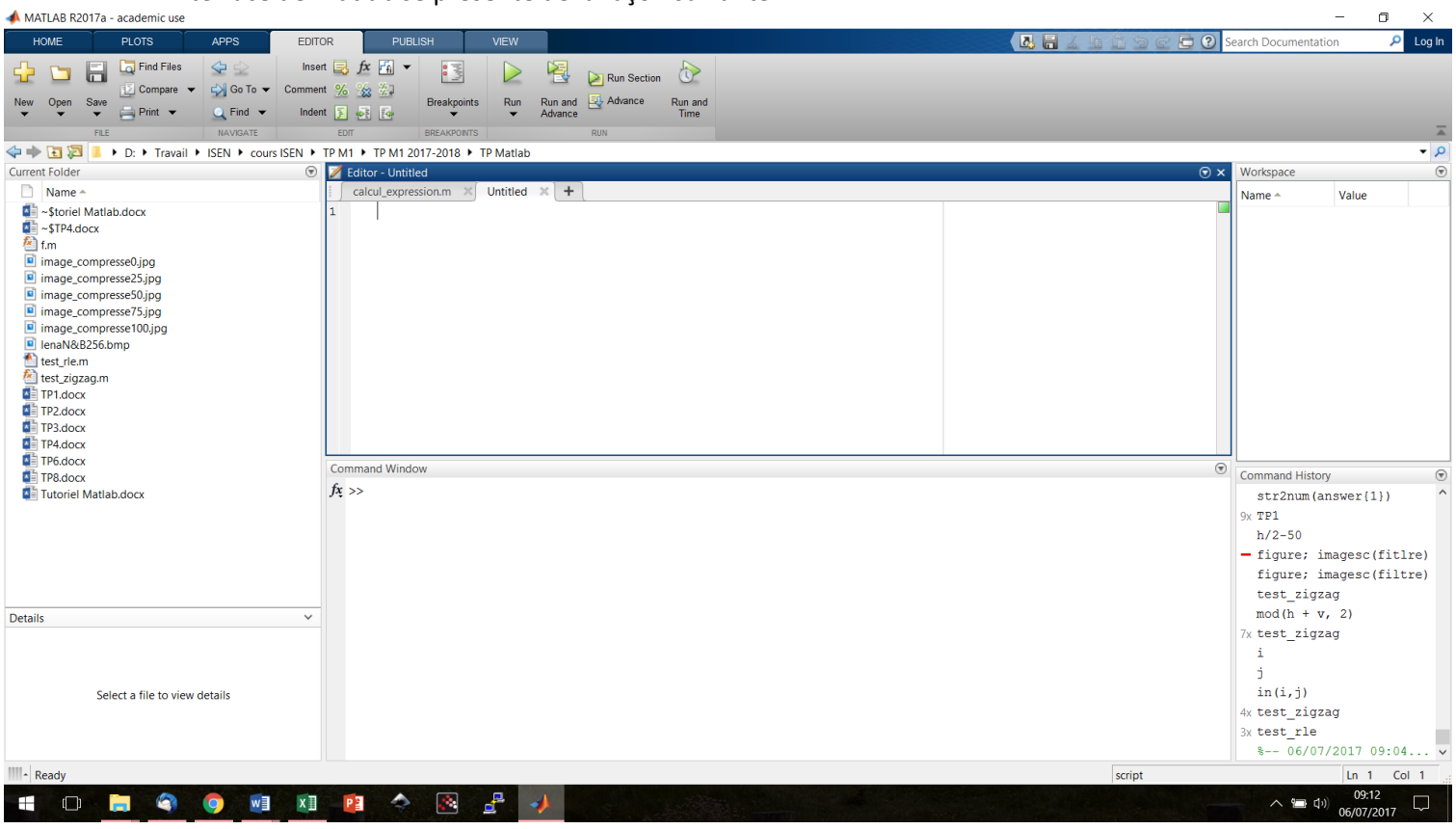
L'ISEN dispose d'une licence campus pour le logiciel Matlab, depuis la version R2016. Suivez les recommandations du fichier "Licence MATLAB Etudiants".

2 remarques à prendre en compte pour que tout fonctionne au mieux !

- Créer un compte MathWorks en utilisant votre adresse mail (du type prenom.nom@isen-bretagne.fr).
- Télécharger le programme d'installation (attention il doit se trouver dans un dossier dont le nom ne comporte pas d'accents. Par exemple si le programme est situé dans le dossier « Téléchargements » cela ne fonctionnera pas !)

## 2. Prise en main du logiciel Matlab

L'interface de Matlab se présente de la façon suivante :




On peut trouver une fenêtre "**command window**". Dans cette fenêtre vous pouvez écrire vos lignes de commande. C'est aussi ici que s'afficheront les résultats ainsi que les erreurs de compilation.

La fenêtre "**command history**" permet de retrouver l'historique des commandes exécutées dans la "command window". Un double-clic sur la commande voulue permet de la ré-exécuter.
Il y a deux espaces nommés "**current directory**". L'un vous informe du dossier dans lequel vous vous situez. L'autre vous montre les fichiers contenus dans ce dossier. Vous devez exécuter votre programme dans le dossier dans lequel il est enregistré.
La fenêtre "**workspace**" recense toutes les variables utilisées lors de l'exécution de votre programme. Vous connaitrez ainsi la taille de la variable, ses valeurs minimale et maximale.
Enfin la fenêtre "**editor**" vous permet d'écrire un programme, de le sauvegarder,.... Dans la partie droite de la fenêtre, il y a un petit carré vert, qui peut aussi être orange ou rouge. S'il est vert, le programme écrit ne contient pas d'erreur de syntaxe. S'il est orange ou rouge, des petits tirets vous indiqueront la ligne et le problème de syntaxe (un clic sur ce tiret vous emmène directement à la ligne concernée). Orange est la couleur de l'avertissement, rouge de l'erreur. Dans ce dernier cas, le programme ne compilera pas!

Si l'une de ces fenêtres n'est pas affichée, vous la trouverez dans le menu "Desktop".

Matlab est un calculateur. Lorsque vous avez un calcul à faire et que vous travaillez sur Matlab, ***inutile de sortir votre calculatrice***!

Une aide très complète est fournie avec le logiciel. Vous la trouverez dans l'onglet "Help" -> "Matlab Help" ou avec le raccourci clavier F1. Des exemples d'utilisation des différentes fonctions sont donnés ainsi que des fonctions proches (rubrique *See Also*). Pour une réponse rapide, il est possible d'utiliser la "command window". Vous pouvez écrire, par exemple, "help sin" dans la "command window" pour obtenir de l'aide sur la fonction sinus.
Les différentes fonctions de Matlab sont classées en "Toolbox". Les fonctions de références sont inclues dans une toolbox Matlab les autres sont en option. De plus d'une version Matlab à l'autre, des fonctions ont été ajoutées, supprimées ou mises à jour. Il faut donc faire attention à la version de Matlab que vous utilisez.

Les fichiers de code enregistrés par Matlab ont l'extension ".m". Les figures sont enregistrées en ".fig". Il est possible de les enregistrer sous d'autres formats avec le menu "save as". Les données sont au format ".mat".

Il existe des logiciels proches de Matlab en version libre. Citons par exemple Octave ou Scilab.

# 3. Généralités

## a. Nom des fichiers

- Ne pas utiliser le signe " -"
- Ne pas commencer le nom du fichier par un chiffre ou un nombre
- Attention à ne pas nommer un fichier comme une fonction déjà implantée dans Matlab. Dans ce cas, Matlab utilisera en priorité votre propre fonction

Exemples :

- 1-monfichier.m ne marchera pas, écrire plutôt mon_fichier_1.m
- vous appelez un fichier "sin.m". Ce document contient votre algorithme de calcul de la fonction sinus. Il y a alors conflit entre votre fichier et la fonction sinus de Matlab.

### b. Syntaxe

- Pour exécuter une ligne de code sans afficher le résultat dans la fenêtre "command window", il faut mettre un ";" à la fin de la ligne.
  Exemple : x=2
  x=2;
- Pour commenter une ligne, on utilise le signe "%".
  Raccourcis clavier : ctrl+r pour commenter et ctrl+t pour décommenter

- Un bon réflexe à prendre : au début de votre script, mettre les lignes de codes suivantes :

```
clc;
clearvars;
close all;
```

"clc;" sert à effacer la fenêtre "command window". "clearvars;" sert à effacer toutes les variables présentes dans la mémoire du logiciel (si vous ne souhaitez n'effacer que la variable 'x', vous devez écrire "clear x"). "close all;" sert à fermer toutes les figures ouvertes. Ainsi lorsque vous exécutez votre programme, vous êtes sûr que les variables/résultats/figures/erreurs/... correspondent à l'exécution du programme.

## 4. Debug

Il est possible d'exécuter un programme ligne par ligne. Pour cela, il faut mettre un point d'arrêt à la ligne à partir de laquelle vous souhaitez exécuter le programme.
Voici les icones correspondant à l'exécution pas à pas :

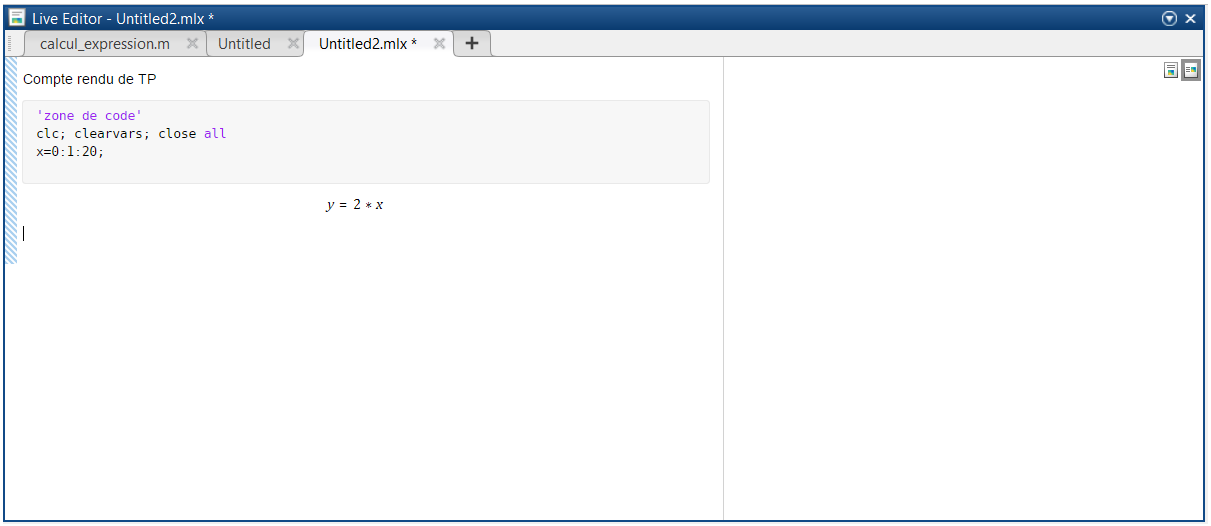
- La 1ère icone permet de positionner et d'enlever des points d'arrêts (raccourci clavier F12)
- La 2ème icone permet de supprimer tous les points d'arrêts
- La 3ème icone permet d'exécuter la ligne et de passer à la suivante (raccourci clavier F10)
- La 4ème icone permet d'entrer dans la fonction appelée dans la ligne de commande (raccourci clavier F11)
- La 5ème icone permet de sortir de la fonction dans laquelle vous êtes entrés avec le bouton précédent
- La 6ème icone permet de finir l'exécution du programme (raccourci clavier F5). Ce bouton permet aussi l'exécution du programme
- La dernière icone permet d'arrêter le mode debug

Ce mode ne fonctionne que pour les scripts enregistrés.

## 5. Ecriture de rapports avec Matlab

Il est possible d'écrire des rapports directement dans Matlab avec « **live script** ». Pour cela aller dans New -> Live Script.

Cette fonctionnalité ouvre un fichier .mlx. Dans ce fichier il est possible d'insérer du texte, du code Matlab, des équations, des images,… Lors de l'enregistrement il est possible de créer un pdf.




## 6. Vecteur / matrice

Matlab est un logiciel de calcul matriciel. Les données doivent être disposées sous forme de vecteurs ou de matrices.

### a. Définition de matrices

Une matrice de valeurs est définie par : **[]**. La séparation entre deux colonnes se fait avec un espace ou une virgule. La séparation entre deux lignes se fait avec un point virgule "**;**".
Exemple :

```
>> x = [1 2 3; 4 5 6]
x =
   1   2   3
   4   5   6
```

Pour définir une matrice contenant uniquement des éléments à 0, il existe une fonction : **zeros**(nombre_de_lignes, nombre_de_colonnes)
Pour définir une matrice remplie d'une seule valeur, il existe la fonction **ones** qui définit une matrice remplie de 1. Il suffit ensuite de la multiplier par la valeur souhaitée.
Exemple :

```
>> x=5*ones(4,3)
x =
   5   5   5
   5   5   5
   5   5   5
   5   5   5
```

Il est possible de générer des valeurs/vecteurs/matrices de valeurs aléatoires comprises entre 0 et 1 avec la fonction "**rand**". Il est aussi possible de générer des matrices identité avec la fonction "**eye**".
Pour accéder à un élément de la matrice, il faut donner en paramètres l'indice de la ligne puis l'indice de la colonne :

```
>> x=[1 2 3; 4 5 6]
x =
   1   2   3
```

```
    4    5    6
>> x(1,2)
ans =
    2
```

De même, on peut extraire une sous matrice A(1:2,1:3) qui aura 2 lignes et 3 colonnes.
Pour obtenir la taille d'un vecteur, on utilise la fonction "**length**". Pour obtenir la taille d'une matrice, on utilise la commande "**size**" :

```
>> [a b]=size(x)
a =
    4
b =
    3
```

Attention, mettre autant de paramètres de sortie que de dimensions de x. Par exemple si x est une matrice en 3 dimensions et qu'il n'y a que deux paramètres de sortie, le deuxième correspondra au produit de la dimension 2 avec la dimension 3.

```
>> x=zeros(2,3,5);
>> [a b]=size(x)
a =
    2
b =
   15
>> [a b c]=size(x)
a =
    2
b =
    3
c =
    5
```

## b. Opérations sur les matrices

Matlab connait deux types de multiplications : la multiplication matricielle définie en mathématique et la multiplication matricielle terme à terme.
Pour multiplier deux matrices, au sens mathématique, il suffit d'utiliser le symbole " * " : A*B.
Pour multiplier deux matrices terme à terme, il faut utiliser les symboles " .* " : A.*B.
Cette distinction existe aussi pour la division : " / " et " ./ " et pour la puissance : " ^ " et ".^ ".

Quelques fonctions utiles :

- Racine carrée : sqrt(A)
- Exponentielle : exp(A)
- Logarithme népérien : log(A)
- Logarithme en base 2 : log2(A)
- Logarithme en base 10 : log10(A)
- Norme : norm(A)
- Déterminant : det(A)
- Trace : trace(A)

### c. Définition d'un vecteur axe

Il est possible d'avoir à utiliser un vecteur dont on connait la valeur de départ, la valeur finale, le nombre de points ou l'espace entre deux points consécutifs. Cela est très pratique lors de l'affichage d'un graphe, pour faire l'axe des abscisses par exemple. C'est aussi utile pour les boucles.

- 1er cas : on connait l'espace entre deux points. Ici on souhaite commencer à 0 et finir à 1 avec des points tous les 0.1. On définit un vecteur de la façon suivante : [valeur_depart : pas : valeur_arrivée]. Matlab calcule automatiquement le nombre de points nécessaires. Si le pas n'est pas défini, il sera par défaut de 1.

```
>> [0 :0.1 :1]
ans =
    0  0.1000  0.2000  0.3000  0.4000  0.5000  0.6000  0.7000  0.8000  0.9000
1.0000
```

- 2ème cas : on connait le nombre de points. Par exemple, on veut commencer à 0, finir à 1 avec 10 points. Pour cela on utilise la fonction linspace(valeur_depart, valeur_arrivee, nombre_points)

```
>> linspace(0,1,10)
ans =
    0  0.1111  0.2222  0.3333  0.4444  0.5556  0.6667  0.7778  0.8889  1.0000
```

### d. Divers

Les indices des valeurs commencent à 1. X(0) conduira à une erreur.

Pour passer d'un vecteur ligne à un vecteur colonne (transposée), il suffit d'utiliser le symbole " ' " :

```
>> x=[1 2 3 4 5]
x =
   1   2   3   4   5
>> x'
ans =
   1
   2
   3
   4
   5
```

```
>> x=[1 2 3; 4 5 6]
x =
   1   2   3
   4   5   6
>> x'
ans =
   1   4
   2   5
   3   6
```

Dans le cas de valeurs complexes, on obtiendra la transposée conjuguée complexe.

```
>> x=[1+i 2+3*i 5i]
```

```
x =
  1.0000 + 1.0000i   2.0000 + 3.0000i       0 + 5.0000i
>> x'
ans =
  1.0000 - 1.0000i
  2.0000 - 3.0000i
     0 - 5.0000i
```

Il est possible d'avoir besoin de concaténer deux vecteurs. Pour cela, on utilise les symboles "[" et "]" :

```
>> x=[1 2 3 4 5]
x =
   1   2   3   4   5
>> y=[9 8]
y =
   9   8
>> z=[x y]
z =
   1   2   3   4   5   9   8
```

Il est possible de supprimer un élément d'un vecteur:

```
>> A=[9 8 7 8 6 5 4]
A(2)=[]
A =
   9   8   7   8   6   5   4
A =
   9   7   8   6   5   4
```

Il est aussi possible de supprimer une ligne ou une colonne d'une matrice:

```
>> B=[1 2 3; 4 5 6; 7 8 9]
B(2,:)=[]
B =
   1   2   3
   4   5   6
   7   8   9
B =
   1   2   3
   7   8   9
>> C=[9 8 7; 6 5 4; 3 2 1]
C(:,3)=[]
C =
   9   8   7
   6   5   4
   3   2   1
C =
   9   8
   6   5
   3   2
```

## 7. Boucle

### a. Boucle for

Une boucle for s'écrit de la façon suivante :

```
for variable = valeur depart  : pas  : valeur finale
        instruction
        ...
        instruction
end
```

Ne pas oublier le mot "end" à la fin de la boucle
Si le pas n'est pas défini, il sera par défaut de 1.

### b. Boucle while

Une boucle while s'écrit de la façon suivante :

```
while expression
        instruction
end
```

### c. Instruction if

Une instruction if s'écrit de la façon suivante :

```
if expression1
   instruction1
elseif expression2
   instruction2
else
   instruction3
end
```

Attention, à l'écriture des opérateurs logiques. Certains opérateurs sont doublés dans l'instruction d'une boucle (les signes "=", "|" et "&") :

```
a=2;
b=2;
if a==b
        c=1;
else
        c=0;
end
```

Les différents opérateurs logiques :

| | |
|---|---|
| = | Egalité |
| ~= | Inégalité |
| \| | Ou |
| & | Et |
| > | Supérieur |
| < | Inférieur |

## 8. Analyse d'un signal (cf. cours de traitement du signal)

Transformée de Fourier : **fft**(x)
Pour forcer le nombre de points sur lequel est calculée la transformée de Fourier, on utilisera la commande : fft(x,n).
Transformée de Fourier inverse : **ifft**(x)

Tout signal x(t) sur Matlab est en fait un signal échantillonné avec un pas d'échantillonnage Te. Par conséquent, la transformée Discrète de Fourier du signal est périodique de période Fe. Ainsi, les valeurs de l'intervalle [0, Fe] sont équivalentes aux valeurs de l'intervalle [-Fe/2, Fe/2].
Matlab calcule les valeurs sur l'intervalle [0, Fe]. Ce sont ces valeurs qui sont représentées. Pour représenter le spectre sur l'intervalle [-Fe/2, Fe/2], on peut utiliser la fonction **fftshift**.

## 9. Fonction

Pour limiter le nombre de lignes de codes, il est possible de créer ses propres fonctions.
Dans un nouveau fichier .m, il faut commencer par

```
function [out1, out2, ...] = nom_de_la_fonction (in1, in2, ...)
```

Ensuite il suffit d'écrire le code que l'on veut en restant en accord avec les paramètres d'entrée et de sortie que l'on a définis.

Lors de l'enregistrement, Matlab vous proposera automatiquement d'enregistrer le fichier avec le nom de la fonction. Il faut enregistrer la fonction dans le répertoire de travail, sinon Matlab ne la trouvera pas lors de l'exécution.

Exemple :
Un fichier contenant la fonction :

```
function b=carre(a)
%fonction qui calcule le carré de a, a étant un nombre
b=a*a;
```

Dans le fichier principal :

```
d=5;
c=carre(d);
```

## 10. Affichage

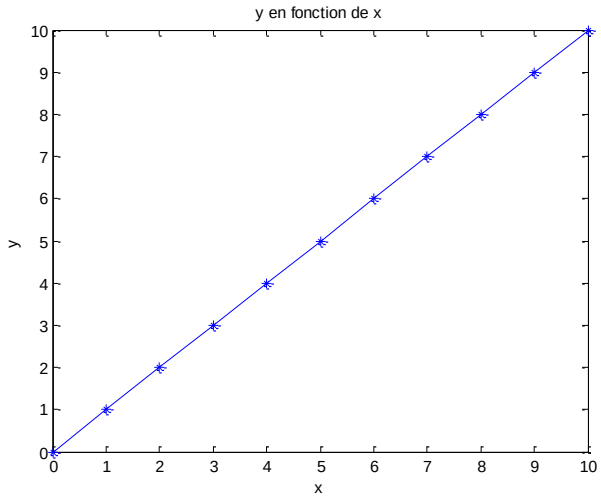
Pour tracer une courbe, on utilise la fonction "**plot**" dans l'environnement "**figure**".
La fonction "plot" prend comme argument :

- plot(y)
- plot(x,y)
- plot(x,y,options)

Si l'on ne définit pas d'axe x, Matlab graduera cet axe en nombre de points, de 1 en 1.
Les options sont multiples. On utilise principalement la couleur et les symboles. Vous en trouverez le détail en écrivant "Linespec" dans l'aide Matlab.
Pour afficher une grille dans la figure, on utilisera la ligne de commande "**grid on**".
Pour afficher plusieurs courbes sur la même figure, on utilisera la commande "**hold on**" après chaque tracé de courbe.
Les fonctions "**title**", "**xlabel**" et "**ylabel**" permettent de mettre un titre et des noms aux axes x et y. Ces fonctions prennent en paramètre une chaine de caractères (du texte écrit entre 2 apostrophes).

Exemple :

```
x=0 :10;
y=x;
figure; plot(x,y,'*-');
title('y en fonction de x');
xlabel('x');
ylabel('y');
```



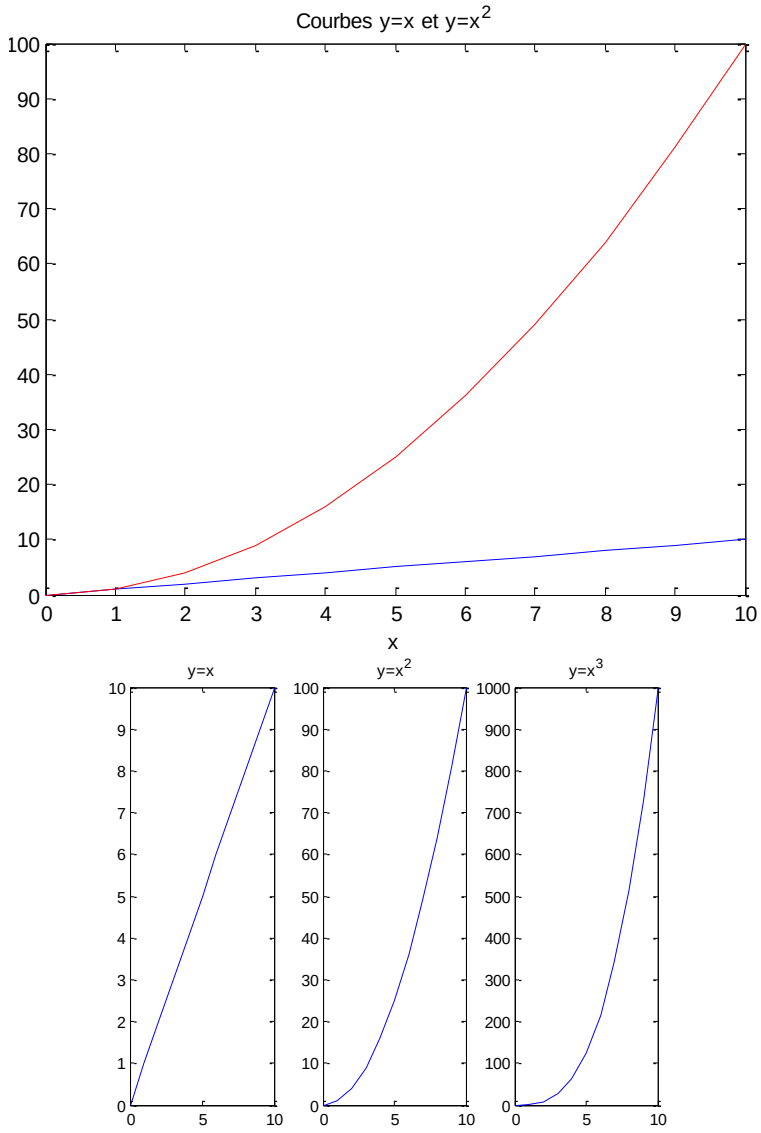

On peut vouloir afficher plusieurs courbes sur le même graphe ou plusieurs graphes sur la même figure.

Exemple :

```
x=0 :10;
y1=x;
y2=x.*x;
y3=x.*x.*x;

figure; plot(x,y1,'b'); hold on; plot(x,y2,'r');

figure;
subplot(1,3,1); plot(x,y1);title('y=x')
subplot(1,3,2); plot(x,y2);title('y=x^2')
subplot(1,3,3); plot(x,y3);title('y=x^3')
```




Par défaut Matlab ouvre une nouvelle figure à chaque appel du mot "figure". Il est possible de le faire afficher dans une figure spécifique : "figure(1)". Attention si cette figure a déjà été ouverte dans l'exécution du programme, la courbe affichée sera écrasée.

Attention, Matlab ne sait pas afficher des nombres complexes. Il faut alors afficher la valeur absolue, la partie réelle,....

## 11. Lecture/écriture d'un fichier

Pour enregistrer tout le workspace ou seulement quelques variables

```
save 'filename'
save('filename', 'var1', 'var2', ...)
```

Pour lire les données enregistrées

```
load('filename');
```

Pour lire un fichier excel
xlsread('filename')
Pour écrire dans un fichier excel
xlswrite('filename', données)

## 12. Image

Il existe plusieurs types d'images. Les images binaires ne contiennent que deux valeurs de pixels : noir et blanc. Les images en niveaux de gris contiennent toute une palette de gris, allant du blanc au noir. Une image en couleurs est une image dont chaque pixel est défini par un mélange de rouge, de bleu et de vert. Lorsque l'on étudie ces images avec Matlab, on peut noter qu'elles sont en 3D. En effet, la première dimension contient les coefficients du rouge, la deuxième ceux du vert et la troisième ceux du bleu. Ainsi toutes les couleurs peuvent être reconstruites. Attention il existe d'autres systèmes de représentation des couleurs (HSV, YUV, YCbCr, ....)

Lecture d'une image
im=**imread**('nom.png');

Conversion d'une image RBG en image en niveaux de gris
im_gris=**rgb2gray**(im);
(Il existe d'autres fonctions de conversion selon l'espace couleur de départ et celui d'arrivée)

Il est existe une multitude de méthodes pour extraire et analyser le contenu d'une image. Nous allons aborder ici les méthodes de bases : histogramme, contraste, contours.

Histogramme d'une image
Définir ce qu'est l'histogramme d'une image et son utilisation.
**imhist**(im);

Dynamique d'une image
Quel est l'intérêt de modifier la dynamique d'une image?
Il est possible de modifier la dynamique d'une image :
**imadjust**(X,[low high], [bottom top])

Les contours
Le contour d'une image est défini comme l'ensemble des points à fort gradient où la dérivée seconde est nulle. Pour plusieurs applications le contour d'une image (représenté par les hautes fréquences) contient les informations nécessaires et suffisantes applicables à la reconnaissance des formes. Matlab contient plusieurs détecteurs de contours :
contour=**edge**(im,options)
options : le détecteur de contours souhaité et éventuellement les seuils

Affichage d'une image
Il existe plusieurs fonctions pour afficher une image :
- La fonction **image** pour les images en couleurs
- La fonction **imagesc** pour les images dont l'intensité varie entre une valeur minimale et une valeur maximale arbitraire

- La fonction **imshow** prend tous les types d'images. Les valeurs d'affichage des images codées en uintX sont comprises entre [0 $2^X$-1]. Par conséquent les images uint8 (par exemple) dont les valeurs sont comprises entre 0 et 1 produiront une image noire.

Les fonctions image et imagesc appartiennent aux fonctions de base Matlab. La carte de couleur par défaut est la carte Jet (du rouge au bleu). Il est possible de la modifier avec la commande **colormap** (voir l'aide pour les différentes colormap disponibles). La fonction imshow appartient à la toolbox image processing.

Exemples d'écriture
figure; image(im); title('lena');
figure; imagesc(im); title('lena');
figure; imshow(im); title('lena');

A noter: Lors de l'affichage d'une image, le point (0,0) se situe en haut à gauche.