Matlab

Prise en main

Généralités

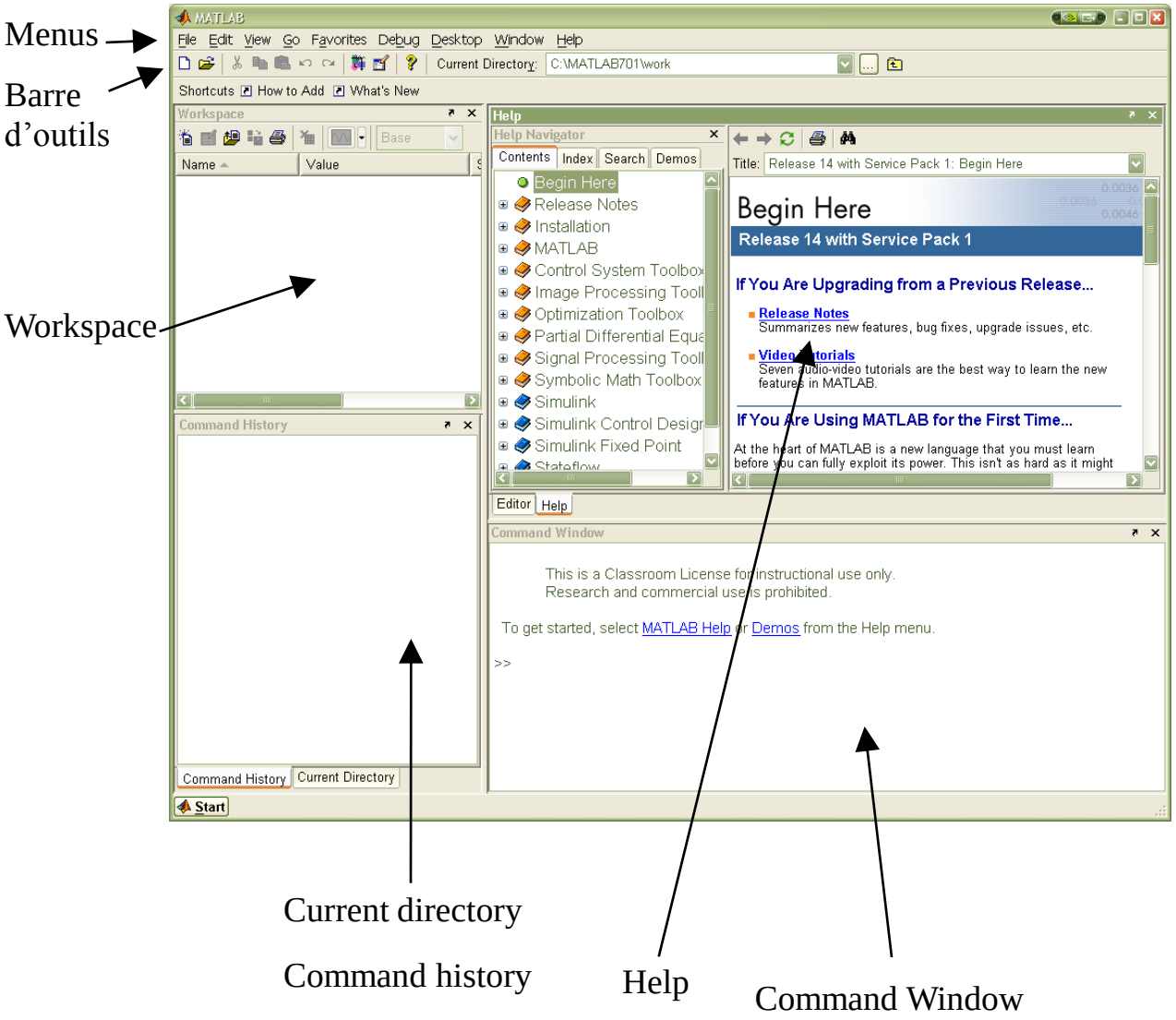
- MATLAB: contraction de MATRIX LABORATORY
- Première version crée en 1970 par Cleve Moler
- 500 000 utilisateurs répartis dans l'industrie, les administrations et les établissements scolaires
- utilisé dans de nombreux domaines d'application:
  - Traitement du signal,
  - Traitement d' images,
  - Conception de systèmes de contrôle,
  - Instrumentation
  - Sciences de la terre et de la vie,
  - Finances, économie
- architecture ouverte facilitant l'utilisation de MATLAB et de ses produits compagnons pour explorer des données et créer des outils personnalisés fournissant des aperçus rapides

Généralités

- Environnement de programmation pour le calcul scientifique
- Outil pour le calcul numérique, la visualisation de graphiques
- Permet d'élaborer rapidement des schémas de solution numérique.
- Langage interprété

- Outils inclus dans MATLAB
  - acquisition de données
  - analyse et exploration de données
  - visualisation et traitement d'images
  - prototypage et développement d'algorithmes
  - modélisation et simulation
  - programmation et développement d'applications

L'environnement MATLAB



Entièrement configurable

Fenêtres graphiques

- Command Window
- Current directory
- Command history
- Workspace
- Help

Barres de menus

- File
- Edit
- View
- Go
- Favorites
- Debug
- Desktop
- Window
- Help

barres d'outils

- Icônes : fichiers, copier/coller, Simulink, Help…

Prise en main de MATLAB

- **Utilisation de MATLAB en calculette**
  - → Dans la fenêtre de commande (command window)
  - exemples
    - 4/5
    - 4/5;
    - [1,8]
    - [1 3;2 9]
    - pi
    - sin(pi)
    - sin pi
    - 7:2:20
    - 7:-2:-6
    - pi
    - format short
    - pi

Prise en main de Matlab

- **Utilisation en ligne de commande (variables)**
  - exemples
    - a=2/3
    - b=2/3
    - c=[1 3;2 5.2]
    - d=a*c
    - e=1:1:10
    - who
    - whos
- **Utilisation des aides**
  - exemples
    - helpwin
    - help sin
    - lookfor sin

Éléments de langage

- **Variables**
  - Nom : débute par une lettre (différence entre Majuscules et minuscules)
  - Pas de noms de variables réservés → prudence !
  - **PAS** de déclaration de type ou de taille des variables utilisées : le type est établi automatiquement à partir des valeurs affectées à la variable
  - 4 types de données
    - Réels
    - Complexes
    - Caractères
    - Logiques
  - Exemple

```
>> clear;
>> a=5.1;b=1+i;voyelles= 'aeiouy';
>> whos ;
 Name Size Bytes Class
 a 1x1 8 double array
 b 1x1 16 double array(complex)
 voyelles 1x6 12 char array
Grand total is 8 elements using 36 bytes
```

Éléments de langage

- **Les types de données** (help datatypes)
  - Les réels
    - Représentés par des nombres flottants ;
    - Pas de différence entre entier, « entiers longs », réels, « réels longs »
    - Notation décimale (+1234.5678, -1234.5678) ou scientifique (±1234e±5678)
  - les complexes (complex)
    - Constante désignant l'imaginaire i (i^2 =-1) : i ou j (Attention aux noms des indices de boucles) ;
    - Ecriture : a+ib , a+i*b , r*exp(iT) , r*exp(i*T) …
    - Fonctions utiles : imag , real , abs , angle ;
  - les chaînes de caractères (char)
    - Suites de caractères, encadrées par des apostrophes (') ;
    - Interprétées comme des tableaux de caractères (de composantes, chaque caractère de la chaîne) ;
    - Fonctions utiles : voir manipulation de tableaux
  - Le type Logique (logical)
    - 0 pour FAUX / 1 pour VRAI ;

Eléments de langage

- **Vecteurs et matrices**
  - Définition : liste des éléments entre crochets ([…])
  - Séparation des éléments d'une même ligne par , ou espace
  - Séparation des lignes par ; ou retour chariot
  - Extraction de sous-matrices :
    - A(i,j) : élément de la $i^{iéme}$ ligne et $ji^{ème}$ colonne de A
    - A(:,j) : $j^{ième}$ colonne de A
    - A(i,:) : $i^{iéme}$ ligne
    - A($i_{début}$:i_{fin},j) : éléments des lignes $i_{début}$ à i_{fin} de la $j^{ième}$ colonne
    - diag(A) : éléments de la diagonale de A
  - Création de matrices particulières
    - eye(n) : matrice identité de taille n
    - ones(n,m) : matrice de taille (n,m), de composantes 1
    - zeros(n,m) : matrice de taille (n,m), de composantes 0
    - rand(n,m) : matrice de taille (n,m), de composantes aléatoires (entre 0 et 1)
    - diag(v) : matrice diagonale, de diagonale égale au vecteur v

Éléments de langage

- Extraction de sous-vecteurs
  - v(k) : $k^{ième}$ élément du vecteur v
- Création de vecteurs particuliers
  - linspace(a,b,N) : crée un vecteur de taille N, de composantes
    $$a+(k-1)(b-a)/(N-1)$$
  - a:h:b : vecteur de composantes a, a+h, a+2h, …,a+Kh avec $a+Kh \leq b$

Eléments de langage

- **Polynômes**
  - Définition :

    un polynôme de degré n, $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ est défini par le vecteur $p = [a_n, a_{n-1}, \ldots, a_1, a_0]$;
  - Fonctions utiles
    - polyval : évaluation du polynôme p en des points donnés
    - poly : représentation canonique d 'un polynôme à partir de ses racines
    - roots : racines d'un polynôme
- **Variables spéciales et constantes (help elmat)**
  - ans : résultat le plus récent
  - pi=3.1415926535897
  - i (ou j) : unité imaginaire
  - eps : précision numérique relative
  - realmin : plus petit nombre flottant
  - realmax : plus grand nombre flottant
  - inf : infini, obtenu pour les expressions excédant realmax
  - NaN : Not A Number, obtenu dans les opérations d 'indétermination

  - **Attention !**

    le nom des constantes n 'est pas réservé

Éléments de langage

- **Opérateurs (help ops)**
  - Opérations matricielles : + , - , * , / , ^
    \ (division à gauche (x=a\b est la solution de ax=b)
  - Opérations terme à terme sur les tableaux : .* , ./ , .^
  - Opérateurs relationnels : == , ~= , < , > , <= , >=
  - Opérateurs logiques : & , | , ~ , xor
- **Fonctions opérant sur des Scalaires (help elfun) :**
  - entiers
    - rem : division entière
    - lcm : plus petit multiple commun
    - gcd :plus grand multiple commun
    - factor :décomposition en facteurs premiers
  - complexes
    - conj :conjugué
    - abs : module
    - angle : phase
    - real : partie réelle
    - imag : partie imaginaire

Éléments de langage

- **Fonctions mathématiques**

  log, log10, exp, sqrt, abs, sign, cos, acos, cosh, acosh, …
- **Fonctions d ’arrondi**
  - round : arrondi à l’entier le plus proche
  - floor : arrondi par défaut
  - ceil : arrondi par excès
  - fix : arrondi par défaut (resp. excès) pour un réel positif (resp. négatif)
- **Fonctions opérant sur des vecteurs**
  - cross : produit vectoriel
  - dot : produit scalaire
  - sum , prod , max , min , mean : somme, produit, maximum, minimum et moyenne des éléments
  - sort : tri par ordre croissant
  - any : 1 si au moins un des éléments du vecteur est non nul, et 0 sinon
  - all : 1 si tous les éléments du vecteur sont non nuls, et 0 sinon

Éléments de langage

- **Fonction opérant sur des matrices (help matfun)**
  - A’ : transposée de la matrice A
  - inv : inverse de la matrice
  - det : déterminant de la matrice
  - rank : rang de la matrice
  - norm : norme euclidienne de la matrice
  - eig : valeurs propres et vecteurs propres de la matrice
  - size : taille de la matrice A
  - poly : polynôme caractéristique d ’une matrice
  - trace : trace
  - expm : exponentielle de matrice
  - any / all : même fonctionnement que pour les vecteurs, mais travail par colonnes (par défaut) ou par lignes

Les structures d'enchaînement

- **boucle FOR**

  →Répétition d'une action un nombre déterminé de fois

  - Syntaxe

```
for variable = expression
 instruction
 ...
 instruction
End
```

  - Exemple

```
m=5;n=3;
for i = 1:m
 for j = 1:n
 H(i,j) = 1/(i+j);
 end
end
```

Les structures d'enchaînement

- **boucle WHILE**

  → Exécution "a priori" un nombre de fois inconnu.
  Peut boucler de manière infinie
  Incrémentation à la charge de l'utilisateur.
  Initialisation nécessaire
  Permet des conditions de test booléennes de bouclage complexes

  - Syntaxe

```
while test
 instruction
 ...
 instruction
end;
```

Le booléen `test` est la négation de la condition d'arrêt

Les structures d'enchaînement

- **Test**
  - →Réalisation d'une action sous condition

  - Syntaxe

```
if expression1
 instructions1
elseif expression2
 instructions2
else
 instructions3
end
```

  - Exemple

```
if a<b
 min=a
else
 min=b
end
```

Les structures d'enchaînement

- **Branchement à choix multiples**
  - Syntaxe

```
switch expression
 case valeur_1
    instructions1;
   case valeur_2
    instructions2;
   case {valeur_3,valeur_4,...,valeur_n}
    instructions3;
   otherwise
    instruction4;
   end;
```

Matlab

Fichiers de commandes

Fichiers de commandes

Matlab permet d'exécuter facilement une séquence de commandes (script) enregistrées dans un fichier

→ langage de programmation

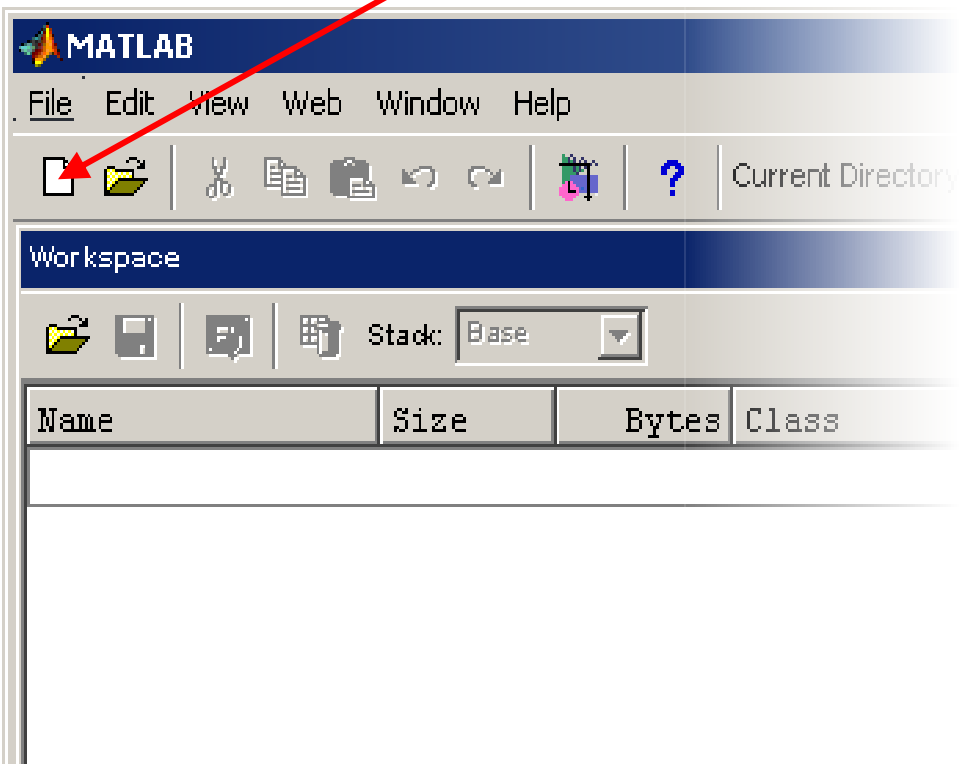
→ Avantages évidents pour :

- la sauvegarde du travail
- sa mise au point
- la rapidité d'exécution

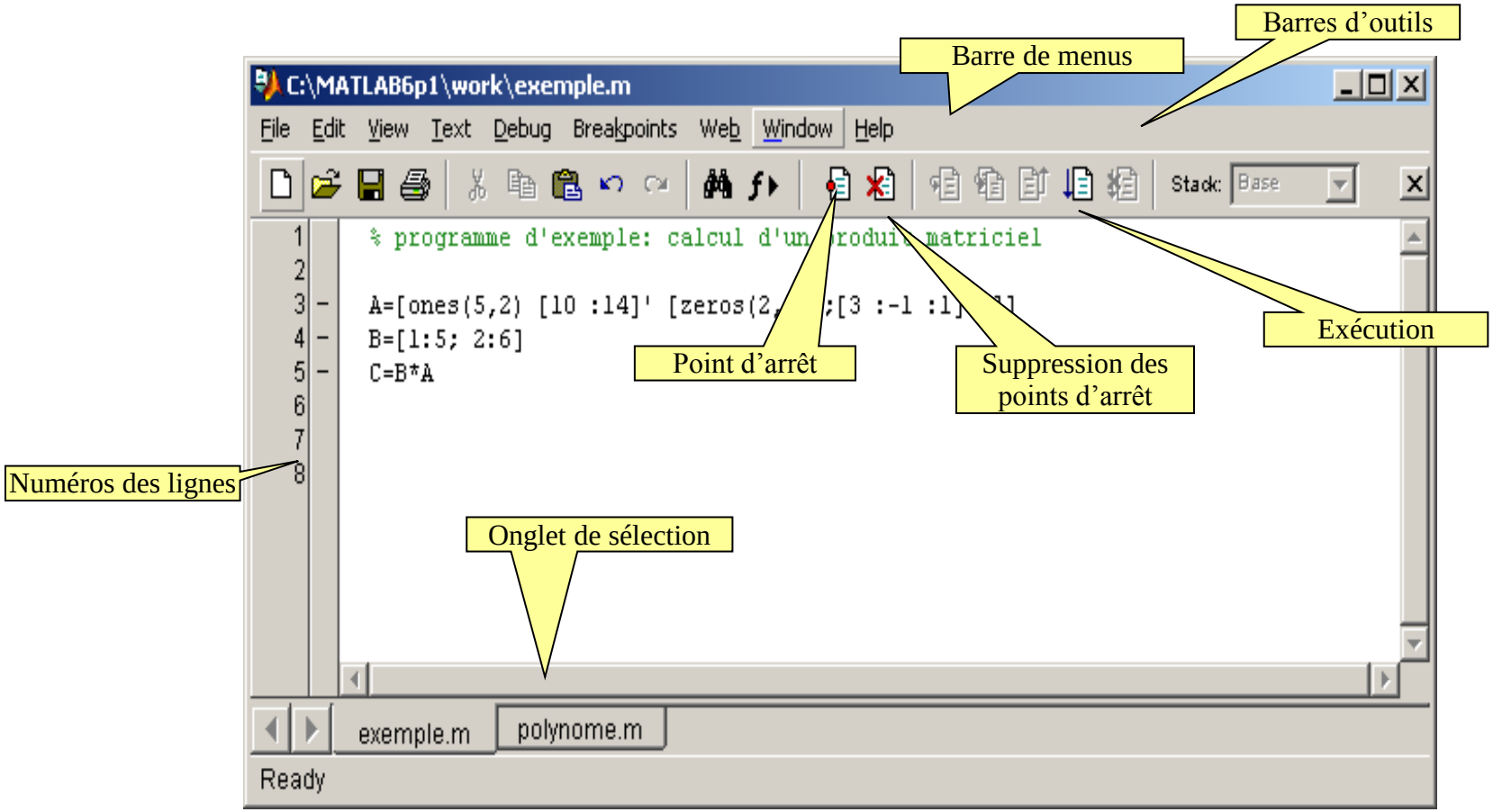
- Les fichiers qui contiennent les instructions Matlab sont appelés « M-files » et comportent une extension .m (ou **.mex**)
- Ils consistent en une succession de commandes telles qu'elles seraient entrées sous l'interpréteur Matlab
- On peut les créer, à l'aide de l'éditeur de Matalab ou de n'importe quel éditeur de texte, puis les utiliser de la même manière que toutes les autres commandes de Matlab

L'éditeur de texte de Matlab

- **Lancement**
  - À partir du menu File/New/M-file
  - Au clavier : CTRL + N
  - En cliquant sur l'icône



La fenêtre d'édition



La fenêtre d'édition

Fonctionnalités de l'éditeur

- Création et modification de fichiers textes
  - Indentation
  - Reconnaissance des éléments syntaxiques et mise en couleur
  - Contrôle des parenthèses, crochets, accolades…
  - Possibilité d'atteindre rapidement une ligne donnée (Edit/Go to line ou CTRL+G)
  - Marque-pages pour atteindre rapidement un endroit donné (Edit/Set/clear bookmark, previous bookmark, next bookmark)
  - Recherche et remplacement de texte (Edit/Find and replace ou CTRL + F)
- Sauvegarde et impression
- Exécution et mise au point
  - Lancement (icône exécution ou Debug/Run ou F5)
  - Points d'arrêt (menu Breakpoints)
  - Exécution en pas à pas (Debug/Step, Step in, Step out)

Exemple de programme

Fichier de script exemple.m

```
% programme d'exemple : calcul
% matriciel
%
v1=[2 -1 4]
v2=[1:4]
v3=v2'
A1=[1 2;3 4]
A1(1,:)
A1(:,2)
A2=[2 -1 0;3 4 3]
A2(1:2,1:2)
eye(4)
zeros(5,2)
M1=diag(2:4)
M2=diag(1:2,1)
M3=diag([-1 -2],-1)
M4=M1+M3-diag(3:4,1)
```

% indique une ligne de commentaire

- **Exécution**
  - dans la fenêtre de commande

    >> exemple
  - dans l'éditeur

    cliquer sur le bouton d'exécution ou F5 …

Fichiers de fonctions

- **Script**
  - Suite de commandes MATLAB (les mêmes que celles de la ligne commande)
  - Pas d'arguments d'entrée et de sortie
  - Variables utilisées : celles de l'espace de travail
- **Fichier de fonction**
  - Fichier permettant de programmer des fonctions avec des arguments d'entrée et de sortie
  - Passage des arguments par valeur
  - Syntaxe

```
function [ret1,…,retN] = nom_fonction (arg1,…,argP)
  Bloc d'instructions impliquant les arguments (d'entrée), et au
  moins une assignation aux arguments de retour
```

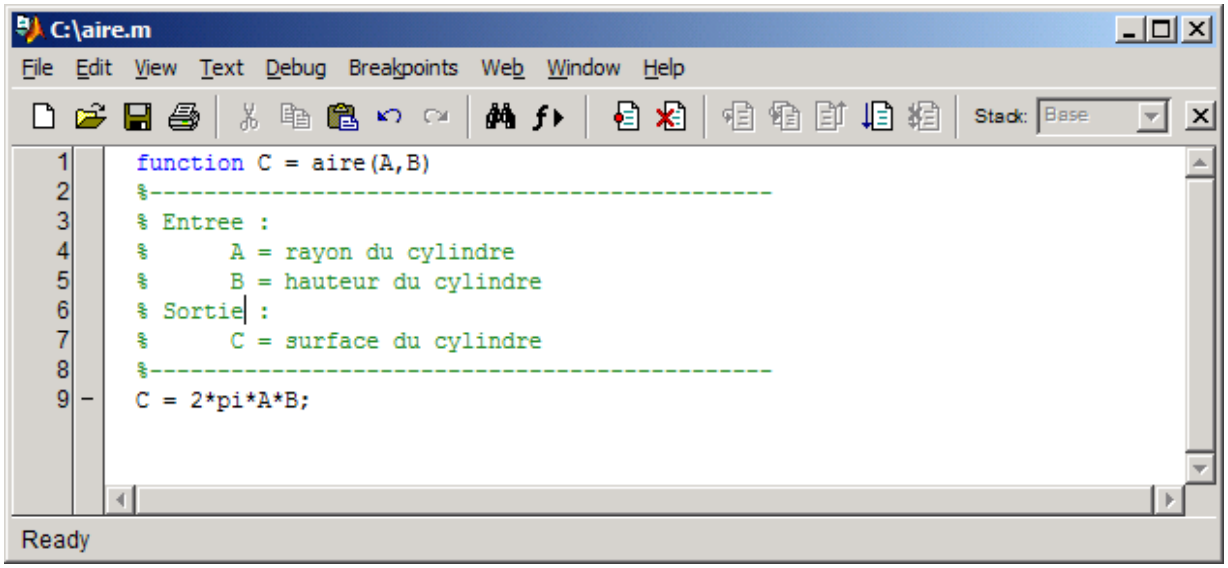
arg1, arg2,..., argP : valeurs fournies à la fonction comme données d'entrée (appelées arguments d'entrée)

ret1, ret2,..., retN : arguments de retour (ou de sortie) des résultats des opérations effectuées à l'intérieur de la fonction, et pouvant être assignés à une variable ou à plusieurs variables à l'extérieur de la fonction

Exemple

- Calcul de la surface d'un cylindre :
  - Arguments d'entrée : le rayon et la hauteur du cylindre ;
  - Variable de sortie : la surface, égale à 2π fois le rayon par la hauteur.

- Fichier aire.m



- Appel

```
>> R = 1.2
R =
 1.2000
>> H = 12.14
H =
 12.1400
>> S = aire(R,H)
S =
 91.5334
>> aire(3,10)
ans =
 188.4956
```

Matlab

Graphiques

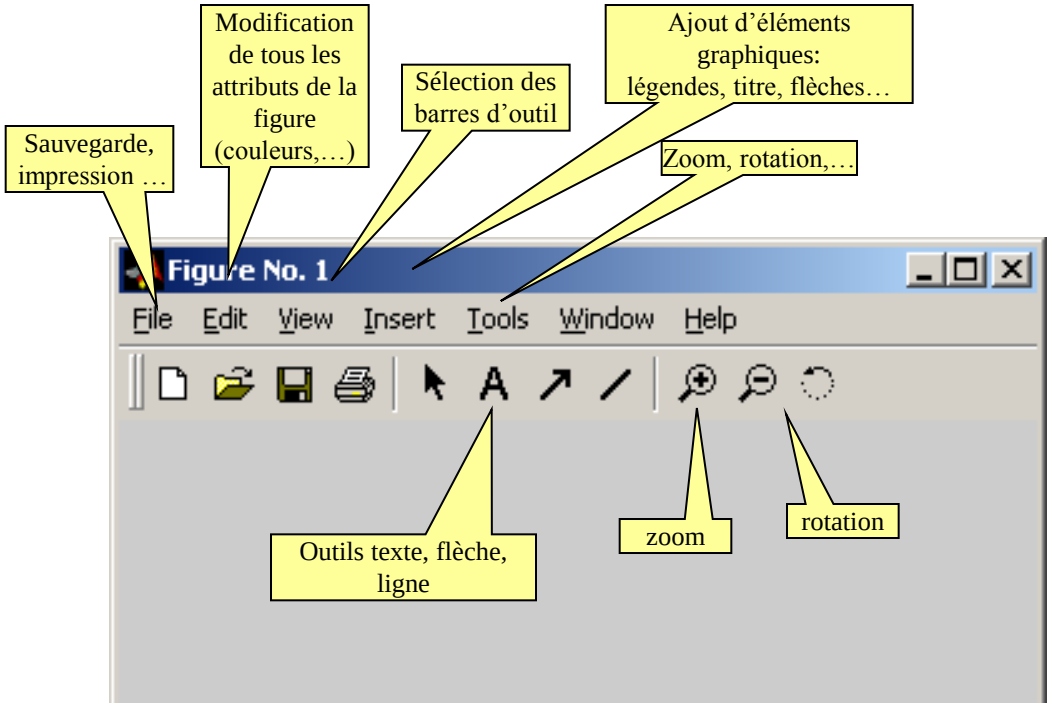
Fenêtre graphique

L'ouverture d'une fenêtre graphique se fait à l'aide de la commande `figure`

`figure` → ouvre une nouvelle fenêtre

`figure(2)` → ouvre la figure N°2 ou l'active si elle existe déjà

Les menus et boutons permettent de modifier interactivement tous les attributs du graphique (style, couleurs, rendu, point de vue…). Ils peuvent aussi être définis par le programme.



Tracé de graphes multiples

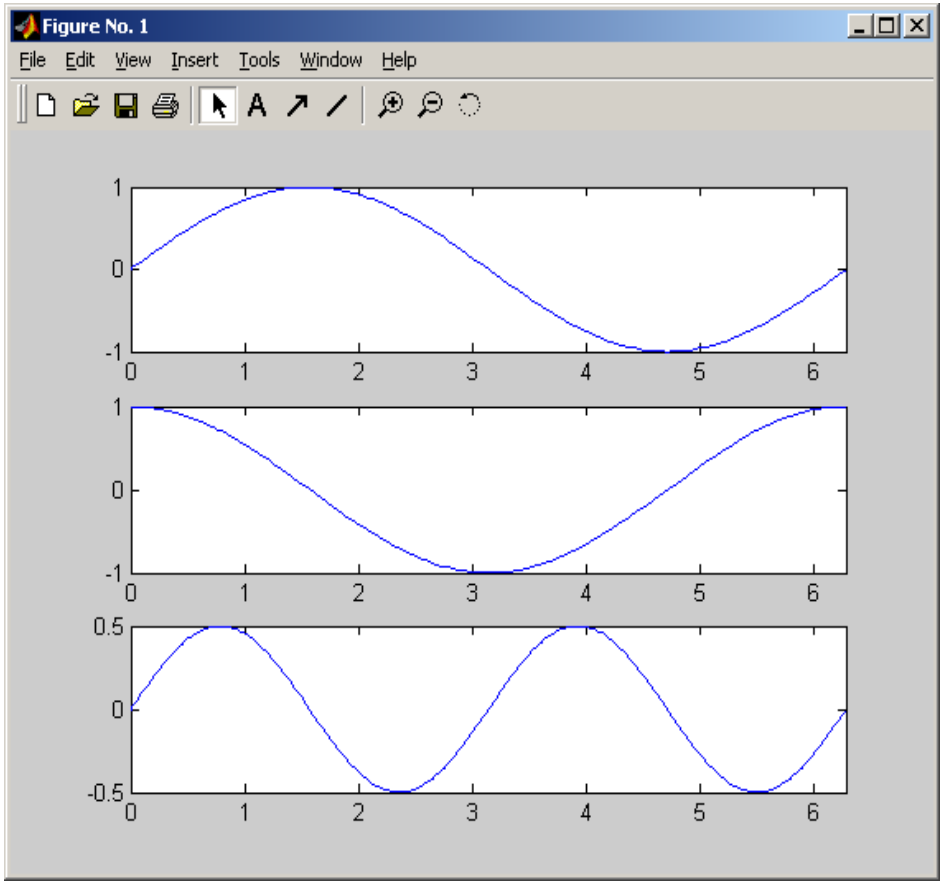
- La fenêtre graphique peut être subdivisée en plusieurs zones de tracé par :

```
subplot(m,n,p)
```

```
m:lignes
n: colonnes
p:graphe courant, p ∈[1,nxm]
```

Exemple

```
figure(1)
subplot(3,1,1);
fplot('sin(x)',[0,2*pi])
subplot(3,1,2);
fplot('cos(x)',[0,2*pi])
subplot(3,1,3);
fplot('sin(x)*cos(x)',[0,2*pi])
```



Tracé de graphes multiples

- Plusieurs graphes peuvent être superposés dans la même zone de tracé par :

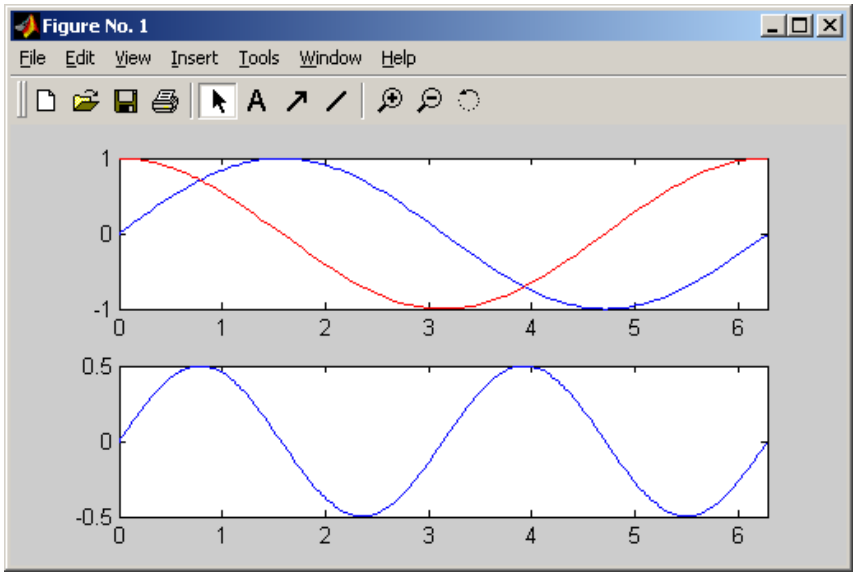
```
hold on
```

- Pour effacer le graphe précédent lors d'un nouveau tracé, on emploie :

```
hold off
```

Exemple

```
figure(1)
subplot(2,1,1);
fplot('sin(x)',[0,2*pi])
hold on;
fplot('cos(x)',[0,2*pi],'r');
hold off;
subplot(2,1,2);
fplot('sin(x)*cos(x)',[0,2*pi])
```



Les types de graphiques

Matlab permet une grande variété de représentations 2D ou 3D

- **Courbes 2D**

  `plot(X,Y) X,Y: vecteurs de coordonnées`

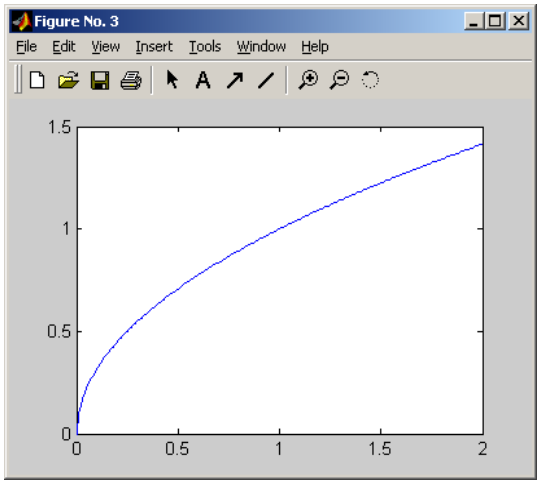
  et aussi

  `stem, stairs, semilogx, semilogy,`

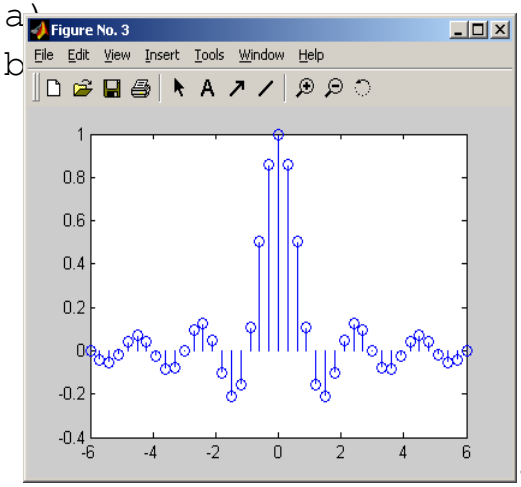
  `loglog, plotyy, errorbar,polar,…`

  exemples

```
figure
x=0:0.01:2
y=x.^0.5
plot(x,y)
```




```
figure
a=-6:0.3:6
b=sinc(a)
stem(a,b)
```



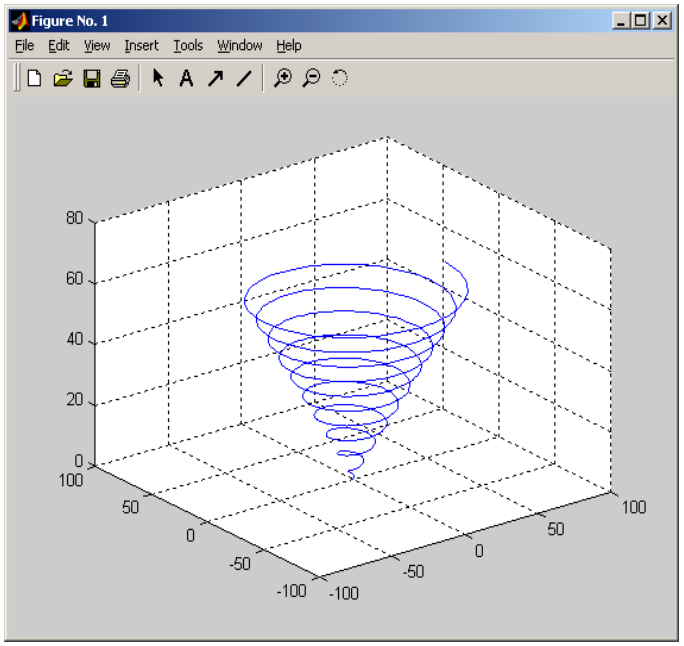
Les types de graphiques

- **Courbes 3D**

```
plot3(X,Y,Z)        X,Y,Z: vecteurs de coordonnées
```

exemple

```
figure
tetha=0:pi/50:20*pi
y=sin(tetha).*tetha
x=cos(tetha).*tetha
z=tetha
plot3(x,y,z)
grid
```



Les types de graphiques

- **Tracé de fonctions**

  `fplot('fonction', bornes)`

  et aussi

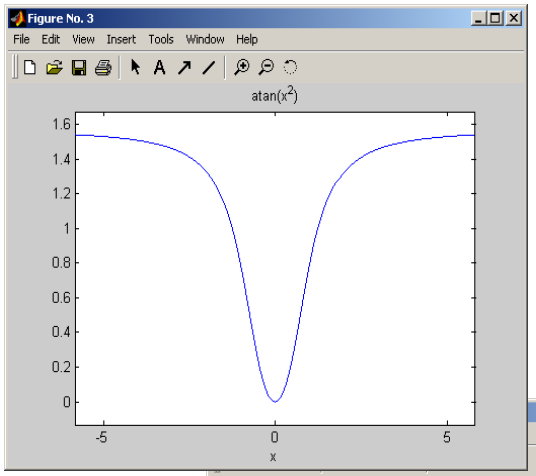
  `ezplot('fonction')`

  `ezpolar('fonction')`

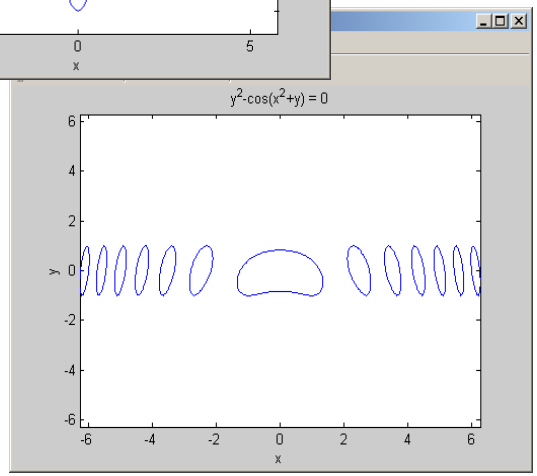
  `ezplot3('fonction')`

  exemples:

  `ezplot('atan(x^2)')`



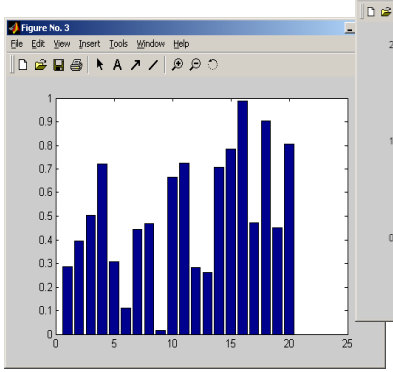
`ezplot('y^2-cos(x^2+y)')`

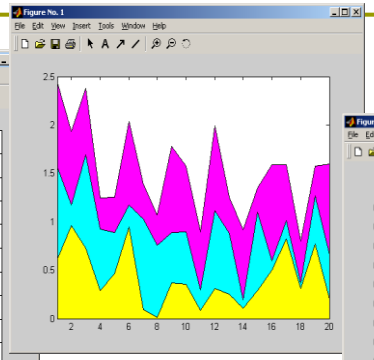


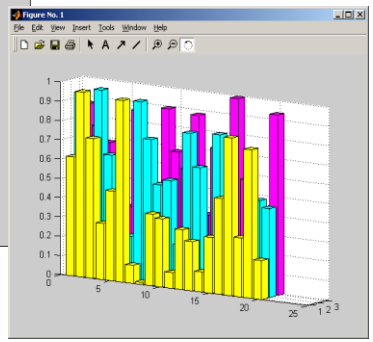
Les types de graphiques

- **Histogrammes**
  → bar, bar3, area

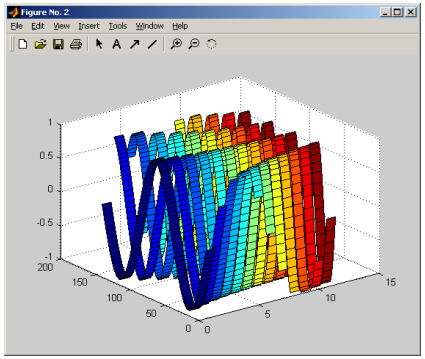
  voir aussi :
  `rose`



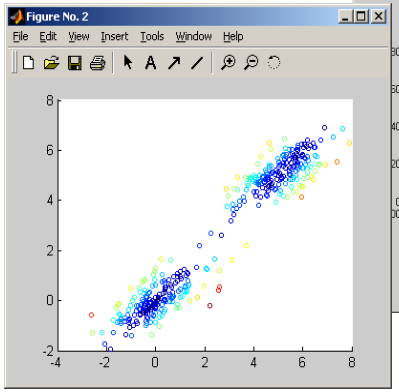


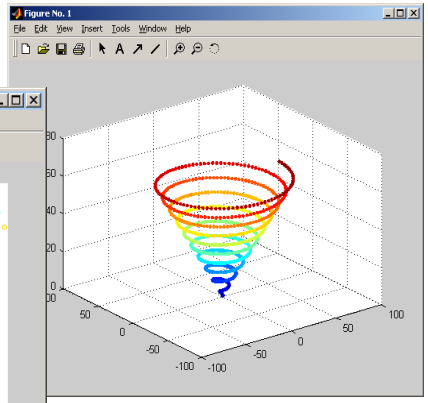


- **Rubans**
  →ribbon



- **Nuages de points 2D**
  →scatter, scatter3

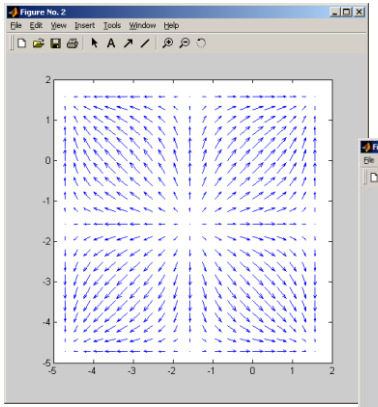


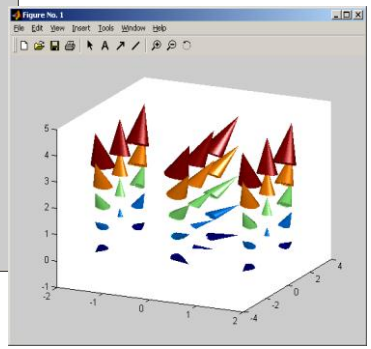


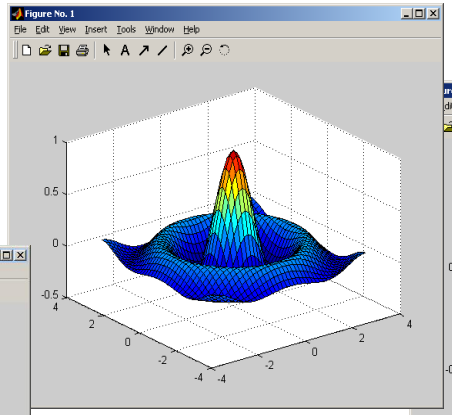
Les types de graphiques

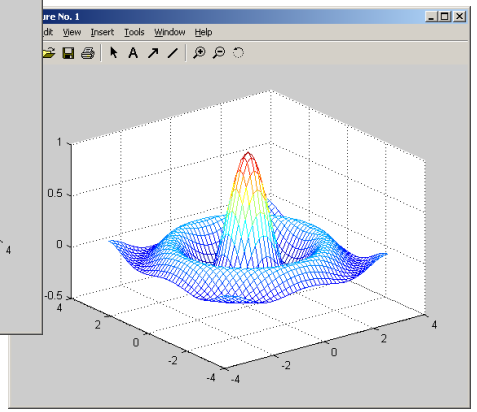
- **Champs de vecteurs 2D**
  → quiver,compass,feather
- **Champs de vecteurs 3D**
  → coneplot
- **Maillages et surfaces**
  → mesh,surf,…

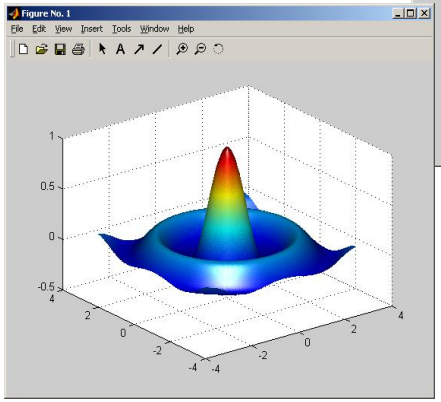
Pour des fonctions :
ezmesh,ezsurf,…





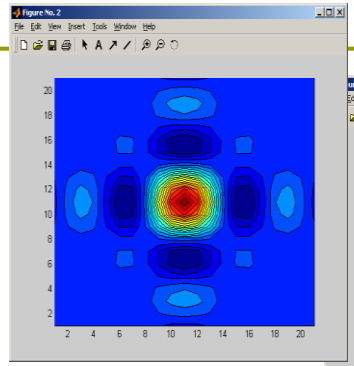


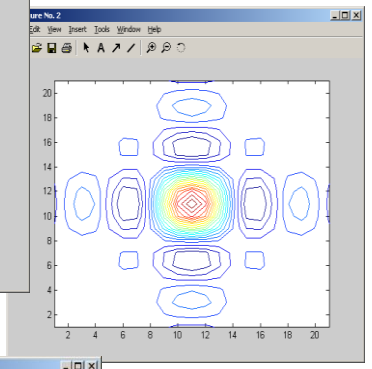




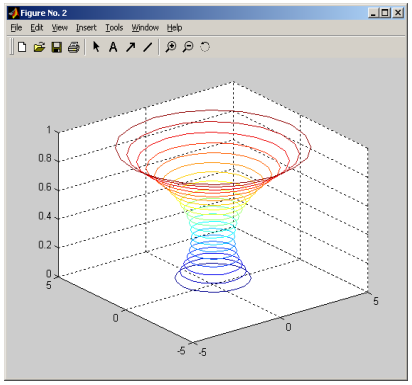
Les types de graphiques

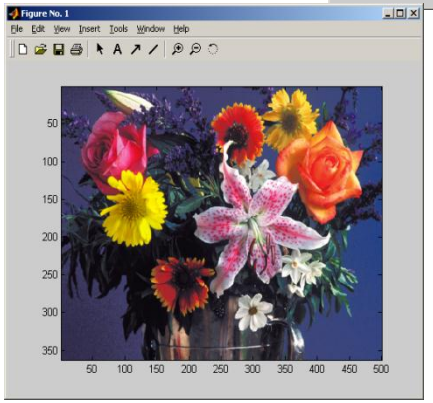
- **isocontours (courbes de niveau)**
  → contour, contourf, ezcontour,…
  et ezcontourf





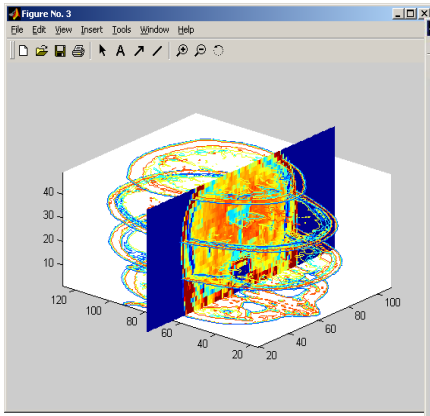
- **isocontours 3D**
  → contour3

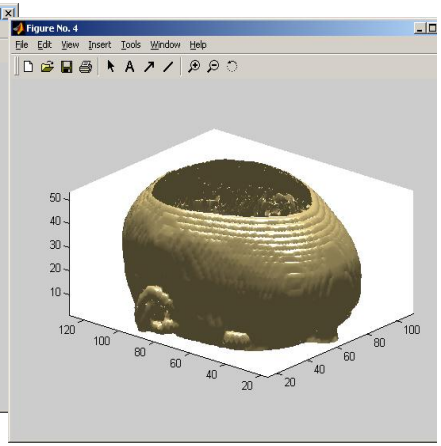




- **Images**
  →imagesc, imshow

- **Données volumiques**





Propriétés du graphique

- **axes**
  - Modification des bornes du tracé

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax
  zmin zmax])
axis auto
```

  - Suppression des axes

```
axis off
axis on
```

  - Rapport x/y

```
axis square
axis equal
axis normal
```

  - Ajout d'un quadrillage

```
grid on
grid off
```

titres et légendes

- Titre

```
title('chaîne de caractères')
```

- Légende d'axe

```
xlabel('chaîne de caractères')
ylabel('chaîne de caractères')
```

- Texte n'importe où dans la fenêtre

```
text(x,y,'chaîne de
  caractères')
```

- Légende

```
legend('chaîne de caractères',
  'chaîne de caractères',…)
```

Propriétés du graphique

- **Couleurs, styles de lignes et marqueurs**

  Certains attributs du graphique peuvent être spécifiés dans les tracés 2D.

  - syntaxe

    ```
    plot(x,y,'couleur_style_marqueur')
    ```

- **couleurs**

  'c' 'm' 'y' 'r' 'g' 'b' 'w' 'k'

  cyan magenta yellow red green blue white black

- **styles de lignes**

'-'	trait continu	
'--'	pointillés longs	
':'	pointillés courts	
'-.'	trait mixte	
'none'	pas de trait	

- **marqueurs**

'+'	
'o'	
'x'	
's'	
'd'	
'^'	
'v'	
'>'	
'<'	
'p'	
'h'	
'none'	

Matlab

Entrées-sorties

Entrées-sorties

Le transit de données (‘entrées’ ou ‘sorties’) avec Matlab peut se faire de 3 façons différentes:

- Opérations interactives à travers l’utilisation du clavier (entrées) et de l’écran (sorties)
- Lecture ou écriture d’un fichier de données
- Sauvegarde de variables par l'utilisation des fonctions `save` et `load`

Entrées-sorties interactives

- **Lecture en mode interactif (clavier)**

  `n=input('Entrez la valeur de l'entier: ')`

  Affiche au clavier la chaîne de caractères :

  `'Entrez la valeur de l'entier: '`

  et attend que l'utilisateur entre la valeur. Ensuite, la donnée est stockée dans la variable n

- **Ecriture à l'écran avec fprintf**

  exemple : `fprintf('Le volume est: %12.5f.\n',vol)`

Chaîne de format

Chaîne de format :

- \n spécifie un passage à la ligne suivante
- % spécifie l'endroit dans la chaîne de sortie où doit être écrit la variable
- 12.5 écriture avec 5 chiffres après la virgule. Réserve 12 espaces pour écrire, et met des blancs si pas utilisés.
- f spécifie le format d'affichage. y a d'autres codes d'affichage

Code de format	Descriptif	Exemple
c	Caractère	a
d ou i	entier décimal, signé	-392
e	Notation scientifique, avec e	3.9265e2
E	Notation scientifique, avec E	3.9265E2
f	réel sans exposant	392.65
g	le plus court de %e ou %f	392.65
G	le plus court de %E ou %f	392.65
o	Octal signé	610
s	Chaîne de caractères	pipoule
u	Entier décimal non signé	7235
x	Entier hexadécimal non signé	7fa
X	Entier hexadécimal non signé (capitales)	7FA
p	adresse	B800:0000

Fichiers formatés

- **Ouverture du fichier**

```
fid=fopen(nom,mode)
fid,Message]=fopen(nom,mode)
```

fid : entier constituant un identifiant "poignée", qui permettra l'utilisation du fichier après ouverture (entier positif si succès, -1 sinon)

Message : message d 'erreur renvoyé si l 'ouverture n 'a pas pu se faire

nom : nom sur disque du fichier

Mode : chaîne pouvant prendre les valeurs suivantes :

- ' r ' : pour la lecture
- ' w ' : pour l'écriture (création si nécessaire)
- ' a ' : pour l'ajout (création si nécessaire)

- **Fermeture du fichier**

```
val = fclose(fid)
```

fid : identifiant correspondant au fichier à fermer

val : égal à 0 en cas de succès ou -1 en cas d 'échec

Lecture et écriture formatée dans un fichier

- **Ecriture**

  `fprintf(fid,chaîne_de_format,arg1,arg2,…)`

→ même fonctionnement que pour un affichage à l'écran

- **Lecture**

  `[a,compteur]= fscanf(fid,chaîne_de_format,taille)`

`fid` : identificateur du fichier à lire (qui doit avoir été ouvert par fopen)

`a` : matrice dans laquelle se trouve le résultat de la lecture

`compteur (optionnel)` : retourne le nombre d 'éléments lus avec succès

`taille (optionnel)` :

  - si **`taille`** n 'est pas précisé, le fichier est lu entièrement
  - si **`taille = n,`** n éléments sont lus et mis dans un vecteur colonne a
  - si **`taille = [m,n]`**, les mxn premiers éléments sont mis dans une matrice a de taille mxn, avec remplissage colonne par colonne.

- **`chaîne_de_format:`** chaîne de caractères permettant de préciser le mode de conversion utilise le caractère % suivi de caractères de conversion :
  - **`d`** : décimal ;
  - **`f`** : flottant fixe [-]mmm.nnnnn ;
  - **`e,`** : flottant scientifique [-]m.nnnnnnE[+/-]xxxx ;
  - **`c`** : caractère ;
  - **`s`** : chaîne de caractères.

Lecture et écriture binaire

- **Ecriture**

  `compteur = fwrite(fid,a)`

`fid` : identificateur du fichier à lire (qui doit avoir été ouvert par fopen)

`a` : matrice à écrire dans le fichier

`compteur (optionnel)` : retourne le nombre d ’éléments écrits avec succès

- **Lecture**

  `[a,compteur]= fread(fid,taille)`

`fid` : identificateur du fichier à lire (qui doit avoir été ouvert par fopen)

`a` : matrice dans laquelle se trouve le résultat de la lecture

`compteur (optionnel)` : retourne le nombre d ’éléments lus avec succès

`taille (optionnel)` :

- si **`taille`** n ’est pas précisé, le fichier est lu entièrement
- si **`taille = n,`** n éléments sont lus et mis dans un vecteur colonne a
- si **`taille = [m,n]`**, les mxn premiers éléments sont mis dans une matrice a de taille mxn, avec remplissage colonne par colonne.

Lecture et écriture de variables

- **Ecriture**

```
save fich_sauv a b c
save fich_sauv.txt a b c -ascii
```

fich_sauv : nom du fichier (fich_sauv.mat pour des fichiers binaires)
a, b, c : variables à sauvegarder
- ascii : option pour créer un fichier ascii lisible par d'autres applications

- **Lecture**

```
load fich_sauv
load fich_sauv a c
load fich_sauv.txt
```

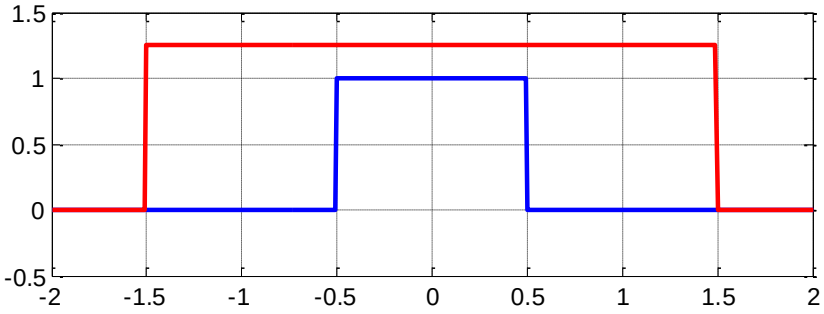
Matlab

Fonctions utiles en traitement du signal

Signaux utiles

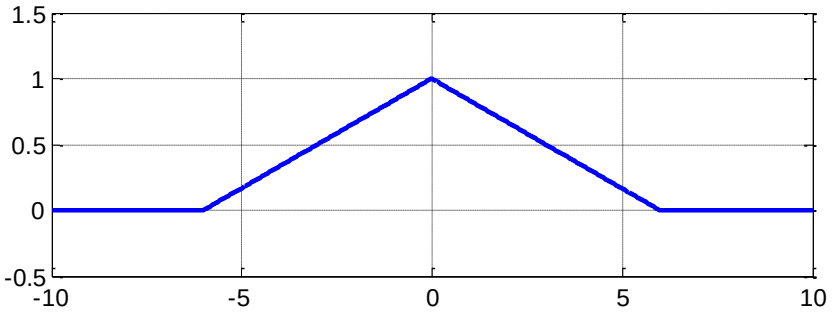
- **Porte**
  - rectpuls
  - Exemple

```
Te=0.01
t=-2:Te:2
x = rectpuls(t)
y = 1.25*rectpuls(t,3)
plot(t,x,t,y)
```



- **Impulsion triangulaire**
  - tripuls
  - Exemple

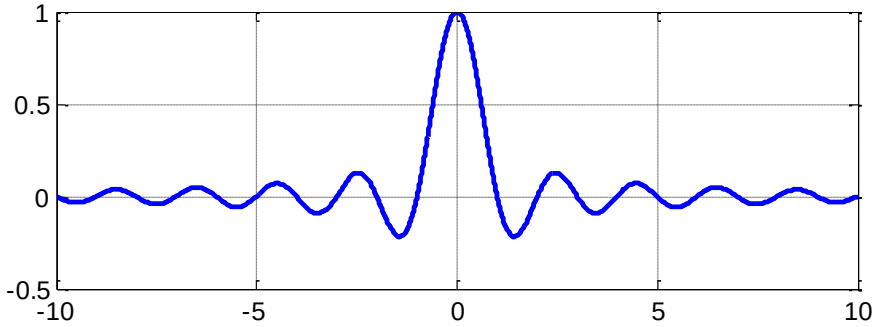
```
t=-10:Te:10
x = tripuls(t,12)
plot(t,x)
```



Signaux utiles

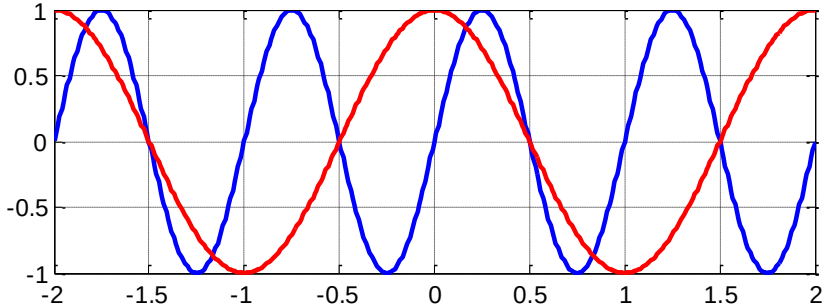
- **Sinus cardinal**
  - sinc
  - Exemple

```
Te=0.01
t=-10:Te:10
x = sinc(t)
plot(t,x)
```



- **Signaux périodiques**
  - sin, cos - signaux sinusoidaux
  - Exemple

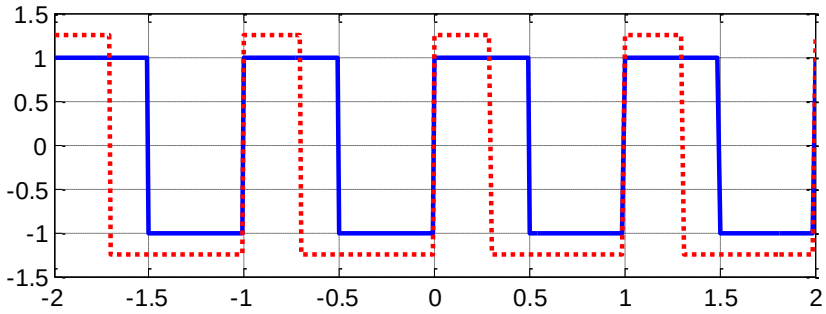
```
x = sin(2*pi*t)
y = cos(pi*t)
plot(t,x,t,y)
```



Signaux utiles

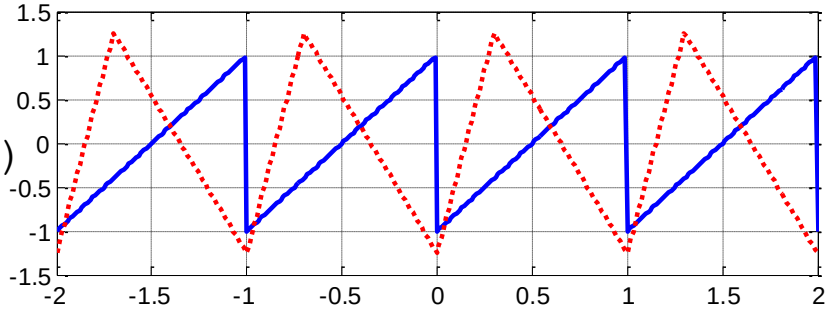
- **Signaux périodiques**
  - square - signal carré
  - Exemple

```
x=square(2*pi*t)
y=1.25*square(2*pi*t, 30)
plot(t,x,t,y,':')
```



  - sawtooth -signal en dents de scie
  - Exemple

```
x = sawtooth(2*pi*t)
y = 1.25*sawtooth(2*pi*t,30)
plot(t,x,t,y,':')
```

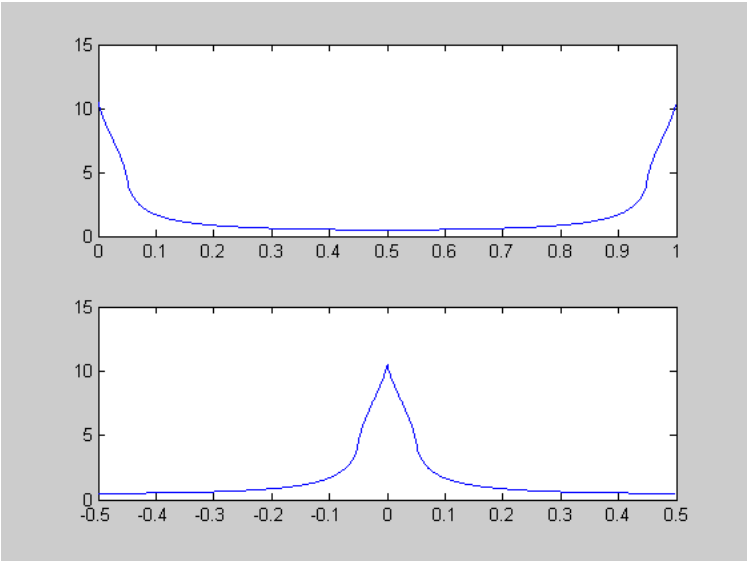


Représentation fréquentielle

- **Transformations**
  - fft - transformée de Fourier discrète
  - fftshift - permutation des hautes fréquences vers les fréquences négatives
  - ifft - transformée de Fourier discrète inverse

- Exemple

```
S=fft(s);
S2=fftshift(S);
subplot(211)
plot(f,abs(S))
subplot(212)
plot(f2,abs(S2))
```



Représentation fréquentielle

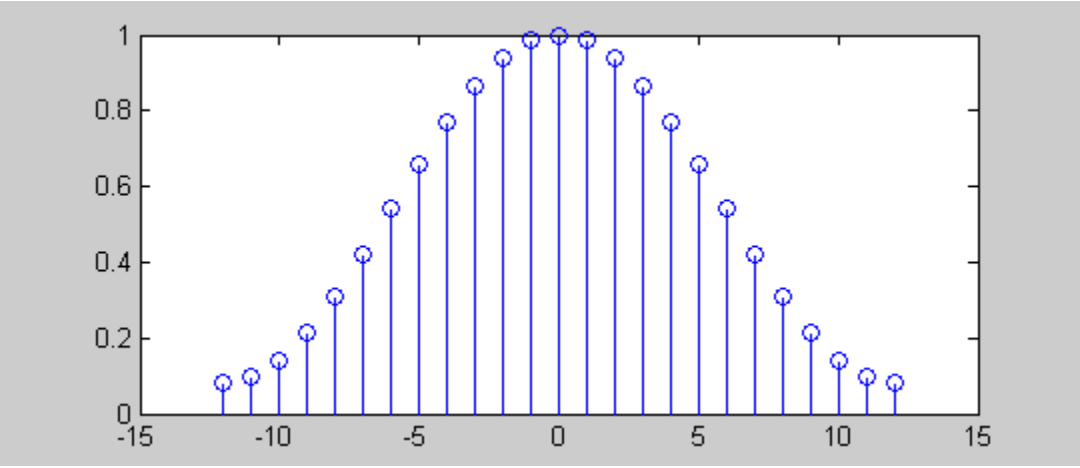
- **Fenêtres de pondération**

  - boxcar - fenêtre rectangulaire
  - triang - fenêtre triangulaire
  - hamming - fenêtre de Hamming
  - hann - fenêtre de Hanning
  - kaiser - fenêtre de Kaiser
  - bartlett - fenêtre de Bartlett
  - blackman - fenêtre de Blackman

Représentation fréquentielle

- Exemple

```
>> N=25;
>> k=-(N-1)/2:(N-1)/2;
>> s=hamming(N);
>> stem(k,s)
```



Filtrage numérique

- **Filtres à réponse impulsionnelle infinie**
  - Synthèse des filtres
    - butter - filtre de Butterworth
    - cheby1 - filtre de Tchebycheff de type I (ondulation dans la bande passante)
    - ellip - filtre elliptique
  - calcul de l'ordre
    - buttord - ordre pour un filtre de Butterworth
    - cheb1ord - ordre pour un filtre Tchebycheff de type I
    - ellipord - ordre pour un filtre elliptique

Filtrage numérique

- Exemple

```
Wp=0.5;                                  % fréquences normalisée
Ws=0.8;
Rp=3;
Rs=40;
[N, Wn] = buttord(Wp, Ws, Rp, Rs) %ordre et fréquence de
                                           coupure
[b,a]=butter(N,Wn)                       % coefficients
```

  - Résultats

```
N = 5
Wn = 0.5642
b =    0.0849    0.4247    0.8494    0.8494    0.4247    0.0849
a =    1.0000    0.6325    0.7735    0.2281    0.0776    0.0063
```

Filtrage numérique

- **Filtrage**

  - filter

    filtrage d'un signal. Le filtre est défini par les coefficients a_n et b_m de la fonction de transfert en z, en supposant $a_0=1$

$$y(k)+\sum_{n=1}^{N} a_n.y(k-n)=\sum_{m=0}^{M} b_m.x(k-m)$$

$$y(k)=b_0x(k)+b_1x(k-1)+\cdots+b_Mx(k-M)$$

$$-a_1y(k-1)-\cdots-a_Ny(k-N)$$

Filtrage numérique

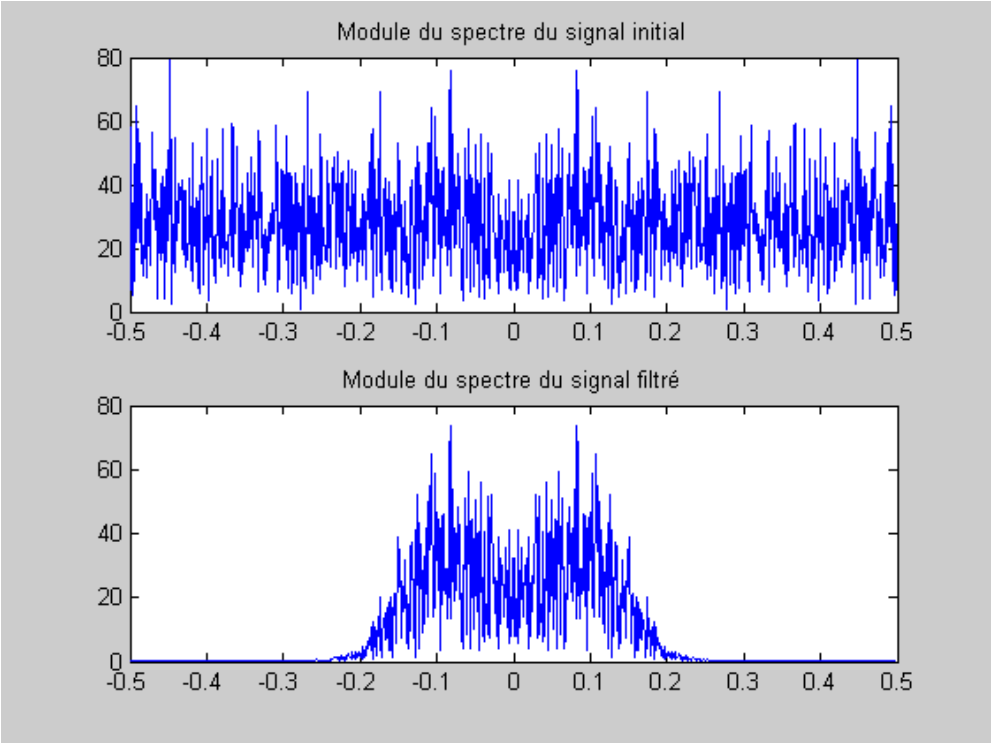
- exemple : filtrage d'un signal aléatoire large bande

```
Wp=0.3; Ws=0.5;Rp=3;Rs=40;N=1024;
t=(0:N-1);f=(0:N-1)/N;f=(-N/2:(N-1)/2)/N;

[n, Wn] = buttord(Wp, Ws, Rp, Rs)
[b,a]=butter(n,Wn)

s=randn(N,1)
v=filter(b,a,s);

S=fftshift(fft(s));
V=fftshift(fft(v));
subplot(211)
plot(f,abs(S))
subplot(212)
plot(f,abs(V))
```



Filtrage numérique

- **Calcul de la réponse impulsionnelle**

  - impz

    Renvoie la réponse impulsionnelle du filtre. Le filtre est défini par les coefficients a_n et b_m de la fonction de transfert en z, en supposant $a_0=1$

Remarque

```
>>h=impz(b,a,N)
```

est équivalent à :

```
>>imp=[1, zeros(1,N-1)];
>>h=filter(b,a,imp)
```

Filtrage numérique

- **Calcul de la réponse en fréquence**

  - freqz

    Renvoie la réponse fréquentielle du filtre. Le filtre est défini par les coefficients a_n et b_m de la fonction de transfert en z, en supposant $a_0=1$

Remarque

```
>>[S,f]=freqz(b,a,N,Fe);
```

est équivalent à :

```
>>f=(0:N-1)*Fe/N;
>>v=impz(b,a,2*N)
>>S=fftshift(fft(v));
>>S=S(N+1:2*N)
```