Table des matières

1) Environnement Matlab
2) Les variables en Matlab
3) Formats d'affichage
4) Fonctions prédéfinies en matlab
5) Programmation avec Matlab
6) Vecteur, Matrice, Cellule et structure
7) Fonctions en Matlab
8) Intégration numérique
9) Corrélations et régression
10) Entrées et sortie fichiers
11) Graphiques

Environnement Matlab

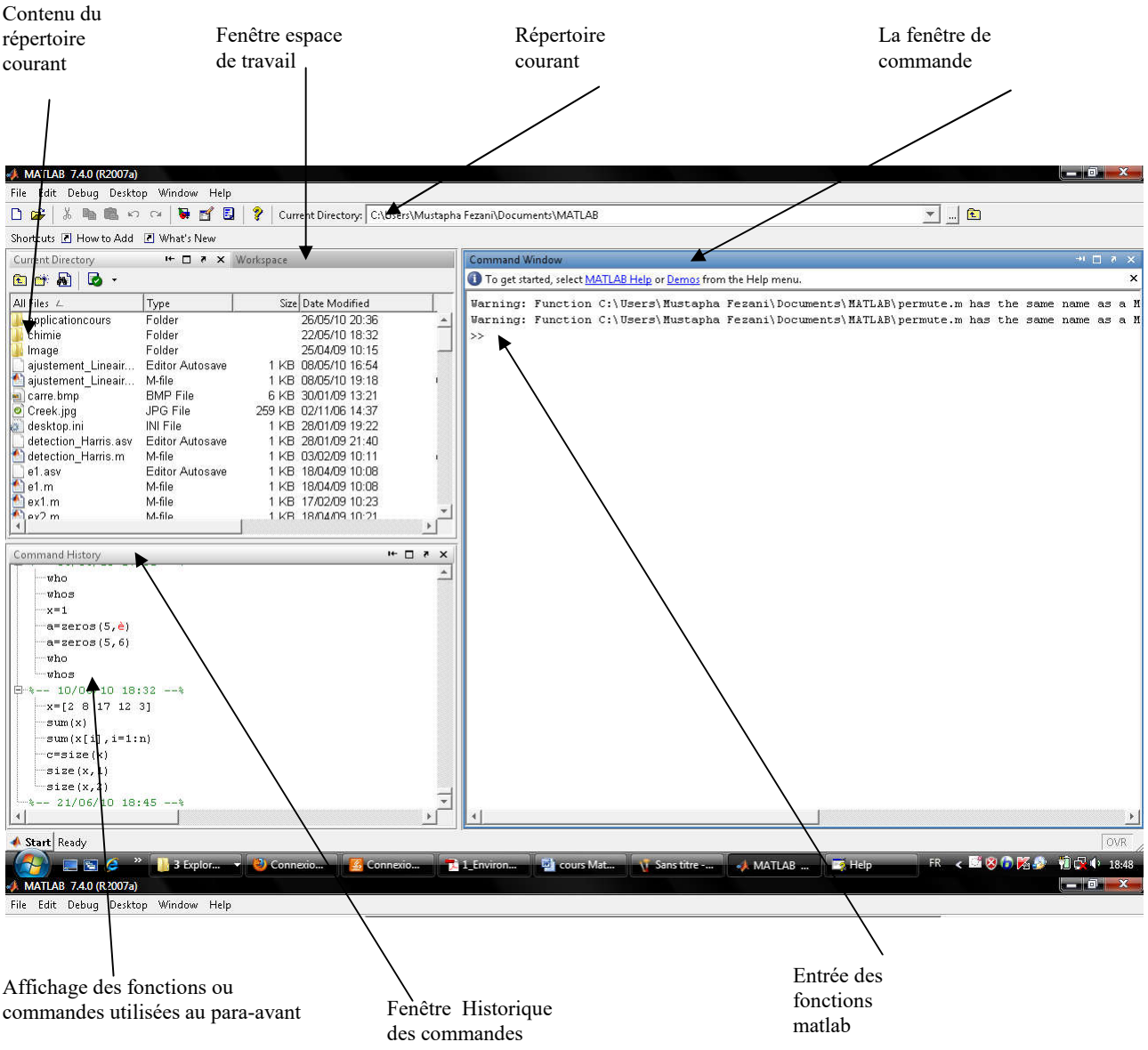
Le nom matlab dérive du terme Matrix laboratory, il s'agit d'un outil de développement pour des problèmes scientifiques et, plus généralement, pour tous les domaines ou des ou des calculs numériques importants doivent être faits.

La simplification suffisante d'un problème et l'application des principes fondamentaux appropriés est la modélisation et la description mathématique qui en résulte, le modèle mathématique.

Espace de travail de Matlab

Matlab travaille essentiellement avec des ensembles rectangulaires ou carrés (matrices) de données, les éléments pouvant être réels ou complexes. Une quantité scalaire sera alors une matrice ne contenant qu'un seul élément.

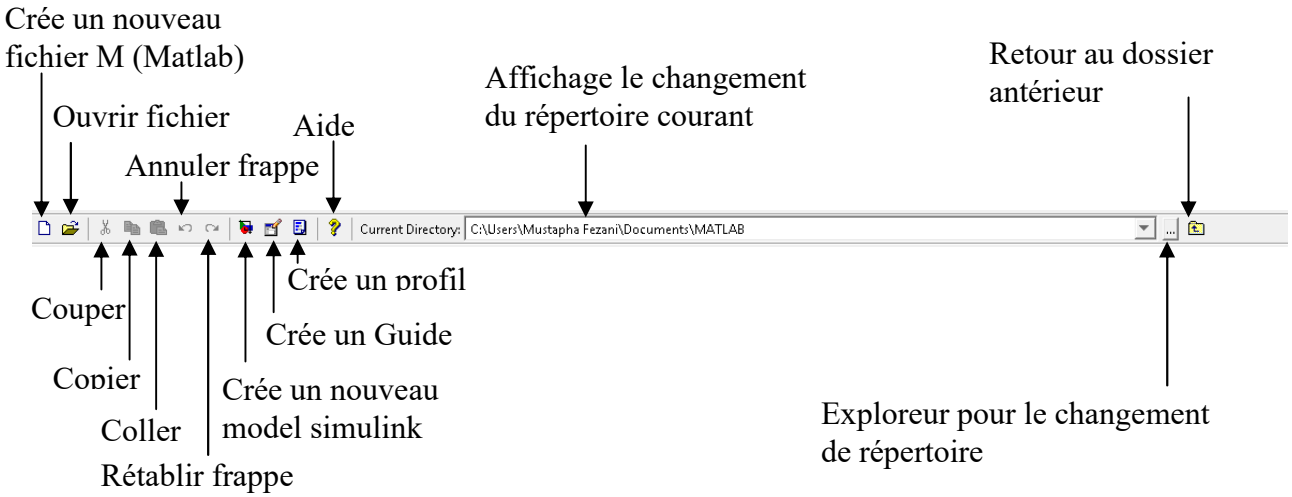
Au démarrage de Matlab, le bureau Matlab apparaît, contenant des outils (interfaces graphiques) pour gérer les fichiers, variables et applications liés à matlab.



Il existe quatre fenêtres :

- La fenêtre de commande (Command window), ou l'on exécute des fonctions Matlab directement ou des fichiers crées par l'utilisateur ;
- La fenêtre de suivi ou historique de commande (Command History) qui permet de visualiser toutes les variables ou fonctions utilisées au cours d'une session Matlab ;
- La fenêtre de l'espace de travail (Workspace) où sont visualisées toutes les variables utilisées dans la session ouverte (taille pour les matrices, valeurs pour les constantes) ;
- La fenêtre du répertoire courant (Current Directory) qui permet de visualiser les fichiers et matrices stockés dans votre répertoire.

Toutes les fenêtres peuvent être fermées ou ouvertes selon les besoins de l'utilisateur.
La barre d'outils



Les menus du bureau Matlab donnent aussi accès à d'autres opérations :

Menu File	
New	Ouvre une boîte de dialogue qui permet de créer un nouveau fichier programme, un M-file, en utilisant l'éditeur de texte (ou d'autre fichiers comme un fichier graphique (ouverture d'une fenêtre graphique), une interface utilisateur GUI (création d'icônes, de boutons, etc…)).
Save	Enregistrement du fichier courant
Open	Ouvre une boîte de dialogue qui permet d'ouvrir dans l'éditeur un fichier programme M-file existant.
Close	Ferme l'espace de travail
workspace	Importer les données
Import Data	Ouvre une boîte de dialogue qui permet de sauvegarder les
Save	noms et les valeurs des variables d'un fichier dont le nom a
Workspace as	été sélectionné
	Spécifie l'adresse disque où le fichier à exécuter est stocké.
Set Path	Ouvre une boîte de dialogue qui permet de personnaliser

Preference	l'espace de travail (réglage des formats numériques, de l'éditeur, des barres d'outils, de la police et taille de caractères,…).
Page setup	Mise en page
Print	Imprime le texte de l'écran de commande
Exit MATLAB	Permet de sortir de Matlab
Menu Edit	
Undo	Annule frappe
Redo	Rétablit frappe
Cut	Coupe le texte sélectionné
Copy	Copie le texte sélectionné pour collage ultérieur
Paste	Insère un texte copié
Paste to Workspace	Insère un texte copié dans l'espace de travail
Select All	Sélectionner tous
Delete	Supprimer
Find	Chercher un mot ou phrase
Find file	Chercher un fichier
Clear command Window	Initialiser la ligne de commande Window
Clear command History	Initialiser la ligne de commande History
Clear Workspace	Initialiser l'espace de travail
Menu View	
Permet d'ouvrir ou de fermer les différentes fenêtres accessibles dans le bureau et de régler certaines options d'affichage	
Menu Window	
Permet de passer rapidement d'une fenêtre à l'autre	
Menu Help	
Matlab Help	Ouvre la fenêtre de Matlab (présentation HTML avec arborescence dans le cadre de gauche pour un accès raide aux sujets recherchés).

Pour exécuter le programme, il faut spécifier le chemin d'accès aux fichiers source. Avec les récentes versions de Matlab, l'exécutable sait retrouver le répertoire du fichier à exécuter, si tel n'est pas le cas et que Matlab refuse obstinément d'exécuter votre fichier, il vous faut spécifier les chemins d'accès. Pour cela deux options sont possibles :

- Au début de chaque session Matlab, spécifier le chemin d'accès en sélectionnant la commande search Path … dans le menu file.
- Au début de chaque session Matlab, taper dans la fenêtre de commande l'instruction suivante :

path (path,'c :\repertoire\') ;

Exemple:
Sur mon ordinateur le répertoire de travail est

C:\Users\Mustapha\Documents\MATLAB
>> path (path,'C:\Users\Mustapha\Documents\MATLAB') ;

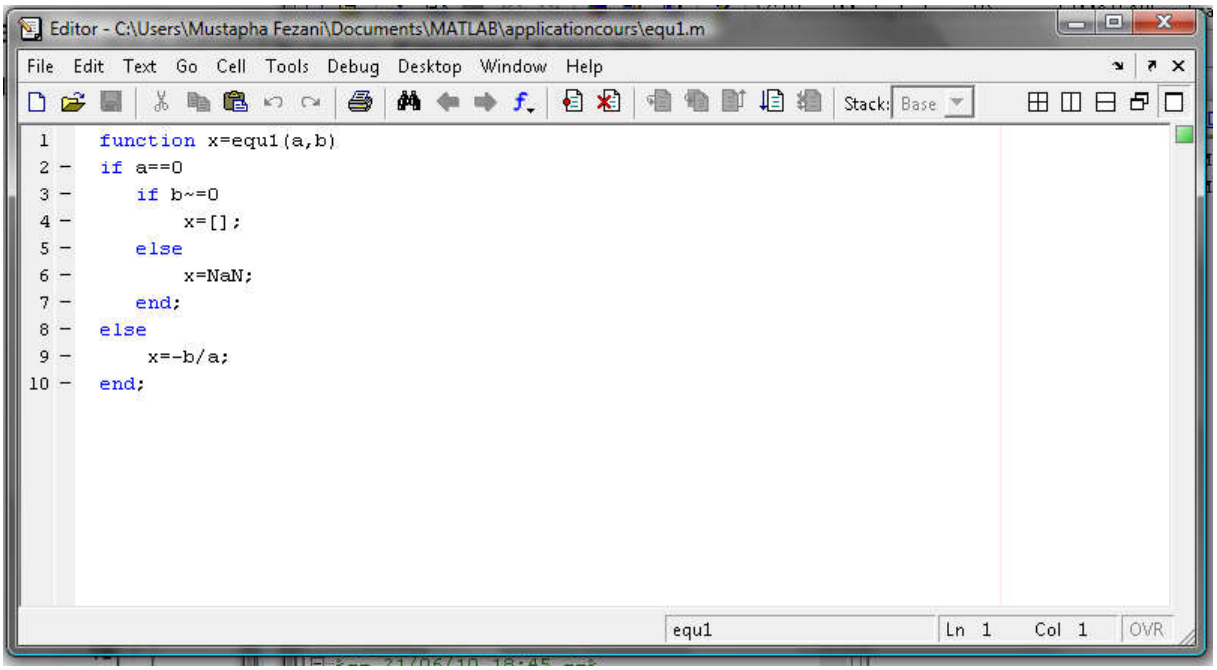
Search Path

Matlab utilise un chemin de recherche (”search path”) pour trouver les M-fichiers et les autres fichiers reliées, qui sont organisés dans des répertoires. Tout fichier que l’on peut exécuter en Matlab doit résider dans le répertoire courant ou dans un répertoire qui est dans le "**search path**". Il faut donc ajouter les répertoires qui contiennent les fichiers qu’on veut pouvoir exécuter au ”search path”. Par défaut, les fichiers fournit par Matlab sont inclus dans de ”search path”.

Pour voir ou éditer les répertoires contenus dans le "**search path**", sélectionner ”Set Path” dans le menu ”File” du bureau, et utiliser la boite de dialogue. On peut aussi utiliser les fonctions path pour voir le ”search path”, **addpath** pour ajouter des répertoires au chemin d’accès et **rmpath** pour ôter des répertoires.

Fenêtre de l’éditeur

A l’ouverture d’un fichier source.m (m-file), l’éditeur-debugger s’ouvre :



La fenêtre de l’éditeur, qui permet d’écrire un programme. Pour ouvrir cet éditeur, il faut chercher dans le menu file de Matlab, la commande New suivie de M-files si c’est un nouveau programme ou open M-files si l programme est déjà existant. Une fois sauvegardée, avait une extension .m, le programme sera exécuté dans la fenêtre de commande en tapant le nom du fichier source sans l’extension.

La syntaxe est mise en exergue : **les fonctions préprogrammées** de Matlab sont en bleu, les opérations et variables en noir, les commentaires en vert, le texte entre cotes en rouge, les erreurs en violet. L’indentation est réglée par défaut. Chaque ligne est numérotée, ce qui permet de repérer plus rapidement les erreurs détectées à l’exécution.

De la même façon que dans la fenêtre de commande, une barre d'outils permet l'accès aux opérations les plus fréquentes (ouvrir un fichier, créer un fichier, copier, coller, couper, enregistrer, etc…)

1 Fonction "HELP"

Pour obtenir de l'aide sur un sujet, une instruction ou une fonction, on tape help suivi par le sujet, l'instruction ou la fonction désirée.

Exemple 1:
» help atan2
atan2 Four quadrant inverse tangent.
atan2(Y,X) is the four quadrant arctangent of the real parts of the elements of X and Y. -pi <= atan2(Y,X) <= pi.

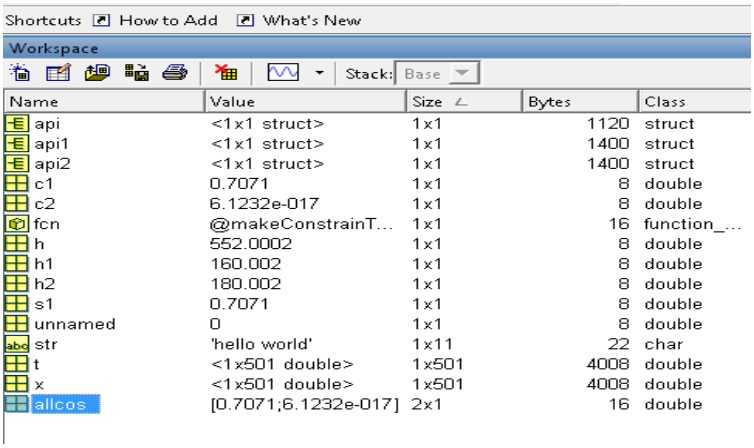
See also atan.

Overloaded functions or methods (ones with the same name in other directories) help darray/atan2.m

Reference page in Help browser doc atan2

<u>Autre façon</u>
» Lookfor atan2
ATAN2 Four quadrant inverse tangent.
ATAN2 Overloaded for distributed arrays

Espace de travail (Workspace)
Ce navigateur consiste en l'ensemble des variables (nommés arrays) utilisées durant une session Matlab et stockées dans la mémoire. On ajoute des variables dans le workspace (espace de travail) en utilisant des fonctions, en exécutant des M-fichiers, et en chargeant des workspaces préalablement sauvés.



Name	Value	Size	Bytes	Class
api	<1x1 struct>	1x1	1120	struct
api1	<1x1 struct>	1x1	1400	struct
api2	<1x1 struct>	1x1	1400	struct
c1	0.7071	1x1	8	double
c2	6.1232e-017	1x1	8	double
fcn	@makeConstrainT...	1x1	16	function_...
h	552.0002	1x1	8	double
h1	160.002	1x1	8	double
h2	180.002	1x1	8	double
s1	0.7071	1x1	8	double
unnamed	0	1x1	8	double
str	'hello world'	1x11	22	char
t	<1x501 double>	1x501	4008	double
x	<1x501 double>	1x501	4008	double
allcos	[0.7071;6.1232e-017]	2x1	16	double

Pour voir le workspace et des informations sur chaque variable, utiliser le navigateur workspace, ou utiliser les fonctions who et whos. Pour effacer des variables de l'espace de

travail, sélectionner la variable et choisir Delete dans le menu d'édition. On peut aussi utiliser la commande clear.
Le workspace est effacé à la fin d'un session Matlab. Pour sauver son état courant et ainsi pouvoir repartir directement en l'état après un redémarrage de Matlab, il faut utiliser soit "Save Workspace" du menu "File", soit la commande save. Ceci sauve toutes les variables dans un fichier binaire appelé un fichier MAT, qui a une extension .mat. Pour relire ce type de fichier, utiliser soit "Import Data" du menu "File", soit la fonction load

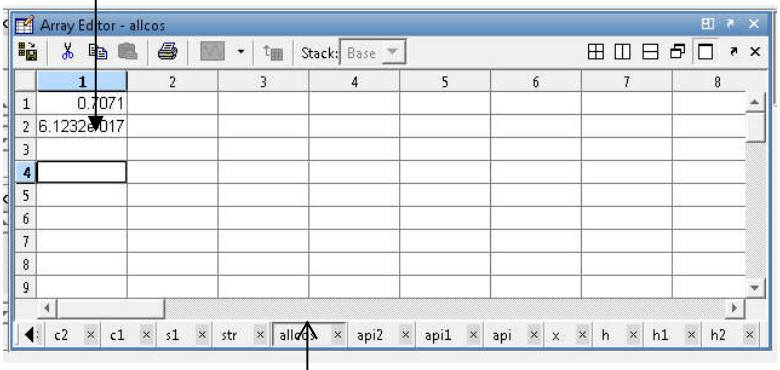
Exemple :
fichier ' work.m'

```
s1 = sin(pi/4);
c1 = cos(pi/4); c2 = cos(pi/2);
str = 'Bonjour tous le monde'; % C'est une chaine
save                    % sauve toutes les variables
                        % dans un fichier binaire nommé matlab.mat
save data    % sauve toutes les variables
             % dans un fichier binaire nommé data.mat
save numdata s1, c1      % sauve les variables s1 et c1 dans numdata.mat
save strdata str         % sauve la variable str dans strdata.mat
save allcos.dat c* -ascii    % sauve c1,c2 dans un format ascii dans
                             %allcos.dat
load              % charge toutes les variables du fichier matlab.mat
load data s1 c1   % charge seulement les variables s1 et c1 de
                  %data.mat
load allcos.dat   % charge toutes les données de allcos.dat
                  %dans la variable allcos
```

Information sur l'espace de travail
Pour obtenir une liste des variables dans l'espace de travail, on utilise les instructions suivantes:
who Affichage des variables dans la commande Windows.
whos Affichage détaillé des variables dans la commande Windows.

Array Editor : Double-cliquer une variable dans le navigateur workspace pour la voir dans l'éditeur de variables « Array Editor ». Il est alors possible de l'´editer sous une représentation visuelle d'un tableau uni ou bidimensionnel, d'une chaîne ou d'un tableau de cellules de chaînes.

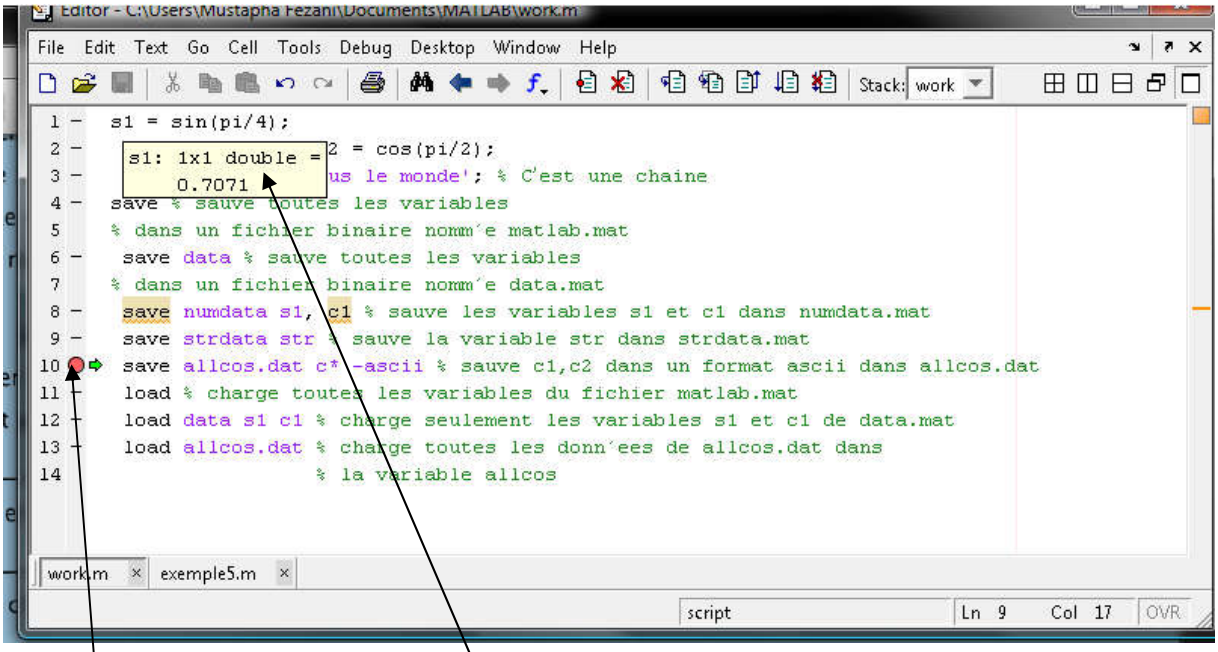
Changer les valeurs d'un élément



Utiliser les onglets pour voir les différentes variables ouvertes

Editeur/Debuggeur

Utiliser l'Editeur/Débuggeur pour créer ou débuggeur un **M-fichier** ou des fonctions. L'Editeur/Débuggeur fournit une interface graphique pour des ' éditions de textes' basiques mais aussi un outil de débogage



Positionne un point d'arrêt pour faire une pause dans L'exécution

Maintenir le curseur sur une variable pour voir apparaitre sa valeur courante

Les variables

Définitions de variables

On définit une variable en donnant son nom et sa valeur numérique ou son expression mathématique:

```
a =1.25;
x = 0:0.5:10;
y = a*x;
z = y.^2;
```

NB : Il n'est pas nécessaire de déclarer les variables ou bien encore leurs dimensions. Quand Matlab rencontre un nouveau nom de variable, il créer automatiquement cette variable et alloue la place mémoire nécessaire à sons stockage. Si la variable existe déjà, Matlab change son contenu et si nécessaire réalloue une nouvelle place pour le stockage.

Exemple :

nombre_Etudiants= 25

Crée une matrice 1 × 1 de nom nombre_etudiants et stocke la valeur 25 dans cet élément simple.

Les noms de variables consistent en une lettre, suivie par un nombre quelconque de lettres, de chiffres ou d'underscores. Matlab n'utilise que les 31 premiers caractères pour identifier les variables. Attention, Matlab fait la distinction entre majuscules et minuscules. Ainsi, A et a ne désigne pas la même variable. Pour voir le contenu d'une variable, il suffit simplement de taper son nom suivi d'un retour chariot.

Expressions mathématiques

On écrit les expressions mathématiques de la façon habituelle:

```
x =4
y=5
z = 5*exp(-0.4*x).*sin(7.5*y)
```

Les nombres

Matlab utilise par convention le point «. » comme notation décimale. On peut définir aussi les nombres à l'aide de la notation scientifique. Elle utilise la lettre e pour spécifier le facteur de puissance de 10.

Examples:

```
3           -99           0.0001
9.6397238   1.60210e-20   6.02252e23
e1i          -3.1415e9j      3e5i
```

Tous les nombre sont stockes de manière interne en utilisant le format long défini par la norme flottante IEEE. La précision est donc de 16 chiffres après la virgule et un intervalle de nombre entre 10-308 et 10+308.

Exercice :
Choisir deux nombres complexes, par exemple -3 + 2i et 5 - 7i. Ajouter, soustraire, multiplier et diviser ces deux nombres.

Typage de variable en Matlab

Les variables entières
Matlab contient huit types entiers quatre signées et quatre sans signe qui sont :
1) int8, int16, int32, int64

Exemple :
```
x=int64(0);% cette variable est codée sur 64 bits.
x=int8(2) ;% cette variable est codée sur 8 bits.
```

2) uint8, uint16, uint32, uint64

Exemple :
```
x=uint32(0);% variable sans signe codée sur 32 bits.
x=uint16(2) ;% variable sans signe codée sur 16 bits.
```

Les Variables de type double
On peut déclarer des variables de type double

Exemple
```
x=int8(2) ;
Y = double(x) ;
```

Les variables globales
On peut déclarer des variables globales

```
global var1 ... varN
```

2. Opérations mathématiques

Nombres et opérations arithmétiques

Nombres : Les nombres réels peuvent être écrits sous différents formats:
5 1.0237 0.5245E-12 12.78e6 0.001234 -235.087
Les nombres complexes peuvent être écrits sous forme cartésienne ou polaire:
Forme cartésienne: 0.5 + i*2.7 -1.2 + j*0.789 2.5 + 9.7i
Forme polaire: 1.25*exp(j*0.246)

Formats d'affichage

La fonction format contrôle le format numérique des valeurs affichées. Cette fonction modifie seulement la manière dont ces valeurs sont affichées, mais pas leur valeur intrinsèque.

Exemples :

```
>> x=[4/3  1.2345e-6]
x =
1.3333       1.2345e-006

>> format short;x
x =
1.3333        0.0000

>> format short e;x
x =
1.3333e+000   1.2345e-006

>> format short g;x
x =
1.3333    1.2345e-006

>> format long;x
x =
1.33333333333333     0.00000123450000

>> format long e;x
x =
1.333333333333333e+000    1.234500000000000e-006

>> format long g;x
x =
1.33333333333333  1.2345e-006

>> format bank;x
x =
1.33      0.00

>> format rat;x
x =
4/3    1/810045

>> format hex;x
x =
```

3ff5555555555555 3eb4b6231abfd271

>> format compact;x; % Supprime les lignes blanches
x =
3ff5555555555555 3eb4b6231abfd271

Pour avoir un format

Utiliser get(0,'format')

Suppression de l'affichage des résultats
Il suffit simplement de rajouter un point-virgule à la fin de l'instruction.

Saisie d'une instruction longue
Si une instruction ne tient pas sur une ligne, il suffit de taper trois points ... suivi d'un retour chariot.

Exemple :
s = 1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;……

Opérations arithmétiques
+ : Addition
- : Soustraction
* : Multiplication
/ : Division à droite
\ : Division à gauche (pour les matrices)
^ : Puissance
' : Transposition et conjugaison complexe
( ) : Spécifie l'ordre d'′evaluation

Fonctions Prédéfini en Matlab

Quelques fonctions Matlab

Matlab fournit un grand nombre de fonctions mathématiques standards (voir dans le help la bibliothèque Mathworks R2010a), incluant par exemple : abs, sqrt, sin et exp.
La racine carrée ou le logarithme d'un nombre négatif ne donne pas d'erreur ; Le résultat approprie en complexe est renvoyé. Matlab met aussi à disposition des fonctions mathématiques plus avancées incluant les fonctions de Bessel ou gamma. La plupart de ces fonctions acceptent les arguments complexes. Pour une liste des fonctions mathématiques élémentaires, taper help elfun

```
>> help elfun
```

-Trigonometric.

```
  sin       - Sine.
  sind      - Sine of argument in degrees.
  sinh      - Hyperbolic sine.
  asin      - Inverse sine.
  asind     - Inverse sine, result in degrees.
  asinh     - Inverse hyperbolic sine.
  cos       - Cosine.
  cosd      - Cosine of argument in degrees.
  cosh      - Hyperbolic cosine.
  acos      - Inverse cosine.
  acosd     - Inverse cosine, result in degrees.
  acosh     - Inverse hyperbolic cosine.
  tan       - Tangent.
  tand      - Tangent of argument in degrees.
  tanh      - Hyperbolic tangent.
  atan      - Inverse tangent.
  atand     - Inverse tangent, result in degrees.
  atan2     - Four quadrant inverse tangent.
  atanh     - Inverse hyperbolic tangent.
  sec       - Secant.
  secd      - Secant of argument in degrees.
  sech      - Hyperbolic secant.
  asec      - Inverse secant.
  asecd     - Inverse secant, result in degrees.
  asech     - Inverse hyperbolic secant.
  csc       - Cosecant.
  cscd      - Cosecant of argument in degrees.
  csch      - Hyperbolic cosecant.
  acsc      - Inverse cosecant.
  acscd     - Inverse cosecant,  result in degrees.
  acsch     - Inverse hyperbolic cosecant.
```

cot - Cotangent.
cotd - Cotangent of argument in degrees.
coth - Hyperbolic cotangent.
acot - Inverse cotangent.
acotd - Inverse cotangent, result in degrees.
acoth - Inverse hyperbolic cotangent.
hypot - Square root of sum of squares.

- Exponential.

exp - Exponential.
expm1 - Compute exp(x)-1 accurately.
log - Natural logarithm.
log1p - Compute log(1+x) accurately.
log10 - Common (base 10) logarithm.
log2 - Base 2 logarithm and dissect floating point number.
pow2 - Base 2 power and scale floating point number.
realpow -Power that will error out on complex result.
reallog - Natural logarithm of real number.
realsqrt - Square root of number greater than or equal to zero. >=0
sqrt - Square root.
nthroot - Real n-th root of real numbers.
nextpow2 - Next higher power of 2.

Exemple :

```
>> y= (1+sqrt(5))/2
y =
          1.6180

>> x = exp(log(realmax))
x =
1.7977e+308

>> tropgrand = pi*x
tropgrand =
             Inf
```

- Complex.

abs - Absolute value.
angle - Phase angle.
complex - Construct complex data from real and imaginary parts.
conj - Complex conjugate.
imag - Complex imaginary part.
real - Complex real part.

unwrap - Unwrap phase angle.
isreal - True for real array.
cplxpair - Sort numbers into complex conjugate pairs.

Exemple :

```
>> a = abs(3+4i)
 a =
              5
```

Rounding and remainder.

fix - Round towards zero.
floor - Round towards minus infinity.
ceil - Round towards plus infinity.
round - Round towards nearest integer.
mod - Modulus (signed remainder after division).
rem - Remainder after division.
sign - Signum.

Pour une liste de fonctions mathématiques plus avancées ou sur les matrices, taper help specfun.

```
>>help specfun
```

Specialized math functions.

airy - Airy functions.
besselj - Bessel function of the first kind.
bessely - Bessel function of the second kind.
besselh - Bessel functions of the third kind
(Hankel function).
besseli - Modified Bessel function of the first kind.
besselk - Modified Bessel function of the second kind.
beta - Beta function.
betainc - Incomplete beta function.
betaln - Logarithm of beta function.
ellipj - Jacobi elliptic functions.
ellipke - Complete elliptic integral.
erf - Error function.
erfc - Complementary error function.
erfcx - Scaled complementary error function.
erfinv - Inverse error function.
expint - Exponential integral function.
gamma - Gamma function.
gammainc - Incomplete gamma function.
gammaln - Logarithm of gamma function.

psi - Psi (polygamma) function.
legendre - Associated Legendre function.
cross - Vector cross product.
dot - Vector dot product.

Exemple :

```
>> rho = (1+sqrt(5))/2
>> z = sqrt(besselk(4/3,rho-i))
z =
0.3730 + 0.3214i
```

Number theoretic functions.

factor - Prime factors.
isprime - True for prime numbers (nombre premier).
primes - Generate list of prime numbers.
gcd - Greatest common divisor.
lcm - Least common multiple.
rat - Rational approximation.
rats - Rational output.
perms - All possible permutations.
nchoosek - All combinations of N elements taken K at a time.
factorial - Factorial function.

Coordinate transforms.

cart2sph - Transform Cartesian to spherical coordinates.
cart2pol - Transform Cartesian to polar coordinates.
pol2cart - Transform polar to Cartesian coordinates.
sph2cart - Transform spherical to Cartesian coordinates.
hsv2rgb - Convert hue-saturation-value colors to red-green-blue.
rgb2hsv - Convert red-green-blue colors to hue-saturation-value.

Autre

```
>>help elmat
```
Elementary matrices and matrix manipulation.

Elementary matrices.

zeros - Zeros array.
ones - Ones array.
eye - Identity matrix.
repmat - Replicate and tile array.
rand - Uniformly distributed random numbers.

randn - Normally distributed random numbers.
linspace - Linearly spaced vector.
logspace - Logarithmically spaced vector.
freqspace - Frequency spacing for frequency response.
meshgrid - X and Y arrays for 3-D plots.
accumarray - Construct an array with accumulation.
: - Regularly spaced vector and index into matrix.

Basic array information.

size - Size of array.
length - Length of vector.
ndims - Number of dimensions.
numel - Number of elements.
disp - Display matrix or text.
isempty - True for empty array.
isequal - True if arrays are numerically equal.
isequalwithequalnans - True if arrays are numerically equal.

Matrix manipulation.

cat - Concatenate arrays.
reshape - Change size.
diag - Diagonal matrices and diagonals of matrix.
blkdiag - Block diagonal concatenation.
tril - Extract lower triangular part.
triu - Extract upper triangular part.
fliplr - Flip matrix in left/right direction.
flipud - Flip matrix in up/down direction.
flipdim - Flip matrix along specified dimension.
rot90 - Rotate matrix 90 degrees.
: - Regularly spaced vector and index into matrix.
find - Find indices of nonzero elements.
end - Last index.
sub2ind - Linear index from multiple subscripts.
ind2sub - Multiple subscripts from linear index.
bsxfun - Binary singleton expansion function.

Multi-dimensional array functions.

ndgrid - Generate arrays for N-D functions and interpolation.
permute - Permute array dimensions.
ipermute - Inverse permute array dimensions.
shiftdim - Shift dimensions.
circshift - Shift array circularly.
squeeze - Remove singleton dimensions.

Array utility functions.

isscalar - True for scalar.
isvector - True for vector.

Special variables and constants.

ans - Most recent answer.
eps - Floating point relative accuracy.
realmax - Largest positive floating point number.
realmin - Smallest positive floating point number.
pi - 3.1415926535897....
i - Imaginary unit.
inf - Infinity.
nan - Not-a-Number.
isnan - True for Not-a-Number.
isinf - True for infinite elements.
isfinite - True for finite elements.
j - Imaginary unit.
why - Succinct answer.

Specialized matrices.

compan - Companion matrix.
gallery - Higham test matrices.
hadamard - Hadamard matrix.
hankel - Hankel matrix.
hilb - Hilbert matrix.
invhilb - Inverse Hilbert matrix.
magic - Magic square.
pascal - Pascal matrix.
rosser - Classic symmetric eigenvalue test problem.
toeplitz - Toeplitz matrix.
vander - Vandermonde matrix.
wilkinson - Wilkinson's eigenvalue test matrix.

Quelques fonctions, comme sqrt et sin sont des fonctions ”built-in”. Ceci signifie qu’elles font parties intégrantes du noyau Matlab et qu’elles sont ainsi très efficaces, mais les règles et les détails de calcul ne sont pas accessibles en lecture. D’autres fonctions, comme gamma et sinh, par contre, sont mises en œuvre à l’aide de M-fichiers. On peut donc voir leur code et même les modifier si l’on veut.
Plusieurs fonctions spéciales donnent les valeurs de constantes usuelles.

Pi	3.14159265...
I	nombre imaginaire, $\sqrt{-1}$
J	comme i
Eps	précision flottante relative, 2^{-52}
Realmin	plus petit nombre flottant, 2^{-1022}
Realmax	plus grand nombre flottant, $(2-\varepsilon)2^{1023}$
Inf	Infini
NaN	Not-a-number

L'infini est génère en divisant une valeur quelconque par 0. Not-a-number est génère en tentant d'évaluer des expressions comme 0/0 ou Inf-Inf.
Les noms de fonctions ne sont pas réservés. Il est possible de les écraser avec une nouvelle variable. Attention, ceci est une source d'erreur. Si l'on utilise i comme indice d'une boucle par exemple, il n'aura plus le sens d'un nombre complexe. Pour le ramener à son état initial, il faut utiliser clear i.

Résumé d'opérations mathématiques et leurs correspondances en Matlab

Notation mathématique	Commande Matlab
A+B	A+B
A-B	A-B
AB	A*B
A/B	A/B ou B\A
X^b	X^b
$\sqrt{x}$	sqrt(x) ou x^0.5
$\|x\|$	abs(x)
π	Pi
4.10^3	4e3 ou 4*10^3
3-4i	3-4*i ou 3-4*j
e, e^x	exp(1) , exp(x)
ln(x) , log(x)	log(x), log10(x)
sin x, artan x,…	sin(x), atan(x)…

Programmation avec Matlab

Structures de contrôles

Operateurs logiques et relationnels

Pour obtenir de l'aide sur les operateurs relationnels, taper help relop. Les operateurs <, <=, >, >=, == et ~= sont utilises pour comparer. Les operateurs logiques &, | et ~ permettent les combinaisons logiques, les négations ou les relations. En plus existent trois fonctions supplémentaires : xor, any et all.

Exemple :

Commande	Résultat
a = (b > c)	a est 1 si b est plus grand que c. Identique avec <, > = et <=
a = (b == c)	a est 1 si b est égal à c
a = (b ~= c)	a est 1 si b n'est pas égal à c
a = ~b	complément logique : a est 1 si b est 0
a = (b & c)	ET logique : a est 1 si b est vrai et c est vrai
a = (b | c)	OU logique : a est 1 si b est vrai ou c est vrai

Les primitives conditionnelles

L'instruction if

La commande if évalue une expression logique et exécute un groupe d'instructions quand l'expression est vraie. Les mots clés optionnels elseif et else donne la possibilité d'exécuter d'autres groupes d'instructions si l'expression est fausse. Il faut rajouter à la fin le mot clés end pour indiquer la fin du dernier groupe d'instructions.

Exemple :

La fonction Matlab qui crée une matrice magique utilise trois algorithmes différents suivant les cas : quand n est impaire, quand n est paire mais pas divisible par 4 ou quand n est divisible par 4.

```
if rem(n,2) ~= 0
M = odd_magic(n)
elseif rem(n,4) ~= 0
M = single_even_magic(n)
else
M = double_even_magic(n)
end
```

Remarque : il est important de comprendre le fonctionnement de if En effet, le test if A == B, ... est l'égal en Matlab. Si A et B sont des scalaires, ceci est le test d'égalite entre A et B. Par contre, si A et B sont des matrices, A==B ne teste pas si A et B sont égales, mais ou elles le sont ! Le résultat est une autre matrice de 0 et de 1. En fait, si A et B ne sont pas de même taille, alors A==B est une erreur. La façon propre de teste l'égalité de deux matrices est if isequal(A,B), ...

Voici quelques fonctions pour comparer les matrices.
isequal isempty isfinite(A) all any

Les instructions switch et case

La commande switch exécute un groupe d'instructions suivant la valeur d'une variable a ou d'une expression.
Les mots clés case et otherwise délimite le groupe. Seulement le premier cas vrai est exécute. Il doit obligatoirement y avoir un end à la fin. La partie logique pour l'algorithme de la matrice magique pourrait être

```
switch (rem(n,4)==0) + (rem(n,2)==0)
case 0
  M = odd_magic(n)
case 1
  M = single_even_magic(n)
case 2
  M = double_even_magic(n)
otherwise
  error('This is impossible')
end
```

Exemple

```
result = 52; switch(result) case 52 disp('result is 52') case {52, 78} disp('result is 52 or 78')
end
```

Les primitives de boucles

L'instruction for

La commande de boucle for répète un groupe d'instructions un nombre prédétermine de fois. Un end délimite la fin.

Syntaxe

```
for  i=valInf : vaSup
     Instructions
end
```

Exemple

```
for n = 3:32
r(n) = rank(magic(n));
end
```

L'instruction while

La boucle while répète un groupe d'instructions un nombre de fois indéterminé sous le contrôle d'une expression logique. Un end termine l'instruction.

Syntaxe

```
Initialiser cond
While cond
  Instructions
end
```

Exemple :

Voici un programme complet qui illustre while, if et else. C'est l'algorithme de la bissection pour la recherche de zéros d'une fonction.

```
a = 0; fa = -inf;
b = 3; fb = inf;
while b-a > eps*b
x = (a+b)/2;
fx = x^3-2*x-5;
if sign(fx) == sign(fa)
a = x; fa = fx;
else
b = x; fb = fx;
end
end
```

Le résultat est une racine du polynôme $x^3 - 2x - 5$.

x = 2.09455148154233

L'instruction continue

L'instruction continue stoppe l'itération en cours dans une boucle dans laquelle elle apparait et passe à l'itération suivante. Dans des boules imbriques, elle stoppe l'itération en cours et donne le contrôle à l'itération externe.

Exemple :

L'exemple suivant montre l'utilisation de continue dans le décompte du nombre de lignes de codes du fichier « fichier.m », sans compter les lignes blanches et les commentaires.

```
fid = fopen('fichier.m','r');
count = 0;
while ~feof(fid)
line = fgetl(fid);
if isempty(line) | strncmp(line,'%',1)
continue
end
count = count + 1;
end
disp(sprintf('%d lines',count));
```

L'instruction break

La commande break permet de quitter une boucle avant la fin de celle ci. Dans des boucles imbriquées, break ne fait que sortir de la boucle interne.

Exemple :

Voici le même exemple que pour la commande while

```
a = 0; fa = -inf;
b = 3; fb = inf;
while b-a > eps*b
x = (a+b)/2;
fx = x^3-2*x-5;
if  fx == 0
break
elseif  sign(fx) == sign(fa)
a = x; fa = fx;
else
b = x; fb = fx;
end
end
```

Les autres structures de donnés
Vecteurs et Matrices

Vecteurs

L'operateur :

L'operateur deux-points : est l'un des plus importants en Matlab. Il est utilisé sous différentes formes.
L'expression « 1:10 » est une vectrice ligne contenant les entiers entre 1 et 10

```
>> 1:10
ans =
1 2 3 4 5 6 7 8 9 10
```

Pour obtenir un espacement autre que un, il suffit de spécifier l'incrément.

Exemple :

```
>> 100:-7:50
 ans =
100   93   86   79   72   65   58   51
```

et

```
>> 0:pi/4:pi
 ans =
0    0.7854   1.5708   2.3562   3.1416
```

La commande linspace

Ou en utilisant une fonction qui génère un vecteur:

```
>> x=linspace(1,10,6) %: créant un vecteur avec 6 éléments
x =
1.0000  2.8000  4.6000  6.4000  8.2000  10.0000

>> X=linspace(1, 10) %:génère 100 éléments ;
```

La commande logspace

```
>> y=logspace(1,3,7)
y =
1.0e+003 *
0.0100  0.0215  0.0464  0.1000  0.2154  0.4642  1.0000
>> y=logspace(1,3) ; génère 50 élément logarithmique entre 1O^1…… 10^3
```

Remarque:

Lors qu'on ajoute un «;» à la fin d'une instruction, elle est exécutée mais le résultat n'est pas affiché:

```
>> a=[1 2 3 4 5];
>> b=-2.5;
```

```
>> c=b*a;
>>
```

Lors qu'il n'y a pas de «;» à la fin d'une instruction, elle est exécutée et le résultat est affiché:

```
>> a=[1 2 3 4 5]
a =
1 2 3 4 5
>> b = -2.5
b =
-2.5000
>> c=b*a
c =
-2.5000   -5.0000   -7.5000   -10.0000  -12.5000
>>
```

On peut définir un vecteur x en donnant la liste de ses éléments:

```
>> x=[0.5  1.2  -3.75   5.82   -0.735]
x =
0.5000   1.2000   -3.7500   5.8200   -0.7350
```

ou en donnant la suite qui forme le vecteur:

```
>> x=2:0.6:5
x =
2.0000  2.6000  3.2000 3.8000  4.4000  5.0000
```

Matrices

Manipulation des Matrices

Dans Matlab, une matrice est un tableau rectangulaire de nombres. Des significations particulières sont liées aux matrices 1 × 1, qui sont des scalaires, et aux matrices avec seulement une ligne ou une colonne, qui sont vecteurs. Ainsi, toutes variables, à quelques exceptions prés, sont des matrices. Matlab possède d'autres méthodes de représentation de l'information, à la fois numérique et non numérique, mais pour commencer, il est préférable de penser que tout est une matrice. Les opérations en Matlab sont définies pour être les plus naturelles possibles.

Avec Matlab, il est possible de manipuler et de travailler simplement avec des matrices complètes, sans être oblige de coder les produits matrices-matrices ou matrices-vecteurs par exemple comme c'est le cas en Fortran ou en C.

Saisie de matrices :

La meilleure manière de débuter avec Matlab est d'apprendre comment manipuler les matrices. Il est possible de saisir des matrices de différentes manières

- entrer une liste explicite d'arguments
- charger une matrice depuis un fichier externe
- générer des matrices avec des fonctions Matlab
- créer une matrice avec des M-fichiers

Nous commençons par saisir la matrice de Durer comme une liste de ses éléments. Il faut suivre les règles suivantes

– séparer les éléments d'une même ligne par des espaces ou des virgules
– utiliser le point virgule pour indiquer la fin d'une ligne
– encadrer toute la liste des éléments par des crochets [et].
En suivant les principes précédents, la saisie de la matrice de Durer se fait par :
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

Matlab affiche alors la matrice
A =

```
16   3   2  13
 5  10  11   8
 9   6   7  12
 4  15  14   1
```

Une fois que la matrice a été saisie, elle est automatiquement stockée dans l'espace de travail ("workspace") Matlab sous forme de la variable A. On peut donc maintenant se référer a cette matrice par A. Maintenant, pourquoi cette matrice est magique ?
Somme, transposée et diagonale
Si l'on prend la somme des lignes ou la somme des colonnes, ou bien des deux diagonales principales, on obtient toujours le même nombre. Vérifions cela avec Matlab.

>>sum(A)

Matlab répond

ans =
34 34 34 34
Quand on ne spécifie par de variable pour le résultat, Matlab utilise la variable par défaut ans. On vient de calculer un vecteur ligne qui contient la somme des colonnes de A. Pour faire la même opération sur les lignes, il suffit de transposer la matrice.
>> A' Produit
ans =

```
16 5 9 4
3 10 6 15
2 11 7 14
13 8 12 1
```

et

>>sum(A')'
Produit une vectrice colonne contenant la somme des lignes
ans =

```
34
34
34
34
```

La somme des éléments de la diagonale principale est obtenue avec les commandes sum et diag.

```
>> diag(A)
ans =
                16
                10
                7
                1
```

et

```
>> sum(diag(A))
ans =
                34
```

L'autre diagonale principale, aussi appelée anti-diagonale, n'est pas très importante mathématiquement et donc Matlab n'a pas de fonctions prédéfinie pour l'obtenir. On va utiliser la fonction « fliplr » qui effectue une symétrie de la matrice de droite à gauche.

```
>> sum(diag(fliplr(A)))
ans =
                34
```

Les indices

L'élément en ligne i et en colonne j de A est désigne par A(i,j). Par exemple, A(4,2) est le nombre en quatrièmes ligne et deuxième colonne. Pour la matrice magique précédente, A(4,2) est 15. Il est donc possible de calculer la somme des éléments dans la quatrième colonne par la commande

```
>> A(1,4) + A(2,4) + A(3,4) + A(4,4)
ans =
                34
```

Mais cette méthode est moins élégante.

Il est aussi possible de designer des éléments d'une matrice À l'aide d'un seul indice À (k). Dans le cas d'un vecteur, cette manière de noter n'a rien de choquant. Dans le cas d'une matrice, elle peut parâtre plus étrange.

Dans ce cas, la matrice est vu comme un long vecteur colonne forme à partir des autres colonnes de la matrice originale. Ainsi, A(8) est une autre manière de référencer la valeur 15 stockée dans A(4,2).

Si l'on tente d'utiliser un élément en dehors de la matrice, on obtient une erreur

```
>> t = A(4,5)
Erreur : ??? Index exceeds matrix dimensions.
```

D'un autre coté, si l'on stocke un élément en dehors de la matrice, sa taille est automatiquement augmentée pour accommoder la nouvelle matrice

```
>> X = A;
X(4,5) = 17
X =
                16  3  2   13  0
                5  10  11  8   0
```

9 6 7 12 0
4 15 14 1 17

t(10) élément no. 10 du vecteur t
A(2,9) élément se trouvant à ligne 2, colonne 9 de la matrice A
B(:,7) la colonne 7 de la matrice B
C(3,:) la ligne 3 de la matrice C

Extension de Matrice

A=[A,A]
A =

0.5000	2.7000	3.9000	0.5000	2.7000	3.9000
4.5000	0.8500	-1.2300	4.5000	0.8500	-1.2300
-5.1200	2.4700	9.0300	-5.1200	2.4700	9.0300

A=[A ;A]
A =

0.5000	2.7000	3.9000	0.5000	2.7000	3.9000
4.5000	0.8500	-1.2300	4.5000	0.8500	-1.2300
-5.1200	2.4700	9.0300	-5.1200	2.4700	9.0300
0.5000	2.7000	3.9000	0.5000	2.7000	3.9000
4.5000	0.8500	-1.2300	4.5000	0.8500	-1.2300
-5.1200	2.4700	9.0300	-5.1200	2.4700	9.0300

Les indices peuvent aussi utiliser l'operateur :. Il devient alors très simples de designer des portions de matrices.

Exemple :

>>A(1:k,j) désigne les k premiers éléments de la j ième colonne de A. Ainsi, sum(A(1:4,4)) calcule la somme de la 4ieme colonne.
Mais, il y a encore mieux. Les : par eux mêmes désignent tous les éléments d'une ligne ou d'une colonne d'une matrice. Le mot clé end désigne la dernière ligne ou dernière colonne.

Exemple :

>> sum(A(:,end)) calcule la somme des éléments dans la dernière colonne de A.
Si les entiers de 1 à 16 sont ranges dans quatre groupes de sommes égales, alors, cette somme doit être

>> sum(1:16)/4
ans =
34

La fonction magic

Matlab a en fait une fonction pour créer des matrices magiques de taille quelconque

>> B=magic(4)

B =

16 2 3 13
5 11 10 8
9 7 6 12
4 14 15 1

Cette matrice est à peu prés la même que celle de Durer. La seule différence est que les deux colonnes centrales ont été permutées. Pour transformer B en A, il suffit donc de permuter ces deux colonnes.

>> A = B(:,[1 3 2 4])
A =

16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1

Pour chaque ligne de B, on réordonne les éléments dans l'ordre 1,3,2,4.
Certaines fonctions prédéfinis permettent de construire des matrices spéciales souvent utilisées :

ones : construit des matrices remplies de 1
eye : construit une matrice identité
zeros : construit une matrice nulle
rand : construit une matrice dont les éléments suivent une loi uniforme sur [0,1]
randn : construit une matrice dont les éléments suivent une loi normale réduite
diag : construit une matrice diagonale ou extrait une diagonale
triu : extrait la partie triangulaire supérieure
tril : extrait la partie triangulaire inferieure

Matrice unitaire:

>> B=eye(4) : crée une matrice identité 4 lignes et 4 colonnes
B =
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
>> b=eye(3,4) : matrice identité 3 lignes et quatre colonnes
b =

1	0	0	0
0	1	0	0
0	0	1	0

>> b=eye(size(A)) : crée une matrice identité de même dimension que A
>> A=ones (3,4)
A =

1	1	1	1
1	1	1	1

```
    1     1     1     1
>> A=ones (3,3)
A =
    1     1     1
    1     1     1
    1     1     1
>> b=eye(size(A))
b =
    1     0     0
    0     1     0
    0     0     1
>> N = fix(10*rand(1,10))
N =
7 4 0 8 4 6 7 9 7 1
```

La concaténation

est un autre moyen de création de matrices. Elle permet de joindre plusieurs petites matrices pour en former une plus importante. Les crochets [et] sont en fait les operateurs de concaténation.

```
>> B = [A A+32; A+48 A+16]
B =
16     3     2    13    48    35    34    45
 5    10    11     8    37    42    43    40
 9     6     7    12    41    38    39    44
 4    15    14     1    36    47    46    33
64    51    50    61    32    19    18    29
53    58    59    56    21    26    27    24
57    54    55    60    25    22    23    28
52    63    62    49    20    31    30    17
```

Il est aussi possible d'effacer des lignes et des colonnes trs simplement. Commençons par copier la matrice A dans X par la commande X=A; Nous notons pour la première fois l'apparition du point virgule. Il sert à éviter l'affichage du résultat par Matlab. Alors, pour effacer la deuxième colonne de X, on utilise

```
>> X=A;
>> X(:,2)=[]
X =
    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

Si on efface simplement un élément, le résultat n'est alors plus une matrice. Ainsi, l'expression X(1,2)=[] provoque une erreur.

On peut par contre effacer des sections d'une matrice

```
>> B=A;
>> B(1:2,2:3)=0
```

B =

16	0	0	13
5	0	0	8
9	7	6	12
4	14	15	1

Opérations matricielles

Les opérations matricielles exécutées par MATLAB sont illustrées dans le tableau suivant:
B = A' La matrice B est égale à la matrice A transposée
E = inv(A) La matrice E est égale à la matrice A inversée
C = A + B Addition
D = A - B Soustraction
Z = X*Y Multiplication
X = A\B Équivalent à inv(A)*B
X = B/A Équivalent à B*inv(A)

Opération « Elément par élément »

Les opérations «élément par élément» des vecteurs et des matrices sont effectuées en ajoutant un point (.) avant les opérateurs * / \ ^ '

Exemple :

```
>> A=[1 2 3 4 5];
>> B=[6 7 8 9 10];
>> C=A.*B
C =
6 14 24 36 50
>> D=A./B
D =
0.1667    0.2857    0.3750    0.4444    0.5000
```

Les tableaux multidimensionnels

Les tableaux multidimensionnels sont des tableaux qui ont plus de deux indices. Ils peuvent être créés à l'aide des fonctions zeros, ones, rand et randn avec plus de deux arguments. Par exemple,

```
A=[1:3;4:6]
A =
    1    2    3
    4    5    6

>> A(:,:,2)=zeros(2,3)
A(:,:,1) =
    1    2    3
    4    5    6
A(:,:,2) =
```

0 0 0
0 0 0

R = randn(3,4,5);

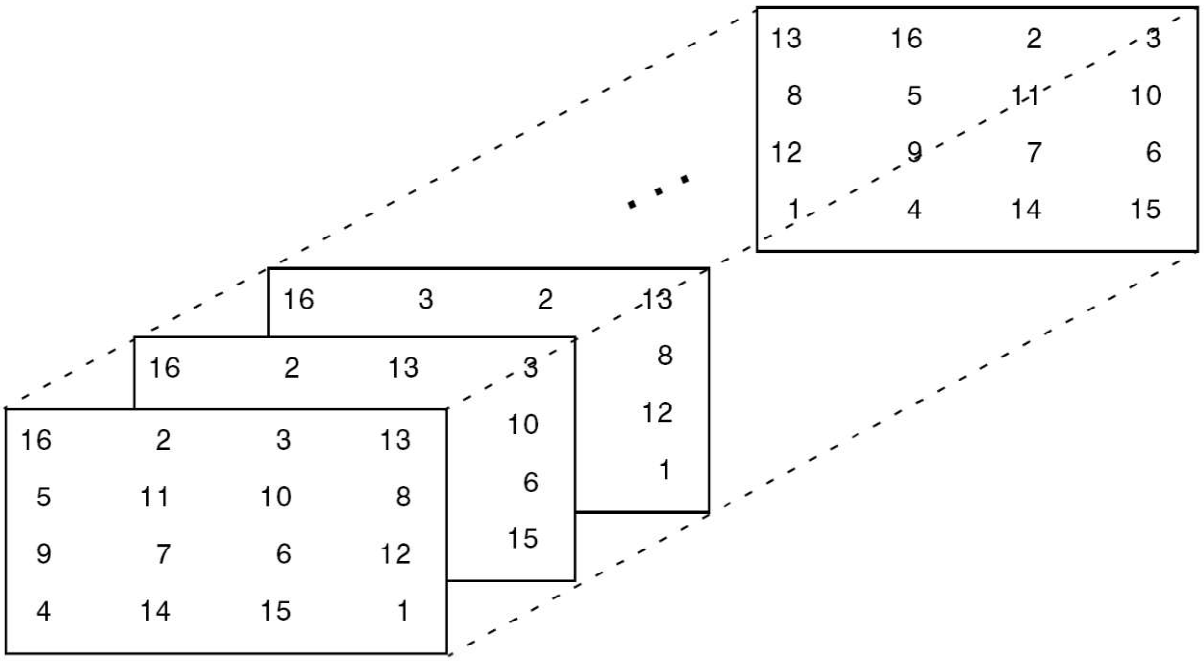
Crée un tableau 3 × 4 × 5. Un tableau multidimensionnel peut représenter un tableau 3D de données physiques, mais aussi une suite de matrices. Dans la suite, le (i, j) ème élément d'une k ème matrice sera noté A(i, j, k).

Les matrices magiques d'ordre 4 peuvent être modifiées en échangeant deux colonnes. La commande p=perm(1:4) génère les 24 permutations de 1:4. La $k^{ième}$ permutation est le vecteur ligne p(k,:). Alors,

```
A = magic(4);
M = zeros(4,4,24);
p=perms(1:4);
for k = 1:24
M(:,:,k) = A(:,p(k,:))
end
```

Stocke la suite des 24 matrices magiques dans un tableau 3D, M. La taille de M est

```
>> size(M)
ans =
    4    4   24
```



Alors, il est évident que la troisième matrice dans la suite est

```
>> M(:,:,3)
ans =
   13    2    3   16
    8   11   10    5
   12    7    6    9
    1   14   15    4
```

Concaténation de matrice

La concaténation est un autre moyen de création de matrices. Elle permet de joindre plusieurs petites matrices pour en former une plus importante. Les crochets [et] sont en fait les operateurs de concaténation.

```
>> B = [A A+32; A+48 A+16]
B =
16    3    2   13   48   35   34   45
 5   10   11    8   37   42   43   40
 9    6    7   12   41   38   39   44
 4   15   14    1   36   47   46   33
64   51   50   61   32   19   18   29
53   58   59   56   21   26   27   24
57   54   55   60   25   22   23   28
52   63   62   49   20   31   30   17
```

Il est aussi possible d'effacer des lignes et des colonnes tr`es simplement. Commençons par copier la matrice A dans X par la commande X=A; Nous notons pour la première fois l'apparition du point virgule. Il sert à éviter l'affichage du résultat par Matlab. Alors, pour effacer la deuxième colonne de X, on utilise

```
>> X=A;
>> X(:,2)=[]
X =
   16    2   13
    5   11    8
    9    7   12
    4   14    1
```

Si on efface simplement un élément, le résultat n'est alors plus une matrice. Ainsi, l'expression X(1,2)=[] provoque une erreur.
On peut par contre effacer des sections d'une matrice

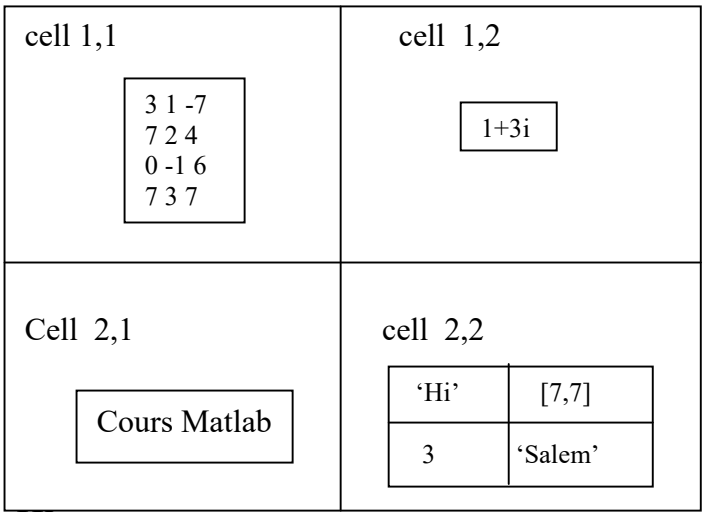
```
>> B=A;
>> B(1:2,2:3)=0
```

Cellule et structure

Dans les versions les plus récentes de MATLAB, il ya deux groupes de données qui sont plus élaborées que les tableaux scalaires et des tableaux de chaînes de caractères: le premier est appelé une cellule et la seconde une structure.

Dans une matrice de cellules, les éléments peuvent être de toute nature, la valeur numérique, chaîne de caractères, tableau, etc Type:

Cellule
Les tableaux de cellules



Les tableaux de cellules sont des tableaux multidimensionnels dont les éléments sont des copies d'autres tableaux.
Un tableau de cellules peut être crée avec la commande cell. Plus souvent, les tableaux de cellules sont crées tout simplement avec des accolades. Elles sont aussi utilisées pour lire une cellule.

Exemple1 de cellule

```
omar={'MATLAB',[6.5;2.3],2010}

omar =

    'MATLAB'    [2x1 double]    [2010]
omar={'MATLAB',[6.5;2.3],2010}

omar =

    'MATLAB'    [2x1 double]    [2010]

>> omar(1)

ans =

    'MATLAB'

>> omar(2)

ans =

    [2x1 double]
```

```
>> omar{2}

ans =

   6.5000
   2.3000

>> omar{3}

ans =

        2010
```

Omar est composé de trois éléments: le premier est une chaîne de caractères, le second est un vecteur colonne, et le troisième est un scalaire. Cet exemple montre la différence de syntaxe entre un tableau et une cellule, une accolade gauche {() et une accolade fermante ()} sont utilisés au lieu d'un crochet gauche ([) et d'un crochet droit (]). Quant au contenu, omar (2) se réfère au vecteur [6.5000, 2.3], omar (2) pour le contenu de ce vecteur, et omar {2} (1) de la valeur numérique 6.5.

Exemple2

```
Boutagou={'Département mathématique', [1 2 3 4 5],[1;2], 'Omar'}

Boutagou =

   [1x24 char]    [1x5 double]    [2x1 double]    'Omar'

>> Boutagou{2}

ans =
    1     2     3     4     5

>> Boutagou{2}(1)
ans =
    1

>> Boutagou{2}(2)
ans =
    2

>> Boutagou{2}(4)

ans =
    4
```

```
>> Boutagou(4)
ans =
   'Omar'
```

Structure

La structure est définie par l'instruction "struct ". L'exemple suivant définit une structure, appelée varstruc, composé de trois domaines: la language, Version, et année. L'instruction assigne la chaîne de caractères MATLAB au premier champ, la chaîne de caractères 6.5 au deuxième champ, et la valeur numérique 2010 pour le troisième champ:

Exemple

```
varstruc=struct('Language','MATLAB','Version','6.5','annee',2010);

varstruc.annee
ans =
      2010

varstruc.Version
ans =
6.5

varstruc.Language
ans =
MATLAB
```

La deuxième instruction affiche le contenu des varstruc.annee, qui est 2010. La dimension A 1 x 1 de la structure est organisée de la même manière que la dimension n x 1 du tableau de cellules, où n est le nombre de champs de la structure. Les cellules peuvent donc être comparées à des structures avec des champs sans nom.

L'exemple suivant définit une structure nommée langstruc, composée de deux enregistrements.

Chaque enregistrement contient les trois domaines language, Version, et de l'année qui ont été respectivement affectés, les séquences des deux chaînes de caractères MATLAB et C, des deux valeurs 6.5 et 15.1, et des deux valeurs de 2009 et 2010:

```
>>varstruc=struct('Langage',{{'MATLAB','C'}},...
'Version',[6.5;15.1],'anne',[2002;2003]);
```

Ou bien

```
>>varstruc=struct('Langage',{{'MATLAB','C'}},'Version',[6.5;15.1],'anne',[2002;2003]);


>>varstruc

varstruc =
```

```
    Langage: {'MATLAB'  'C'}
    Version: [2x1 double]
      anne: [2x1 double]

>> varstruc.Langage{1}

ans =

MATLAB

>> varstruc.Langage(1)

ans =

    'MATLAB'
```

Ces objets peuvent être manipulés à l'aide de certaines fonctions: isstruct, les noms des champs, setField, rmfield, cellfun, celldisp, num2cell, cell2mat, cell2struct, struct2cell.

Un exemple d'une conversion est la suivante:

```
>> clear all
>>clc
>> omarcellule={'MATLAB',[6.5;2.3],2010}

omarcellule =

    'MATLAB'    [2x1 double]    [2010]

>> chps={'Langage','Version','annee'};

>> cell2struct(omarcellule,chps,2)

ans =
    Langage: 'MATLAB'
    Version: [2x1 double]
      annee: 2010
```

Les 2 qui fait partie de l'instruction cell2struct (omarcellule, chps, 2) indique la dimension de omarcellule qui doit être pris en compte pour définir le nombre de domaines. Voici, par exemple, size (omarcellule, 2) signifie que le nombre de champs est de 3.

```
>> size (omarcellule, 2)
ans =
     3
```

Fonctions mathématiques

Les fonctions mathématiques de base sont données dans le tableau suivant:
Abs :valeur absolue
module (nb. complexe)
angle :argument (nb. complexe)
sqrt :racine carrée
real :partie réelle
imag :partie imaginaire
conj :conjuguée (nb. complexe)
round :arrondir
fix :arrondir (vers zéro)
floor :arrondir (vers -□)
ceil :arrondir (vers □)
sign :signe
rem :reste

Exemple : exponentielle
log : logarithme base e
log10 : logarithme base 10

Les fonctions trigonométriques sont données dans le tableau suivant:

sin	cos	asin	acos	atan	tan	atan2
sinh	cosh	tanh	asinh	acosh	atanh	

Exemple 3:

```
>> x=-2+5i %variable complexe
x =
-2.0000 + 5.0000i
>> a=real(x) % partie reel de la variable complexe
a =
-2
>> b=imag(x) % partie imaginaire  de la variable complexe
b =
5
>> X=abs(x)
X =
5.3852
>> alfa=angle(x)
alfa =
1.9513
```

Exemple 4:

```
>> w=50;
>> t=0.5e-3;
>> y=25*exp(-4*t)*cos(w*t)
y =
24.9423
```

Création de fonctions

L'usager peut créer des fonctions particulières pour ses applications. Voir «Programmation avec MATLAB».

Les fonctions sont de M-fichiers qui peuvent accepter des arguments d'entrées et des arguments de retour.
Le nom de ces M-fichiers et de la fonction doivent être rigoureusement identique. Les fonctions opèrent sur des variables qui leur sont locales. Elles sont dans un espace de travail propre complètement séparé du workspace principal.

Déclaration d'une fonction

Soit $s_1,s_2,s_3,\ldots,s_m$ les variables de sorties et $e_1,e_2,e_3,\ldots,e_n$ les variables d'entrées, on déclare la syntaxe de la fonction comme suit :

```
function  [s1,s2,s3,…,sm]=nom_fonction(e1,e2,e3,…,en)
   Instructions
end
```

Exercice :

Résoudre du premier et second dégrée avec deux fonctions d'en-têtes respectives :

function x=equ1(a,b) et function equ2(a,b,c).

Solution :

1) ax+b=0 implique que x=-b/a avec a ≠0

```
function x=equ1(a,b)
if a==0
  if b~=0
     x=[];
  else
     x=NaN;%not a number(ce n'est pas un nombre)
  end;
else
   x=-b/a;
end;
```

2)$ax^2+bx+c=0$

Si a=0 alors appel de la fonction equ1(b,c)
Sinon
Delta=b^2-4*a*c ;
Discuter les résultats

```
function x= equ2(a,b,c)
if nargin==2
   x=equ1(a,b);
elseif a==0;
    x=equ1(b,c);
else
   deltat=b*b-4*a*c;
   if deltat>=0
      if b>0
         x=(-b-sqrt(deltat))/(2*a);
      else
         x=(-b+sqrt(deltat))/(2*a);
      end;
      if x==0
         x=[x 0];
      else
         x=[x (c/a)/x];
      end;
   end;
end;
```

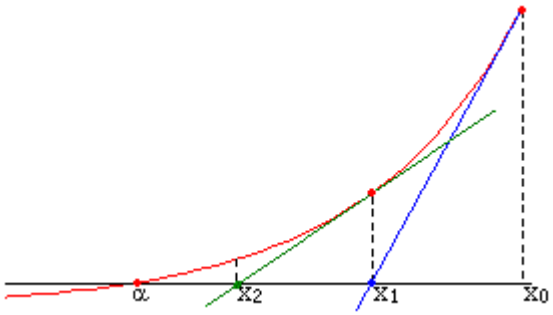
Exercice :
On donne l'équation $ax^3+bx^2+cx+d=0$

1. Trouvez une racine réelle x1par la méthode de Newton à l'aide d'une fonction x1=Newton (a,b,c,d)
2. écrivez la fonction résolvant l'équation du troisième degré.

Solution
Formellement, on part d'un point x_0 appartenant à l'ensemble de définition de la fonction et on construit par récurrence la suite :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

où f' désigne la dérivée de la fonction f. Le point x_{n+1} est bien la solution de l'équation affine $f(x_n) + f'(x_n)(x - x_n) = 0$.



Exemple

Pour illustrer la méthode, recherchons le nombre positif x vérifiant $\cos(x) = x^3$. Reformulons la question pour introduire une fonction devant s'annuler : on recherche le zéro positif (la racine) de $f(x) = \cos(x) - x^3$. La dérivation donne $f'(x) = -\sin(x) - 3x^2$.

Comme $\cos(x) \leq 1$ pour tout x et $x^3 > 1$ pour $x > 1$, nous savons que notre zéro se situe entre 0 et 1. Nous essayons une valeur de départ de $x_0 = 0{,}5$.

$$
\begin{array}{rcccl}
x_1 & = & x_0 - \frac{f(x_0)}{f'(x_0)} & = & 0,5 - \frac{\cos(0,5)-0,5^3}{-\sin(0,5)-3\times 0,5^2} \simeq 1,112\,141\,637\,1 \\
x_2 & = & x_1 - \frac{f(x_1)}{f'(x_1)} & & \vdots \simeq 0,909\,672\,693\,736 \\
x_3 & & \vdots & & \vdots \simeq 0,866\,263\,818\,209 \\
x_4 & & \vdots & & \vdots \simeq 0,865\,477\,135\,298 \\
x_5 & & \vdots & & \vdots \simeq 0,865\,474\,033\,111 \\
x_6 & & \vdots & & \vdots \simeq 0,865\,474\,033\,101 \\
x_7 & & \vdots & & \vdots \simeq 0,865\,474\,033\,102
\end{array}
$$

Les 7 premiers chiffres de cette valeur coïncident avec les 7 premiers chiffres du vrai zéro.

```
function x1=newton1(a,b,c,d)
%cherchant une racine en partant d'un point x0=0,à la précision eps
epsi=1.0E-5;
x1=0;
while 1
   fprime=(3*a*x1+2*b)*x1+c;
   if fprime ==0
      x1=-b/a;
   else
      dx=-(((a*x1+b)*x1+c)*x1+d)/fprime;
      x1=x1+dx;
      if abs(dx)<epsi
         break;
```

```
      end;
   end;
end; %end while

function x=equ3(a,b,c,d)
if a==0
   x=equ2(b,c,d);
else
   x=newton1(a,b,c,d);
   beta=b+a*x;
   gamma=c+beta*x;
   x=[x equ2(a, beta, gamma)];
end;
```

Intégration numérique

Méthodes de base.

Matlab fourni deux méthodes d'intégration numériques de type adaptatif et une méthode pour les intégrales doubles.

Intégrale simple :

a) **quad :** méthode numérique d'évaluation de l'intégrale simple en utilisant la méthode récursif de simpson adaptative avec une erreur de 1^{E}-6 entre la borne a et b.

$$\int_a^b f(x)dx$$

Syntaxe :

```
q = quad(fun,a,b)
```

Exercice :
Ecrivez une fonction d'intégration suivant la méthode de simpson adaptative:

$$\int_0^2 \frac{1}{x^3 - 2x - 5} dx$$

Programme associé

```
function y = myfun(x)
  y = 1./(x.^3-2*x-5);
```

Utilisation de quad en ligne de commande

```
>> Q = quad(@myfun,0,2)
    Q =
         -0.4605
```

Autre façon de calculer l'intégrale on affectant l'intégrale comme un anonyme fonction handle F ou en mode inline.

En utilisant le handle F

Fichier test.m

```
F = @(x)1./(x.^3-2*x-5);
Q = quad(F,0,2)
Q =
         -0.4605
```

En utilisant inline

Fichier test1.m

```
f1=inline ('1./(x.^3-2*x-5)');
quad(f1,0,2)
```

```
Q =
    -0.4605
```

b) **quadl :** méthode numérique d'évaluation de l'intégrale simple en utilisant la méthode récursif de Lobatto adaptative avec une erreur de 1^{E}-6 entre la borne a et b.

$$\int_a^b f(x)dx$$

Exercice : En utilisant la fonction quadl calculer l'intégrale

$$\int_0^2 x^4 \log(x+\sqrt{x^2+1})dx$$

Programme associé

```
function z = fun1(x)
  z = x.^4.*(log(x+sqrt (x.*x+1)));
```

Utilisation de quad en ligne de commande

```
>> Q = quadl(@fun1,0,2)
    Q =
     8.1534
```

En utilisant le mode inline

Fichier test2.m

```
f1=inline ('x.^4.*(log(x+sqrt (x.*x+1)))');
quadl(f1,0,2)
```

résultat :

```
        Q =

    8.1534
```

Intégrale double :

En matlab l'intégrale double $\int_a^b \int_c^d f(x,y)dxdy$,utilise la fonction dblquad.

dblquad : méthode numérique qui appelle la fonction quad pour évaluer l'intégrale double fun(x,y) sur le rectangle délimité par xmin <=x <= xmax, ymin <= y <= ymax. Fun est fonction handle(la fonction handle est une valeur et type de donnée matlab qui fourni un moyen indirect pour appeler une fonction).

Syntaxe :

```
q = dblquad(fun,xmin,xmax,ymin,ymax)
```

Exercice : En utilisant la fonction dbquad calculer l'intégrale double

$$\int_0^5 \int_0^5 e^{-x^2-y^2} dxdy$$

Programme associé

```
function y=doubleIntegrale(x,y)
    y=exp(-x.*x-y.*y);
>> dblquad(@doubleIntegrale,0,5,0,5)
ans =
    0.7854
```

1 Approximation numérique d'une intégrale.
On considère une fonction f continue sur un intervalle [a, b] de R et à valeurs dans R^+. On se propose d'évaluer numériquement l'intégrale I définie par

$$I = \int_a^b f(x)dx .$$

A titre d'exemple, on évaluera l'intégrale entre 0 et 2π de la fonction f définie par

$$f(x) = \cos(x)\exp(-\frac{x}{5}) + 1$$

On peut d'ailleurs calculer la valeur exacte de cette intégrale.

1.1 Méthodes déterministes.
On peut approcher I par une intégrale de Riemann, en discrétisant l'intervalle [a, b]. On peut alors arbitrairement construire les deux approximations suivantes :

$$I_1(n)=\sum_{i=1}^{n-1}(x_{i+1}-x_i)f(x_i) \qquad I_2(n)=\sum_{i=1}^{n-1}(x_{i+1}-x_i)f(x_{i+1})$$

Ou x_1=a et x_n=b.

Les techniques dites de Monte-Carlo

Si on considère une méthode d'intégration sur R, nécessitant N opérations pour obtenir le résultat désiré, il est facile de voir que les algorithmes précédents appliqué au calcul d'une intégrale sur R^P demanderont de l'ordre de N^P opérations.
Ceci rend en général ce type d'approche inexploitable si p>2. Il existe un moyen, permettant d'évaluer des intégrales multidimensionnelles en conservant un nombre raisonnable d'évaluation de la fonction (avec en contrepartie un résultat de précision modérée) : ce sont les techniques dites de Monte-Carlo.

Méthode de Monte-Carlo

La méthode est basée sur le résultat probabiliste suivant :
Si x_n est une suite de variable aléatoires indépendantes de même loi, avec $E(|x_1|)<\infty$,alors :

$$\lim_{n\to\infty}\frac{1}{n}\sum_{p=1}^{n}x_p=E(x_1).$$

Presque surement.
Ce résultat signifie en pratique que si l'on choisit un générateur de valeurs aléatoires donnant des points distribués uniformément dans « l'hyper cube » $[0,1]^P$(comme on peut en réaliser à l'aide de la fonction Matlab rand) et que (x_i) est la suite des points de R^p délivrée par ce générateur on a :

$$\lim_{n\to\infty}\frac{1}{n}\sum_{p=1}^{n}f(x_p)=\int_0^1\int_0^1.....\int_0^1 f(u_1,u_2,....u_p)du_1du_2.....du_p$$

Et plus en généralement si on prend un hyper-rectangle quelconque $I=\prod_1^P[a_i,b_i]$ d'hyper volume $V=\prod_1^P|b_i-a_i|$, alors on a :

$$\lim_{n\to\infty}\frac{1}{n}\sum_{p=1}^{n}f(x_p)=\frac{1}{V}\int_{a_1}^{b_1}\int_{a_2}^{b_2}.....\int_{a_n}^{b_n} f(u_1,u_2,.....u_p)du_1du_2.....du_p$$

Pourvu que les (x_p) aient une distribution uniforme sur l'hyper volume.
L'erreur commise sur l'intégrale peut être estimée à l'aide de l'écart type de la moyenne des $f(x_n)$ et calculée statiquement à l'aide des $f(x_p)$. Sa valeur est le produit de V par l'écart type statique des $f(x_p)$, divisé par la racine carrée du nombre des points. L'écart type est donné par la fonction matlab « std ».

Exercice

1. Ecrivez une fonction Matlab permettant de générer une matrice m x n de nombres aléatoires. Chaque ligne contenant n nombres uniformément répartis sur un intervalle donné.

2. Ecrivez une fonction Matlab permettant de calculer une intégrale multiple sur un huper-rectangle.

Solution

1. Matrices aléatoires

```
function x=random(lims, n) %Génération d'une matrice de nombre aléatoire à
                           %n colonne le nombre de ligne m est  déterminé
                           %par lims, chaque ligne possédant une
                           %distribution uniforme dans un intervalle donné
                           %lims est un tableau à deux colonnes et m lignes
                           %chaque ligne contient les bornes d'un intervalle
                           % m est le nombre d'intervalle
                           % x posséde alors m ligne et n colonnes
                           % lims(i,1) < = x(i,j) < = lims (i,2)

x=rand(size(lims,1),n);    % declaration de la dimension de x
lims(:,2)=lims(:,2)-lims(:,1);%calcul la différence de chaque intervalle
x=lims(:,ones(1,n))+x.*lims(:,2.*ones(1,n)) % détermine les valeurs de x
                    % y, z
```

```
Lims=
[0  3] % x est dans cet intervalle
[-2 2] % y est dans cet intervalle
[-2 2] % z est dans cet intervalle
```

Pour l'appel de la fonction x=random(lims,n)

```
x=random([0 -2 -2; 3 2 2]',10000) ;
```

avec s'écrit lims'=[0 -2 -2; 3 2 2]' % lims transposé

---

2. Méthode de monte-carlo

```
function [int, st]=monte_carlo(func, range,nbpt);%méthode de monte carlo
                                  %d'intégrale multiples
                                  %func : fonction d'un
                                  %vecteur x y=func(x)
                                  %range : matrice nx2 n
                                  %nombre de variables de
                                  %func (dimension de x)
                                  %nbpt : nombre de points
                                  %d'évaluation :défaut 10000
                                  % int : intégrale de func
                                  % sur l'hyper volume défini
                                  % par range
                                  %st : écart type de la
                                  %moyenne, indiquant
```

```
                                %l'erreur commise
if nargin <=2
   nbpt=10000;
end;
vol = prod(range(:,2)-range(:,1));
z=feval(func, random(range,nbpt));
if nargout == 2
   st=vol*std(z)./sqrt(nbpt); %std : écart type function matlab
end;
int=vol.*mean(z); % mean : la moyenne fonction matlab
```

Fonction à utiliser

```
function y=ff(x)
y=x(1,:).^2+x(2,:).^2+x(3,:).^2
```

Exécution :

```
>> [int, st]=monte_carlo('ff', [0 -2 -2;3 2 2]',10000) ;

int =
  270.9255

st =
   1.5191
```

Corrélations et régression

Ajustement linéaire

Corrélation :

La corrélation étudie les relations entre séries numériques. En effet, on observe couramment dans la réalité qu'il existe des liaisons entre deux ou plusieurs phénomènes économiques ou autres.

Exemple :

L'évolution du prix du ciment durant une période T est caractérisée par une suite de valeurs xi

$$X = \{x_1, x_2, \ldots\ldots, x_i, \ldots\ldots x_n\}$$

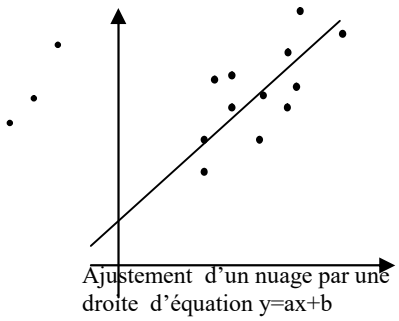
L'évolution des quantités produites pendant la même période T est donnée par la série :

$$Y = \{y_1, y_2, \ldots\ldots, y_i, \ldots\ldots y_n\}$$

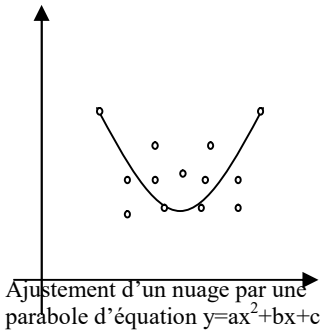
La corrélation va essayer d'établir une relation entre le prix du ciment et les quantités produites. En d'autres termes la corrélation va traiter d'un nouvel ensemble formé à l'aide des valeurs x_i et y_i.

$$Z=\{ x_i , y_i \}$$

Ce nouvel ensemble constitue un nuage statistique qu'il est possible d'ajuster par des fonctions mathématiques simples (droite, parabole)



Ajustement d'un nuage par une droite d'équation y=ax+b



Ajustement d'un nuage par une parabole d'équation $y=ax^2+bx+c$

L'ajustement :

Consiste donc à remplacer un nuage statistique, par une courbe représentative dont l'équation est connue.

L'ajustement linéaire

L'ajustement est dit linaire si le nuage statistique est représenté dans le référentiel yox par une droite d'équation $y=a_0+ax$

Point moyen d'un nuage

Le point moyen « M » d'un nuage est donnée par :

$$M(\bar{x} = \sum \frac{x_i}{n}, \bar{y} = \sum \frac{y_i}{n})$$

Pour trouver a de l'équation $y=ax+a_0$ on le calcul par :

$$a = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2} \text{ et } a_0 = \bar{y} - a\bar{x}$$

De même pour la droite $x=by+b_0$

$$b = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum y_i^2 - n\bar{y}^2} \text{ et } b_0 = \bar{x} - a\bar{y}$$

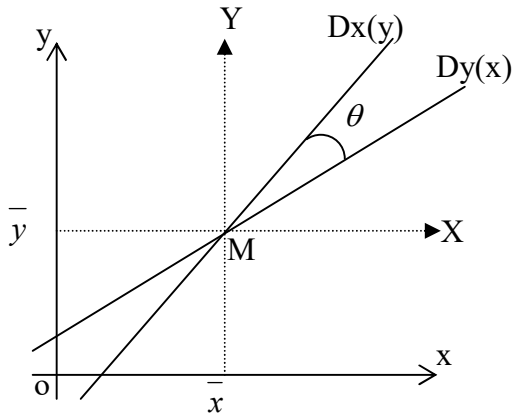
Coefficient de corrélation

$$r^2 = ab$$

$$r^2 = \frac{[\sum x_i y_i - n\bar{x}\bar{y}]^2}{[\sum x_i^2 - n\bar{x}^2][\sum y_i^2 - n\bar{y}^2]}$$

Angle de régression

L'angle de régression θ, est l'angle formé par les droites Dy(x) et Dx(y).



$$\theta = Arctg\frac{1}{b} - Arctg(a)$$

Programme ajustement linéaire

```
function [a b a0 b0 cor teta]=ajust_Correl_Lineaire(n)
XY=zeros(n+1,6)
X=rand(n,1)
Y=rand(n,1)

i=1:n;
XY(i,1)=fix(i);
XY(i,2)=X(i,1)
XY(i,3)=Y(i,1)
for i=1 :n
   XY(i,4)=XY(i,2).*XY(i,3)
   XY(i,5)=XY(i,2).*XY(i,2)
   XY(i,6)=XY(i,3).*XY(i,3)
   XY(n+1,2)=XY(n+1,2)+XY(i,2)
   XY(n+1,3)=XY(n+1,3)+XY(i,3)
   XY(n+1,4)=XY(n+1,4)+XY(i,4)
   XY(n+1,5)=XY(n+1,5)+XY(i,5)
   XY(n+1,6)=XY(n+1,6)+XY(i,6)
end
% Calcul des moyenne
  MoyenneX=XY(n+1,2)/n
  MoyenneY=XY(n+1,3)/n
% Calcul des coefficients a et b
NU=XY(n+1,4)-n*MoyenneX*MoyenneY
DE=XY(n+1,5)-n*MoyenneX*MoyenneX
a=NU/DE
DE1=XY(n+1,6)-n*MoyenneY*MoyenneY
b=NU/DE1

%Calcul de a0 et b0
a0=MoyenneY-a*MoyenneX
b0=MoyenneX-b*MoyenneY

%Calcul coefficient de corrélation
R1=NU*NU;
R2=DE*DE1;
cor=sqrt(R1/R2)
%calcul angle de regression
teta=atan(1/b)-atan(a)
```

L'ajustement parabolique

La parabole des moindres carrés

Un nuage statique peut être ajusté par une parabole ayant pour équation :
$y = a_0 + a_1 x + a_2 x^2$, a_0, a_1, a_2 sont des constantes à déterminer pour cela on utilise le système suivant :

$$\sum y = a_0 N + a_1 \sum x + a_2 \sum x^2$$
$$\sum xy = a_0 \sum x + a_1 \sum x^2 + a_2 \sum x^3$$
$$\sum x^2 y = a_0 \sum x^2 + a_1 \sum x^3 + a_2 \sum x^4$$

Qui s'appelle les équations normales de la parabole des moindres carrés

Algorithme de calcul

Soient deux sériés numériques :

$$X = \{x_1, x_2, \ldots\ldots, x_i, \ldots\ldots x_n\}$$
$$Y = \{y_1, y_2, \ldots\ldots, y_i, \ldots\ldots y_n\}$$

Pour ajuster l'ensemble des données par une parabole des moindres carrés

$$y = a_0 + a_1 x + a_2 x^2$$

On doit les étapes suivantes de l'algorithme :

X	y	x^2	x^3	x^4	xy	x^2y
Σ x	Σ y	Σx^2	Σx^3	Σx^4	Σ xy	Σx^2y

Calculer les déterminants

1°) $\Delta = \begin{bmatrix} N & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{bmatrix}$ $\quad Y = \begin{bmatrix} \sum x \\ \sum xy \\ \sum x^2 y \end{bmatrix}$

2°) $\Delta_0 = \begin{bmatrix} \sum x & \sum x & \sum x^2 \\ \sum xy & \sum x^2 & \sum x^3 \\ \sum x^2 y & \sum x^3 & \sum x^4 \end{bmatrix}$ $\quad \Delta_1 = \begin{bmatrix} N & \sum x & \sum x^2 \\ \sum x & \sum xy & \sum x^3 \\ \sum x^2 & \sum x^2 y & \sum x^4 \end{bmatrix}$ $\quad \Delta = \begin{bmatrix} N & \sum x & \sum x \\ \sum x & \sum x^2 & \sum xy \\ \sum x^2 & \sum x^3 & \sum x^2 y \end{bmatrix}$

3°) Résoudre le système d'équations et déterminer les coefficients

$$a_0 = \frac{|\Delta_0|}{\Delta} \qquad a_1 = \frac{|\Delta_1|}{\Delta} \qquad a_2 = \frac{|\Delta_2|}{\Delta}$$

Ecrire l'équation

$$y = a_0 + a_1 x + a_2 x^2$$

Tracer l'équation

$$y = a_0 + a_1 x + a_2 x^2$$

Exercice en chimie

Dans une solution d'une mole de pyridine dans du benzonitrile, on a mesuré la constante k (de pseudo premier ordre) pour la réaction avec l'iodure de méthyle, ceci à (températures différents :

	1	2	3	4	5
$t_1{}^0c$	0.00	25.0	39.9	60.0	80.0
10^5k	3.59	30.4	91.8	340	1120

Si on pose $y_i = \ln(\frac{k_i}{T_i})$ et $x_i = \frac{1}{T_i}$ (T_i est la température absolue $T_i = t_i + 273$), on sait (en théorie) que la relation suivante est valide :

$$y = (\frac{\Delta S^{++}}{R} - 23.8) - \frac{\Delta H^{++} x}{R}$$

Calculez $\frac{\Delta S^{++}}{R}$ et $\frac{\Delta H^{++}}{R}$ au sens des moindres carrés.

Programme associe :

```
K=[3.5900  30.4000  91.8000  340.000  1120.000];
T=[0.00  25.0  39.9  60.0  80.0]+273;
X=1./T;
Y=log(10^(-5)*K./T);
m=[X'  ones(size(X'))];
xx=m\Y';
xx(2)=xx(2)+23.8
```

Fonctions générales sur les chaînes :

- double : conversion d'une chaîne vers le code ASCII
  Exemple
  ```
  >> z=double('A')
  z =
      65
  ```
- char : conversion d'un code vers une chaîne
  Exemple
  ```
  >> z=char([70 101 122 97 110 105])
  z =
  Fezani
  ```

- ischar : test de type
  ischar(s) est vrai si s est une chaîne faux (à) sinon. ischar (double(x)) est toujours faux et ischar(char(x)) est toujours vrai.

Exemple

```
>> ischar('fezani')
ans =
```

```
     1

>> ischar(65)
ans =
    0
```

- blanks : génération de chaînes de blancs.

Exemple

>>blancs(10) génère une chaîne de 10 blancs.

- Str2mat ou strvcat
  Création d'une matrice de chaînes.

Comparaison des chaînes

Il existe en Matlab des fonctions permettant la comparaison de chaînes et la conversion entre chaînes et nombres.

- strcomp (s1 ,s2): renvoie 1 si les deux chaînes sont identiques, 0 sinon.

Exemple

```
>> strcmp('fezani', 'FEZANI')
    ans =
        0

>> strcmp('fezani', 'fezani')
    ans =
        1
```

- Ind=strmatch(str, strs) :trouve les chaînes correspondants

Exemple

```
>> i=strmatch('max',strvcat('max','min', 'maximum'))
   i =
        1
        3
>> i=strmatch('max',strvcat('max','min', 'maximum'),'exact')
 i =
        1
```

- Ind=strmatch(str,strs, 'exact') ne renvoie que les correspondances exactes.

Exemple

```
>> i=strmatch('max',strvcat('max','min', 'maximum'),'exact')
 i =
        1
```

- I=strfind(s1,s2) : recherche d'une sous chaîne, renvoie dans i l'ensemble des index de début de s2 dans s1.

Exemple

```
s=strfind('fezani est chez lui.','e')
s =
    2     8    14
```

- upper(s) : conversion en majuscules

Exemple

```
>> upper('fezani')
    ans =
          FEZANI
```

- lower (s) : conversion en minuscules.

Exemple

```
>> lower('MUSTAPHA')
    ans =
          mustapha
```

- isletter(c) : vrai si le caractère alphabétique accentué ou non.

Exemple

```
>> isletter('bonjour')
ans =
    1     1     1     1     1     1     1

>> isletter('é')
    ans =
          1
```

- isspace(s) : vrai pour les caractères « blancs ».

Exemple

```
>> isspace('Bonjour Moad')
    ans =
          0     0     0     0     0     0     0     1     0     0     0     0
```

- strrep(s1,s2,s3)

Exemple

```
>> strrep('Bonjour malek','jour','soir')
   ans =
     Bonsoir malek
```

Entrées et sorties

1 Entrées clavier

Une seule fonction d'entrée qui est input, sa syntaxe est :

Val=input(<text>)

```
>> val=input('Donnez la valeur de s = ')
Donnez la valeur de s = 3

val =
    3

>> val=input('Donnez la valeur de s = ')
Donnez la valeur de s = 'a'

val =
     a
```

Il est également possible d'écrire des lignes du style :

```
>> Prod=input('Entrez le premier facteur = ')*input('Entrez le second facteur = ')
Entrez le premier facteur = 2
Entrez le second facteur = 3
Prod =
      6
```

2) Menu simple

Menu :est un menu interactif simple à nombre arbitraire d'éléments, on l'utilise par :

Choix=menu(titre,s1,s2,…) ;

Exemple

```
>> s1=4
s1 =
    4

>> s2=9
s2 =
    9

>> choix=menu('Fezani', s1,s2)
```


```
choix =
    0
```

Autre exemple

```
>> choix = menu('Choisie une couleur','Rouge','verte','Bleu')
choix =
    1
```

3) L'instruction pause

Pause: attend la frappe d'une touché ou un nombre donné de secondes avant de poursuivre l'exécution.

Syntaxe

```
pause
pause(n)
pause on
pause off
```

Sortie écran
disp(a) ;

Exemple

```
>> x=1:5;
>> disp(x)
    1     2     3     4     5
```

Progdisp.m

```
disp('  col1    col2    col3');
disp(fix(100*rand(4,3)));
```

résultat :

```
col1    col2    col3
   67    65    27
   75    17     4
   74    70     9
   39     3    82
```

Entrées et sortie fichiers

Fonctions de bas niveau

-Ouverture d'un fichier

-fopen : ouverture d'un fichier

[fid, message]=fopen(nom, mode, machine)

-fid : est un entier qui constitue la « poignée » qui permettra l'utilisation du fichier après son ouverture

-message : est un message d'erreur renvoyé par le système si l'ouverture n'a pu se faire

- nom : est le nom sur disque du fichier

-mode : est l'une des chaînes décrites dans le tableau qui suit :

'r'	Lecture
'w'	Écriture (creation si necessaries).
'a'	Ajout(creation si nécessaire)
'r+'	Lecture et écriture (pas de création)
'w+'	Troncature ou création pour lecture et écriture
'a+'	Lecture et ajout (création si nécessaire).

Exemple:

En écriture

```
NomFic='g.txt';
fid=fopen(NomFic,'w');
```

En lecture

```
NomFic='g.txt';
fid=fopen(NomFic,'r');
```

Entrée/ sortie non formatées

Pour les données non formatées on utilise fread pour la lecture et fwrite pour l'écriture.

-fwrite : Ecriture des données binaires.

Nb=fwrite(fid, a,type,skip)

- nb : est le nombre d'éléments écrits avec succès.

-a : est la matrice à partir de laquelle les données sont écrites.

-fid : est la poignée du fichier sur lequel en écrit.

-type : est la chaîne déterminant l'interprétation et le nombre d'octets écrit pour chaque élément (voir fread)

- passe : est optionnel et représente un nombre d'octets à passer entre chaque élément écrit. Ce paramètre est pratique pour écrire des champs non contigus dans des fichiers d'enregistrements de taille constante.

-fread : Lecture des données binaires. Sa syntaxe est :

[a{,nb}]=fread(fid{{,taille}{,type}{,passe}})
-a : est la matrice dans laquelle se trouvera le résultat de la lecture.
-nb : est optionnel et contient le nombre d'éléments effectivements lus.
- fid : est la poignée du fichier qui doit avoir été ouvert préalablement (par fopen).
-taille : est optionnel et vaut inf par défaut, sinon le choix est permis entre :

* n : lit a éléments dans un vecteur colonne.
* inf : lit jusqu'à la fin du fichier.
* [m,n] :lit m x n éléments pour remplir par colonnes, une matrice de même taille.

-type : contrôle le nombre de bits lus pour chaque élément et son interprétation en tant que caractère, entier ou réel. Les valeurs de la chaîne type peuvent être les suivantes et correspondent à des types de données Matlab, C ou Fortran. Le défaut est 'uchar'(voir tableau ci après).
-pause est optionnel et représente en nombre d'octets à passer entre chaque élément lu. Ce paramètre est pratique pour lire des champs non contigus dans des fichiers d'enregistrements de taille constante.

Matlab	C ou Fortran	Description
'schar'	'signed char'	Caractère signé, 8 bits
'double'	'double'	Réel long,64 bits
'float32'	'real*4'	Réel, 32 bit
'float64'	'real*8'	Réel, 64 bit
'int8'	'integer*1'	Entier signé, 8 bits
'int16'	'integer*2'	Entier signé, 16 bits
'int32'	'integer*4'	Entier signé, 32 bits
'int64'	'integer*8'	Entier signé, 64 bits
'uchar'	'unsigned char'	Caractère positif, 8 bits
'uint8'	'integer*1'	Entier positif, 8 bits
'uint16'	'integer*2'	Entier positif, 16 bits
'uint32'	'integer*4'	Entier positif, 32 bits
'uint64'	'integer*8'	Entier positif, 64 bits
'intN'		Entier signé, N bits $(1 \le N \le 32)$
'uintN'		Entier positif, N bits $(1 \le N \le 32)$
les types suivants dépendent de la machine		
'char'	'char'	Caractère, 8 bits
'short'	'short'	Entier, 16 bits
'int'	'int'	Entier, 16 ou 32 bits

'long'	'long'	Entier, 32 bits
'float'	'float'	Réel, 32 bits
'ushort'	'unsigned short'	Entier positif, 16 bits
'ulong'	'unsigned long'	Entier positif, 32 bits
'uint'	'unsigned int'	Entier positif, 16 ou 32 bits

Exemple
Ecriture et lecture

```
NomFic='g.txt';
x=rand(3,4).*100
fid=fopen(NomFic,'w');
n=fwrite(fid,x,'char');
disp(n);
fclose(fid);
f=fopen(NomFic,'r');
[a,nb]=fread(f,n,'char')
```

Exemple

```
fid = fopen('fgetl.m');
while 1
   tline = fgetl(fid);
   if ~ischar(tline),   break,   end
   disp(tline)
end
fclose(fid)
```

Entrée/ sortie formatées

-fprintf

{Nb}=fprint({fid,}format, a,….)
- nb : est un argument optionnel qui renvoie le nombre d'octets effectivement écrits.
-a : est la matrice à partir de laquelle les données sont écrites.
-fid : est un argument optionnel qui est la poignée (possiblement obtenue par fopen) du fichier sur lequel on écrit. S'il est absent (ou égal à 1) c'est l'écran (la sortie standard) qui est utilisée, s'il est égal à 2 c'est l'erreur standard.
-format : est une chaîne de caractère permettant de préciser le mode d'impression de la matrice a et de tout autre paramètre subséquent.
Format contient des indications suivant les spécifications de conversion du langage C. ces spécifications utilisent le caractère % suivi de drapeaux d'options et de champs décrivant la précision et le type d'élément à afficher ainsi que des caractères de conversion :
- d, o, u, x, f, e, c, s

Cette fonction se comporte comme sa collègue ANSI C, avec quelques exceptions et extensions.

1. si le double fourni par matlab n'est pas convertible dans le type attendu, le format e est utilisé :
2. les types suivants sont supportés par les conversions o, u, x, X :
   - t on obtient un float au lieu d'un unsigned int.
   - b on obtient un double au lieu d'un unsigned int

Que signifient les indicateurs de conversions ? Leur utilisation peut être décrite dans la table suivante :

d	Décimal
o	Octal non signé
c	Caractère
s	Chaîne de caractères
u	Décimal non signé
x	Hexadécimal non signé
f	Flottant fixe [-]mmm.nnnnn
e	Flottant scientifique [-]m.nnnnnnE[+/-]xxx

Exemple 1

Fichier edit : printf1.m

```
x= [0 1 2];
y = [3 4 5];
%ouvre un fichier ou le créé
fid = fopen('test.txt','w');
%écrit dans ce fichier, fid est sa reference pour matlab
fprintf(fid,'%s\n','vecteur x');
fprintf(fid,'%i\t %i\t %i\n',x);
fprintf(fid,'%s\n','vecteur y');
fprintf(fid,'%i\t %i\t %i\n',y);
%n'oublie pas de fermer le fichier sinon tu ne peux pas le lire
fclose(fid);
```

Exemple 2

Fichier edit : printf2.m

```
%ouvre un fichier ou le créé
x = 0:.1:1;
y = [x; exp(x)];
fid = fopen('exp.txt','w');
%écrit dans ce fichier, fid est sa reference pour matlab
fprintf(fid,'%6.2f %12.8f\n',y);
fclose(fid);
```

-fscanf

[a{,nb}]=fscanf({fid,} format, nb)

-fscanf :lit des données à partir de fid suivant le format spécifie dans format et possiblement contrôlé par size. Les résultats lus sont dans a et facultativement le nombre d'octest lus se retrouve dans nb.
-a : est la matrice contenant le résultat de la lecture.
-nb : est un argument optionnel qui renvoie le nombre d'octets effectivement lus.
-fid : est un argument optionnel qui est la poignée (possiblement obtenue par fopen) du fichier sur lequel on lit. S'il est absent (ou égal à 0) c'est le clavier (l'entrée standard) qui est utilisé.
-format : est une chaîne de caractères permettant de préciser des modes de conversion. Format contient des indications suivant les spécifications de conversion du langage C. Ces spécifications utilisent le caractère % suivi de caractères de conversion :
- d, o, u, x, f, e, c, s

Entre le % et le caractère de conversion peuvent se trouver un ou plusieurs des caractères suivants :

1. astérisque (*) signifiant de ne pas stocker la valeur lue.
2. un entier signifiant une largeur maximale de champs
3. une lettre h ou 1 pour spécifier la taille(courte ou longue) de la destination

Cette fonction se comporte comme sa collègue ANSI C, avec quelques exceptions et extensions.

1. Si un caractère de conversion s est utilisé, un élément lu peut correspondre à plusieurs éléments matriciels chacun contenant un caractère.
2. Le mélange de caractères et d'éléments numériques produira une matrice uniquement numérique, les caractères étant 'remplacés' par leurs code ASCII.
3. La chaîne de format est recyclée jusqu' à remplir « a » à la taille choisie.

Exemple1

Fichier edit : scanf1.m

```
%ouvre un fichier en lecture
fid = fopen('test.txt','r');
%lit dans le fichier, fid est sa reference pour matlab
v1=fscanf(fid,'%s',2);disp(v1);
x=fscanf(fid,'%i %i %i');disp(x);
v2=fscanf(fid,'%s',2);disp(v2);
y=fscanf(fid,'%i %i %i');disp(y);
%n'oublie pas de fermer le fichier
fclose(fid);
```

Exemple2

Fichier edit :scanf2.m

```
%ouvre un fichier en lecture
fid = fopen('exp.txt');
%lit dans le fichier, fid est sa reference pour matlab
a = fscanf(fid,'%g %g',[2 inf]) % Maintenant, il y a deux lignes.
a = a';
```

%n'oublie pas de fermer le fichier
fclose(fid)

-textread :

Equations différentielles ordinaires aux conditions initiales
Objectifs

- Méthodes de base
  - Euler
  - Runge-kuta
- Méthodes effectives
  - méthodes adaptatives
  - méthodes proposées par matlab

I.1 Généralités

Les équations différentielles que nous envisageons de résoudre ici seront toujours de la forme :

$$y' = \frac{dy}{dy} = f(x, y)$$

Si x qui désigne la variable indépendante, est un réel, la variable y est un vecteur :
$y = (y_1, y_2, y_3, ..., y_n)$ et f est une fonction de $\Re^{n+1}$ dans $\Re^n$, $f = (f_1, f_2, f_3, ..., f_n)$.

Cette notation permet donc de représenter un système de n équations à n fonctions inconnues. La valeur n s'appelle l'ordre du système. Cette dénomination est cohérente avec celle rencontrée couramment dans l'étude analytique des équations différentielles, en effet le système :
Générateurs d'interface graphique Matlab
Guide est un générateur d'interfaces graphiques (GUI=Graphical User Interface). C'est-à-dire qu'il permet de créer une fenêtre munie de textes, de boutons, d'ascenseurs, etc.(objets graphiques appelés aussi « contrôles » et de donner le canevas d'un m-fichier dans lequel l'utilisateur n'a qu'à insérer les lignes de Matlab correspondant au commandes (callback)
Qu'il veut voir exécuter en retour d'action de la souris ou du clavier.
Voici la liste de ces contrôles et de leur dénomination anglo-saxonne :
Ascenseur :slider
Bouton bascule :radiobutton
Bouton poussoire :pushbutton
Cadres : frame
Case à cocher : checkbox
Listes : listbox
Menus déroulant : popupmenu
Texte éditable : edit
Texte statique : text
Zone de dessin : axes
Figure complète : figure
On va décrire le guide par un exemple

Graphiques

Graphiques 2D

Traçage de courbes

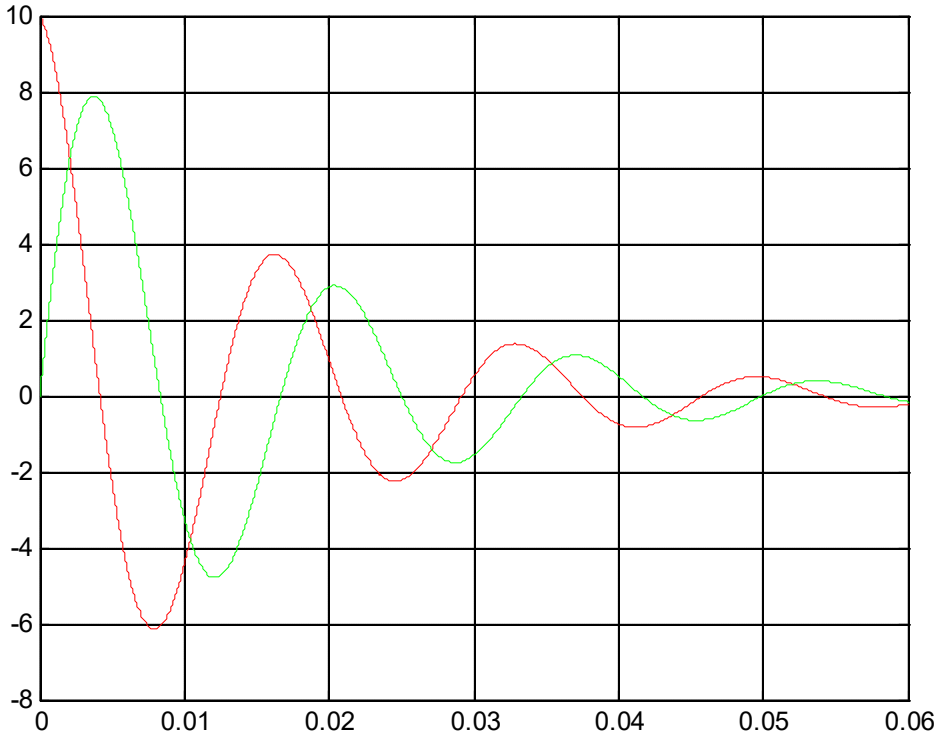
On utilise l'instruction plot pour tracer un graphique 2D:
plot(x,y) Tracer le vecteur y en fonction du vecteur x
plot(t,x,t,y,t,z) Tracer x(t), y(t) et z(t) sur le même graphique
plot(t,z,'r--') Tracer z(t) en trait pointillé rouge

Format de graphique

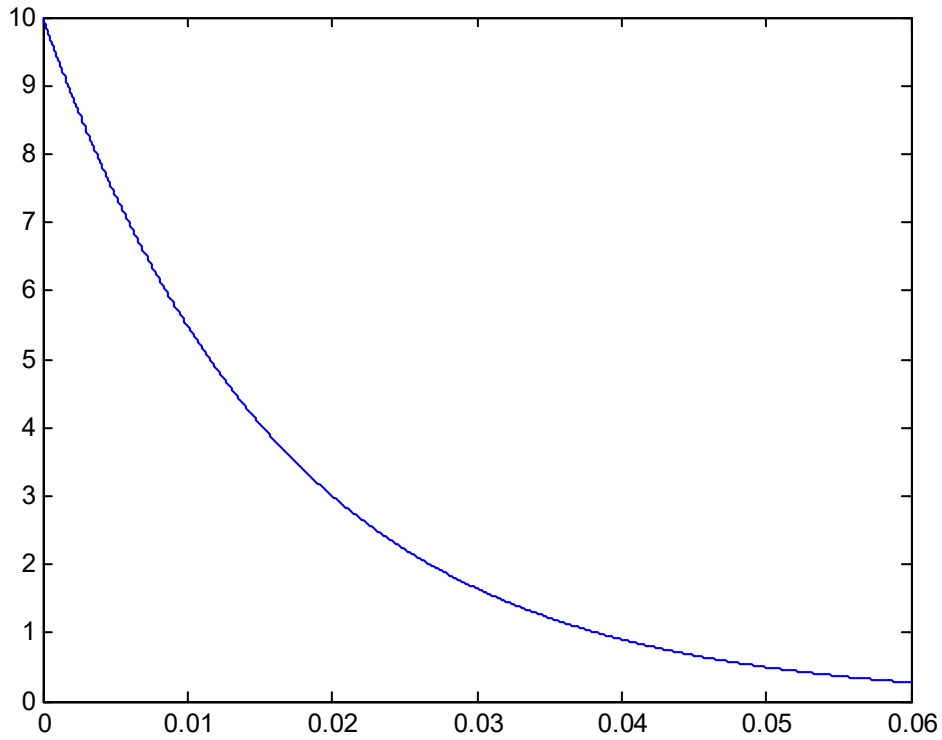
On peut choisir le format du graphique:
plot(x,y) Tracer y(x) avec échelles linéaires
semilogx(f,A) Tracer A(f) avec échelle log(f)
semilogy(w,B) Tracer B(w) avec échelle log(B)
polar(theta,r) Tracer r(theta) en coordonnées polaires
bar(x,y) Tracer y(x) sous forme des barres
grid Ajouter une grille

Exemple 5

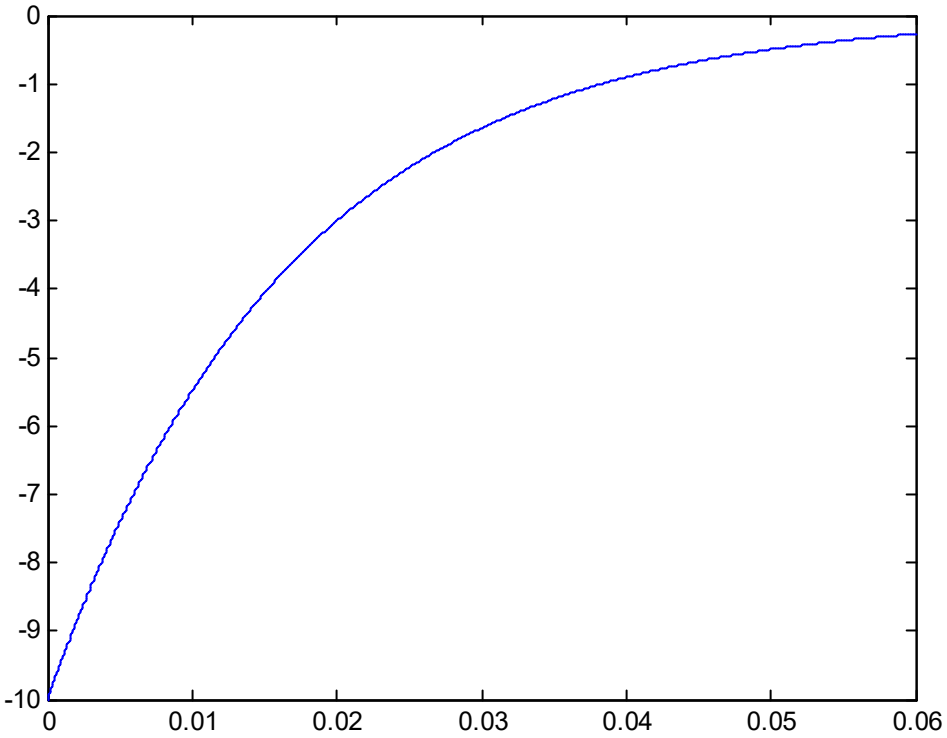
```
t=0:0.01e-3:0.06;
y=10*exp(-60*t).*cos(120*(3.14)*t);
z=10*exp(-60*t).*sin(120*(3.14)*t);
plot(t,y,'r',t,z,'g'),grid
a=10*exp(-60*t);
hold
```



```
plot(t,a,'b--')
```

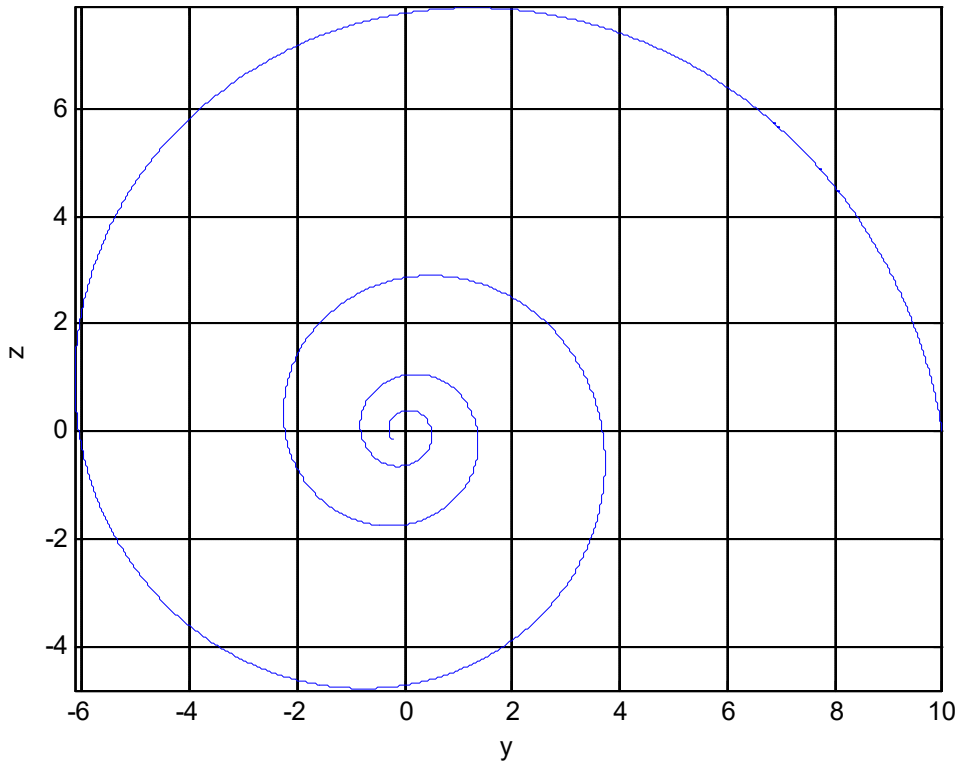



```
plot(t,-a,'b--')
```



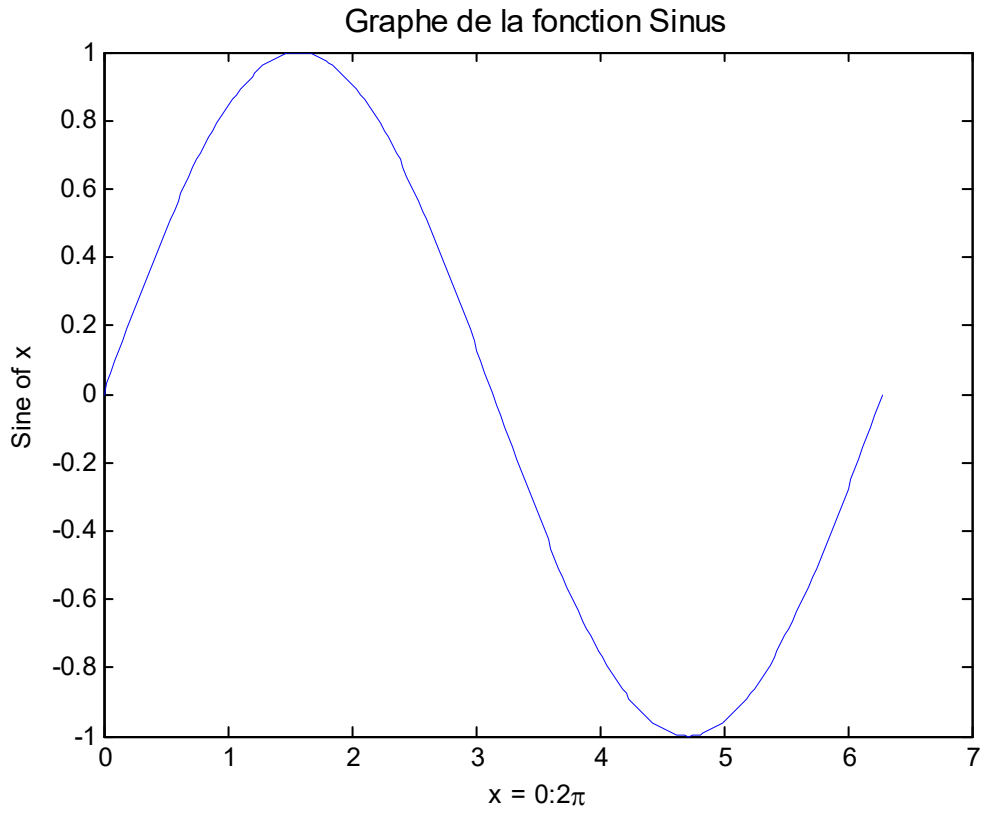

```
title('Fonctions sinusoidales amorties')
xlabel('Temps , s'),ylabel('Tension , V')
hold off
```

```
plot(y,z),grid
axis equal
xlabel('y'),ylabel('z')
```



Exemple

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
xlabel('x = 0:2\pi')
ylabel('Sine of x')
title('Graphe de la fonction Sinus','FontSize',12)
```



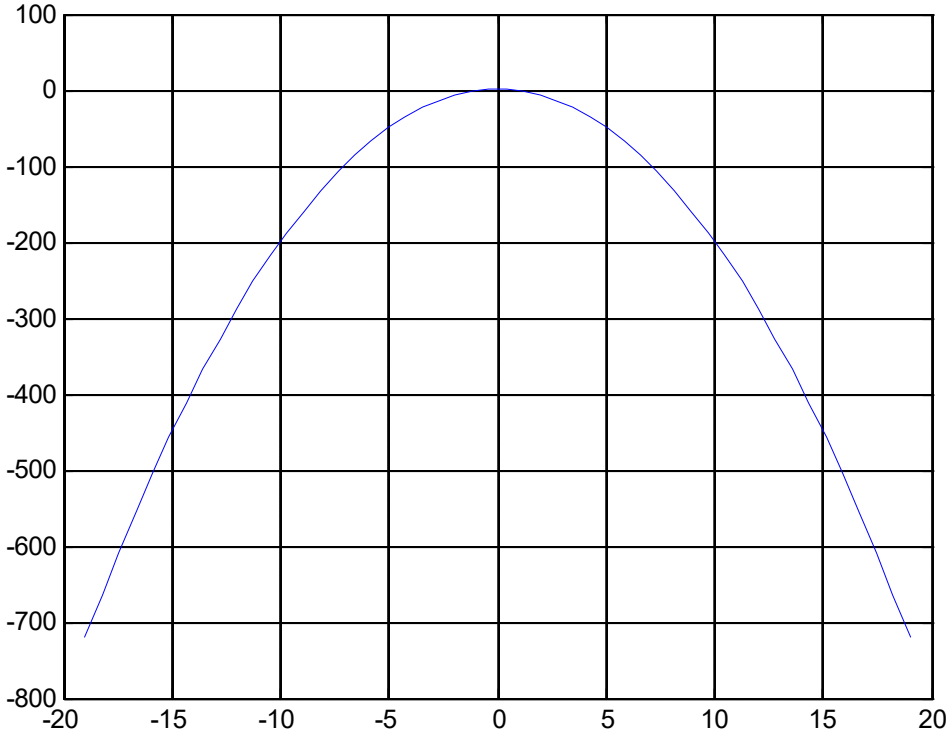
Exemple

Tracer la fonction

$y = -2x^2 + 1$ dans l'intervalle [-19,19], on utilisera les fonctions linspace, plot, et l'opérateur terme à terme « .^ ».

Solution

```
x=linspace (-19,19, 50) ;
y= -2*x.^2+1 ;
plot(x,y) ;grid on
```



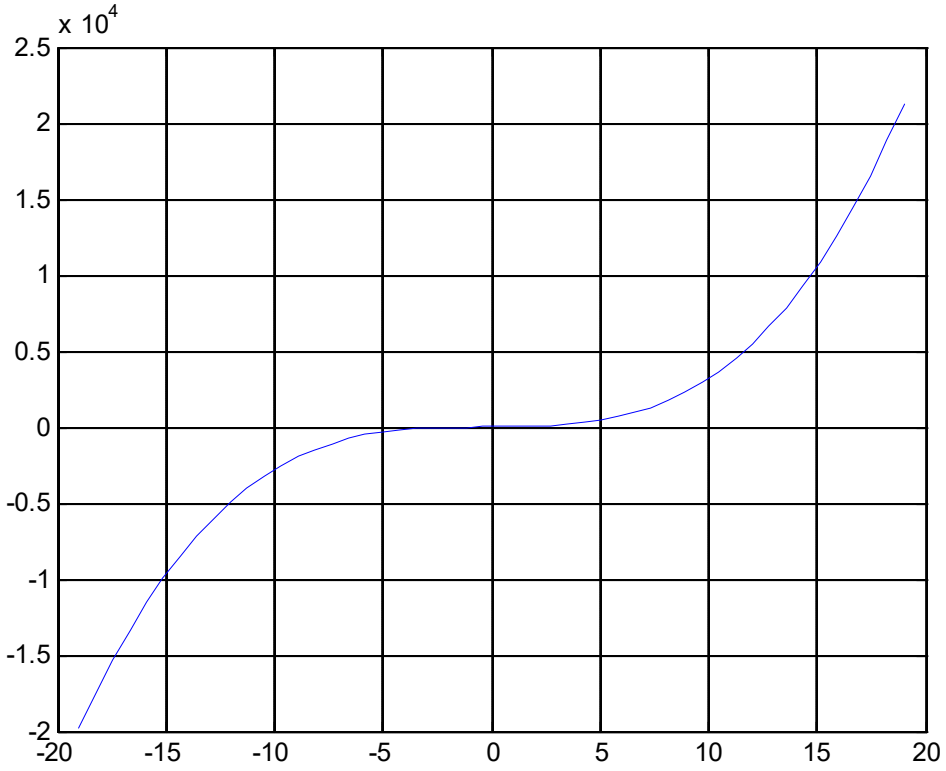
Exemple

Tracer la fonction

$y = 3x^3 + 2x^2 + 1$ dans l'intervalle [-19,19], on utilisera les fonctions linspace, plot, et l'opérateur terme à terme « .^ ».

Solution

```
x=linspace (-19,19, 50) ;
y= 3*x.^3+2*x.^2+1 ;
plot(x,y) ;grid on
```



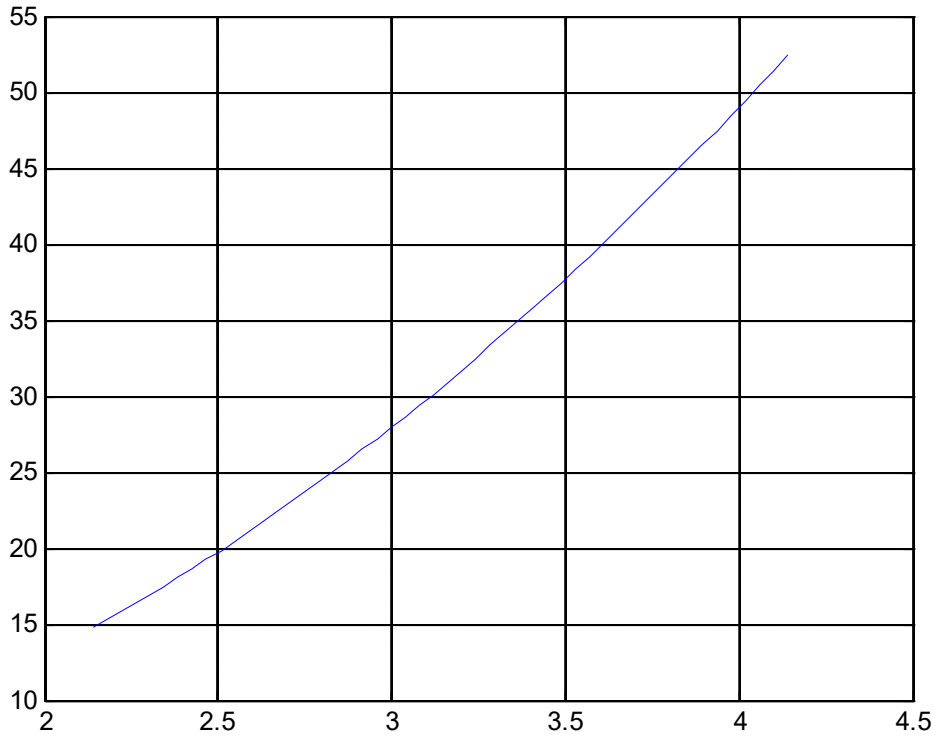
Exemple

Tracer la fonction

$y = 3x^2 + \frac{\ln(x-\pi)^2}{\pi^4} + 1$ dans l'intervalle [л-1,л+1], on utilisera les fonctions linspace, plot, et l'opérateur terme à terme « .^ ».

Solution

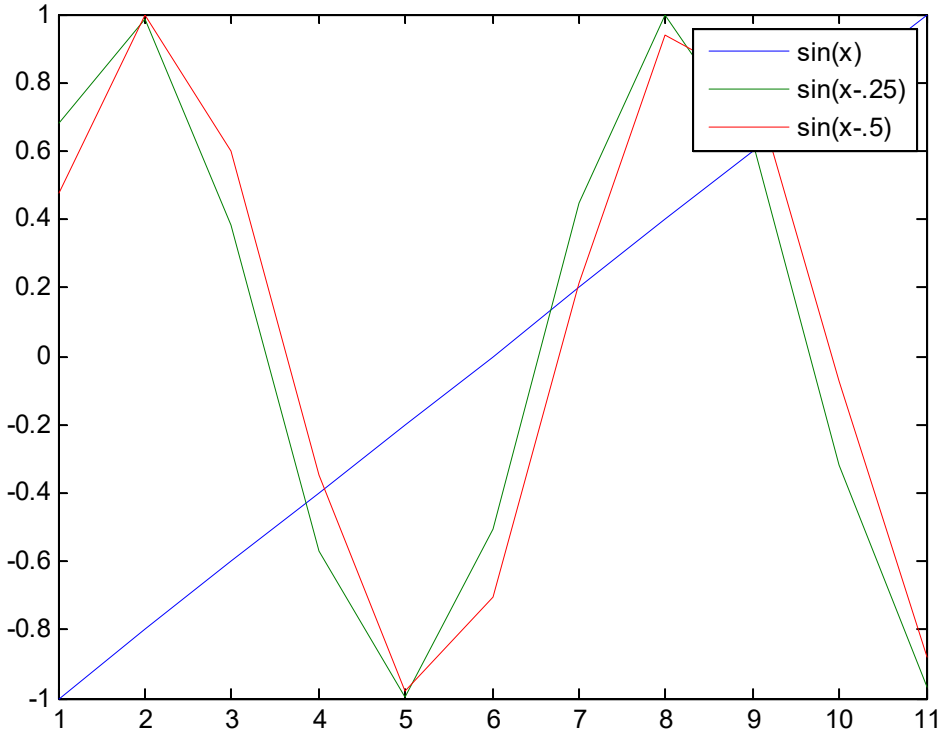
```
x=linspace (3.14-1,3.14+1, 50) ;
y= 3*x.^2+log((x-3.14).^2)/((3.14).^4)+1 ;
plot(x,y) ;grid on
```



Plusieurs données sur un même graphe
Plusieurs couples x-y créent plusieurs graphes avec l'appel à un seul plot.

Exemple

```
x=1:11
y=-1:0.2:1
y2 = sin(x-.25);
y3 = sin(x-.5);
plot(x,y,x,y2,x,y3)
legend('sin(x)','sin(x-.25)','sin(x-.5)')
```



Options

On utilise les options pour changer les couleurs des courbes ou bien les marqueurs. La syntaxe est :

plot(x,y,'color_style_marker')

color_style_marker est une chaîne qui contient de 1 à 4 caractères construite avec les spécifications de couleurs,
de style de lignes et de type de marqueurs.

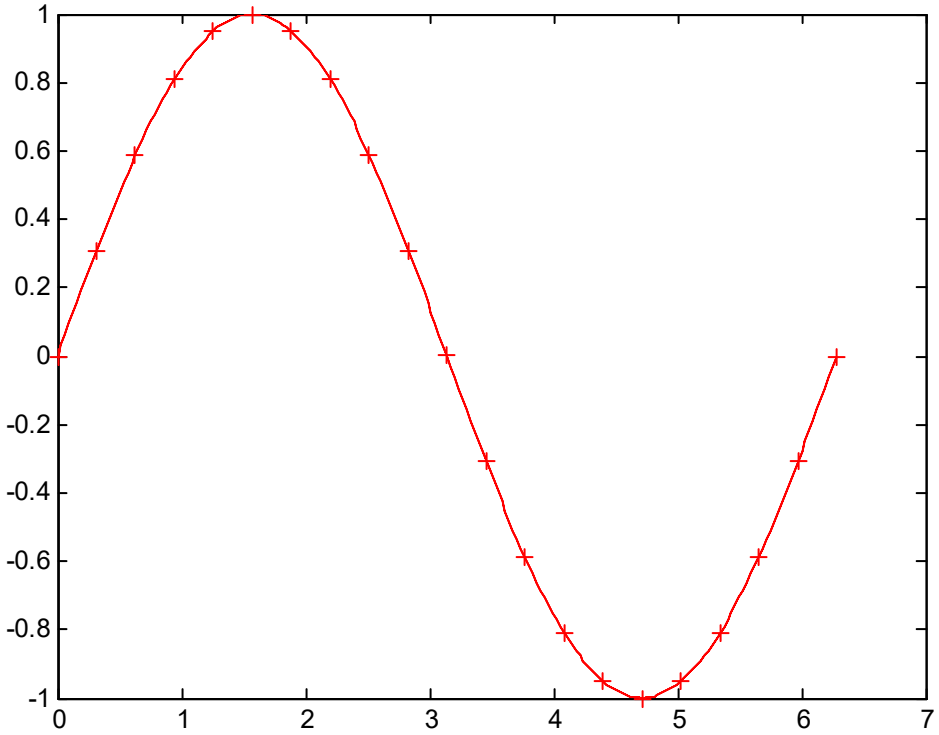
– couleurs : 'c', 'm', 'y', 'r', 'g', 'b', 'w', and 'k'. Ceci correspond à cyan, magenta, jaune, rouge, vert, bleu, blanc, et noir.

– Style de ligne : '-' pour solide, '–' pour des pointillés sous forme de traits, ' :' pour des pointillés, '-.' Pour des pointillés alternés. Il suffit de ne pas mettre de style de ligne pour ne pas tracer de lignes.

– Style de marqueurs : '+', 'o', '*', et 'x', 's' pour des carrés, 'd' pour des losanges, '^' pour des triangles,
'v' pour des triangles vers le bas, '>' pour des triangles vers la droite, '<' pour des triangles vers la gauche,
'p' pour des pentagrammes, 'h' pour des hexagrammes, et rien pour ne pas avoir de marqueurs

Exemple

```
x1 = 0:pi/100:2*pi;
x2 = 0:pi/10:2*pi;
plot(x1,sin(x1),'r:',x2,sin(x2),'r+')
```

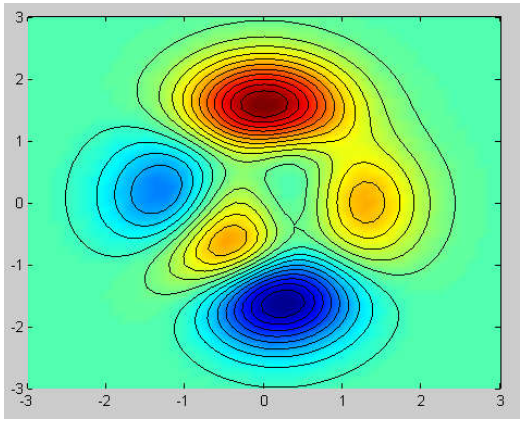


Pour ajouter des graphiques à un graphe déja existant, il suffit de taper la commande hold on. Pour suspendre son utilisation, taper simplement hold off.

Pour effacer un graphique, utiliser la commande clf.

Exemple

```
[x,y,z] = peaks;
contour(x,y,z,20,'k')
hold on
pcolor(x,y,z)
shading interp
hold off
```



Remarque

Si l'on désire des fenêtres supplémentaires pour effectuer des trac´es, il suffit de taper figure(n) ou n est le numéro de la figure.

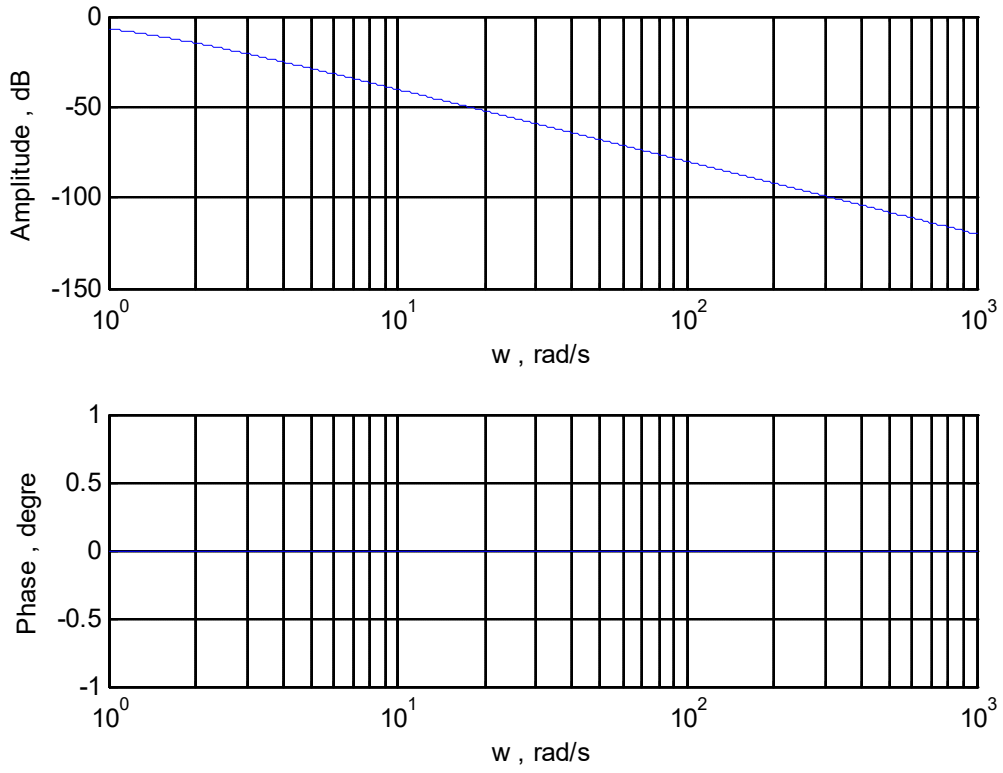
Graphique multiple

On peut tracer plusieurs graphiques dans la même fenêtre en utilisant l'instruction subplot pour diviser la fenêtre en plusieurs parties.

- Diviser la fenêtre en deux parties (2 x 1)

Exemple 6

```
>> n=1000;
>> w=(10).^ [0+(0:n-2)*(3-0)/(floor(n)-1), 3];
>> s=j*w;
>> H=225./(s.*s+3*s+225);
>> AdB=20*log10(abs(H));
>> phase=angle(H)*(180/(3.14));
>> subplot(2,1,1),semilogx(w,AdB),grid
>> xlabel('w , rad/s'),ylabel('Amplitude , dB')
>> subplot(2,1,2),semilogx(w,phase),grid
>> xlabel('w , rad/s'),ylabel('Phase , degre')
```

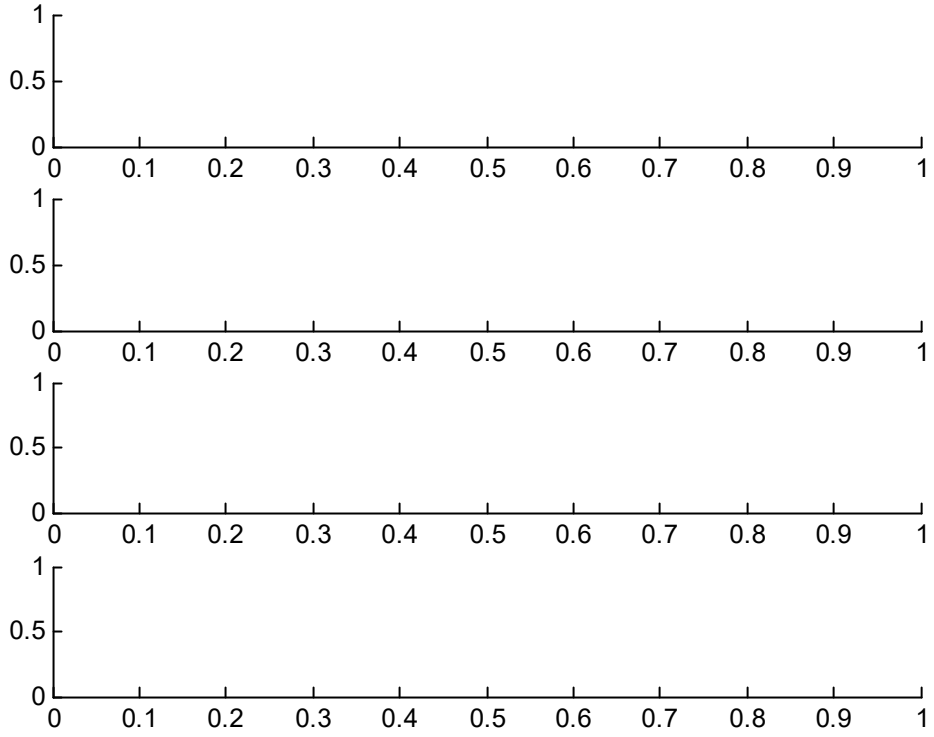


subplot(2,1,1)
subplot(2,1,2)

- Diviser la fenêtre en deux parties (1 x 2)
- Diviser la fenêtre en quatre parties (2 x 2)
- Diviser la fenêtre en quatre parties (4 x 1)

Ajout du texte au graphique

title('Titre du graphique') %Donner un titre au graphique
xlabel('Temps') %Étiquette de l'axe x
ylabel('Tension') %Étiquette de l'axe y
gtext('Valeur absolue') %Ajouter du texte au graphique avec la souris
subplot(1,2,1) ;subplot(1,2,2);
subplot(2,2,1); subplot(2,2,2);
subplot(2,2,3); subplot(2,2,4);
subplot(4,1,1) ;
subplot(4,1,2) ;
subplot(4,1,4) ;
subplot(4,1,3) ;



Manipulation de graphiques

axis([-1 5 -10 10]) Choix des échelles x = (-1,5) et y = (-10,10)
hold Garder le graphique sur l'écran (pour tracer plusieurs courbes sur le même graphique)

Impression et enregistrement de graphiques

print -dps Imprimer le graphique en PostScript
print -dpsc Imprimer le graphique en PostScript Couleur
print -dps dessin.ps Enregistrer le graphique en PostScript dans le fichier dessin.ps

Graphiques 3D

Matlab définit une surface par la coordonnées z des points sur une grille dans le plan x-y. Les commandes mesh et surf affichent des surfaces en 3 dimensions. mesh produit un affichage en lignes tandis que surf colorie en plus les facettes.

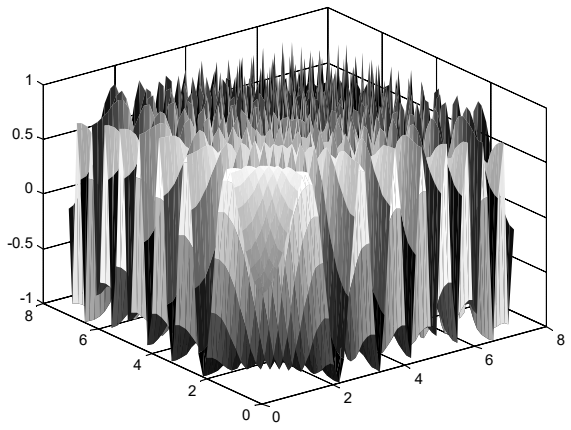
Pour afficher une fonction de deux variables z = f(x, y), il faut générer la liste des points x et y (qu'on appelle maillage) et utiliser x et y pour évaluer la fonction f. La fonction meshgrid transforme le domaine spécifié par un simple vecteur ou par deux vecteurs en maillages.

Le traçage des graphiques 3D est illustré dans les deux exemples suivants.

Exemple
Tracez la surface
$x = \sin(x^2 + y^2)$ pour 0<x, y<4$\sqrt{\lambda}$
On utilisera les fonctions linspace, meshgrid, surf , et l'opérateur terme à terme « .^ ».

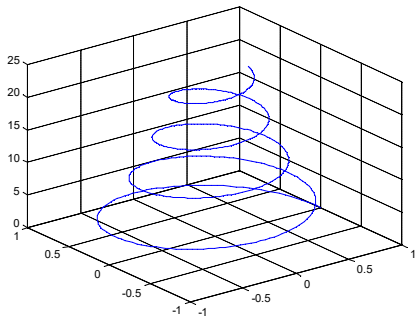
Solution

```
t= linspace (0,(4*sqrt(3.14)), 50) ;
[x,y]=meshgrid(t,t) ;
z=sin(x.^2+y.^2) ;
surf(x,y,z) ;
colormap(Gray) ;
shading('interp') ;
```



Exemple 7

```
t = 0:0.05:25;
 x = exp(-0.05*t).*cos(t);
 y = exp(-0.05*t).*sin(t);
 z = t;
 plot3(x,y,z), grid
```



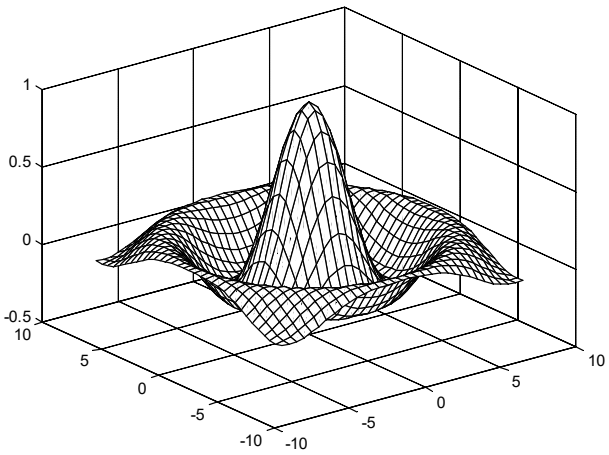
Meshgrid

La fonction meshgrid transforme le domaine spécifié par deux vecteurs, X et Y, dans des matrices X et Y. Vous pouvez alors utiliser ces matrices pour évaluer les fonctions de deux variables. Les lignes de X sont des copies du vecteur X et les colonnes de Y sont des copies du vecteur y.

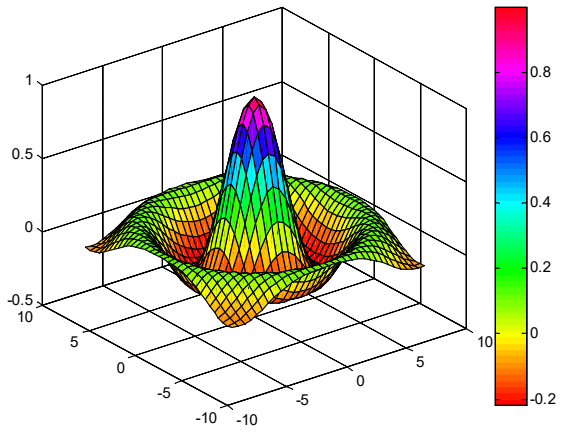
```
[X,Y] = meshgrid(1:3,10:14)

X =

     1     2     3
     1     2     3
     1     2     3
     1     2     3
     1     2     3


Y =

    10    10    10
    11    11    11
    12    12    12
    13    13    13
    14    14    14
```

Exemple

```
[X,Y] = meshgrid(-8:.5:8)
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z,'EdgeColor','black')
```



```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
surf(X,Y,Z)
colormap hsv
colorbar
```



Représentation de la matrice comme une surface

Traçage des maillages et surfaces

Les commandes mesh et surf tracent des surfaces 3D à partir d'une matrice de donnée. Si Z est une matrice pour laquelle les éléments de Z (i, j) définir la hauteur d'une surface :

mesh(Z)

Syntaxe :

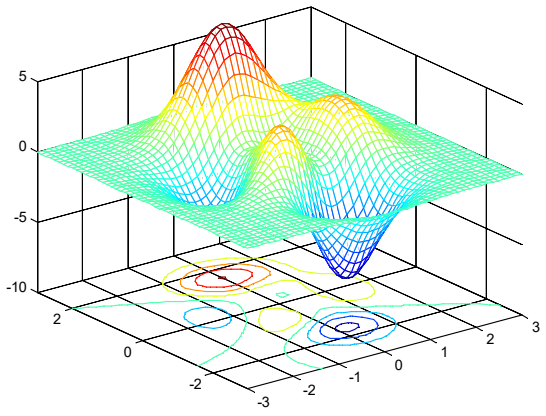
```
mesh(X,Y,Z)
 mesh(Z)
mesh(...,C)
mesh(...,'PropertyName',PropertyValue,...)
mesh(axes_handles,...)
```

meshc(X,Y,Z);
meshz(X,Y,Z);

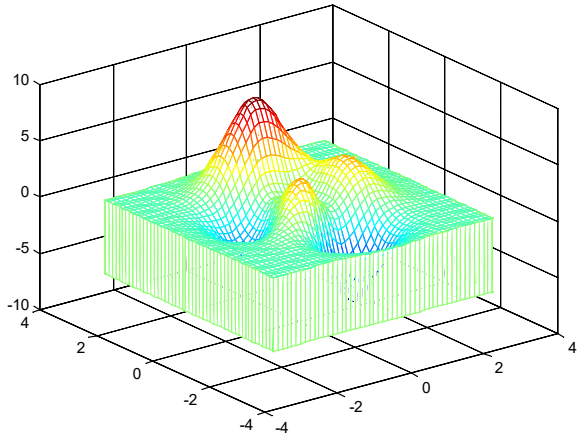
……
Génère une surface maillée coloré affiché en 3-D.

Exemple

Produit une combinaison maillage et contour au niveau des pics de surface:
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);
axis([-3 3 -3 3 -10 5])



Meshz(X,Y,Z)
Trace des maille plus le plan zéro



De même, pour

surf(z)

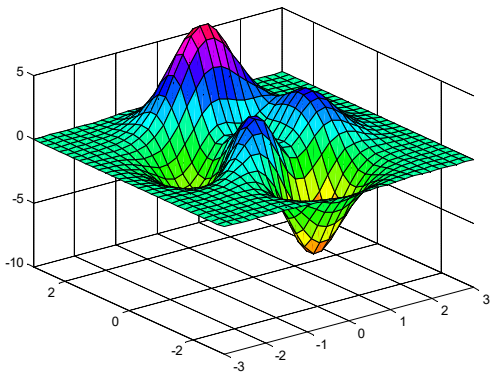
Génère des facettes de surfaces et les affiches en 3D
Génère des facettes colorées de la surface et l'affiche en 3-D. Ordinairement, les facettes sont des quadrilatères, dont chacune est une couleur constante, décrite avec des lignes de

maille noire, mais la commande shading vous permet d'éliminer les lignes de maillage (ombrage flat) ou de sélectionner interpolées ombrage à travers la facette (ombrage interp).

Syntaxe
surf(Z)
surf(Z,C)
surf(X,Y,Z)
surf(X,Y,Z,C)
surf(...,'PropertyName',PropertyValue)
surf(axes_handles,...)
….

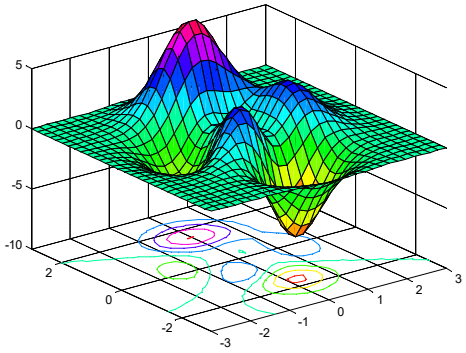
Exemple
Afficher une parcelle de surface et des courbes au niveau des pics de surface.

```
 [X,Y,Z] = peaks(30);
surf(X,Y,Z)
colormap hsv
axis([-3 3 -3 3 -10 5])
```



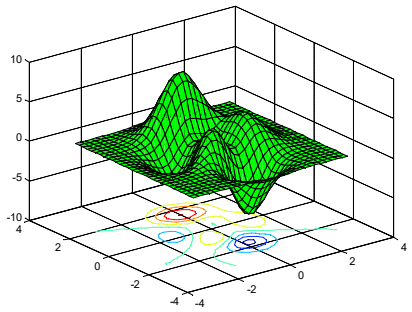
Exemple
Afficher une parcelle de surface et des courbes au niveau des pics de surface.

```
 [X,Y,Z] = peaks(30);
surfc(X,Y,Z)
colormap hsv
axis([-3 3 -3 3 -10 5])
```



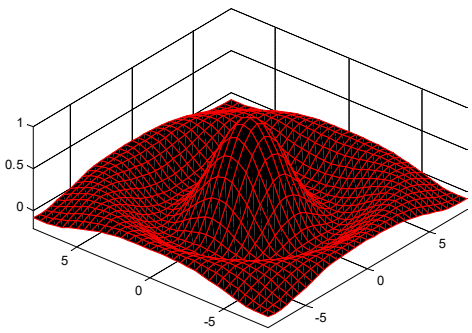
Exemple

```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z, 'FaceColor','green','EdgeColor','black')
```



Exemple

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z)
surf(X,Y,Z,'FaceColor','Black','EdgeColor','red','FaceLighting','phong')
daspect([5 5 1])
```



```
axis tight
view(-50,30)
```

Exemple