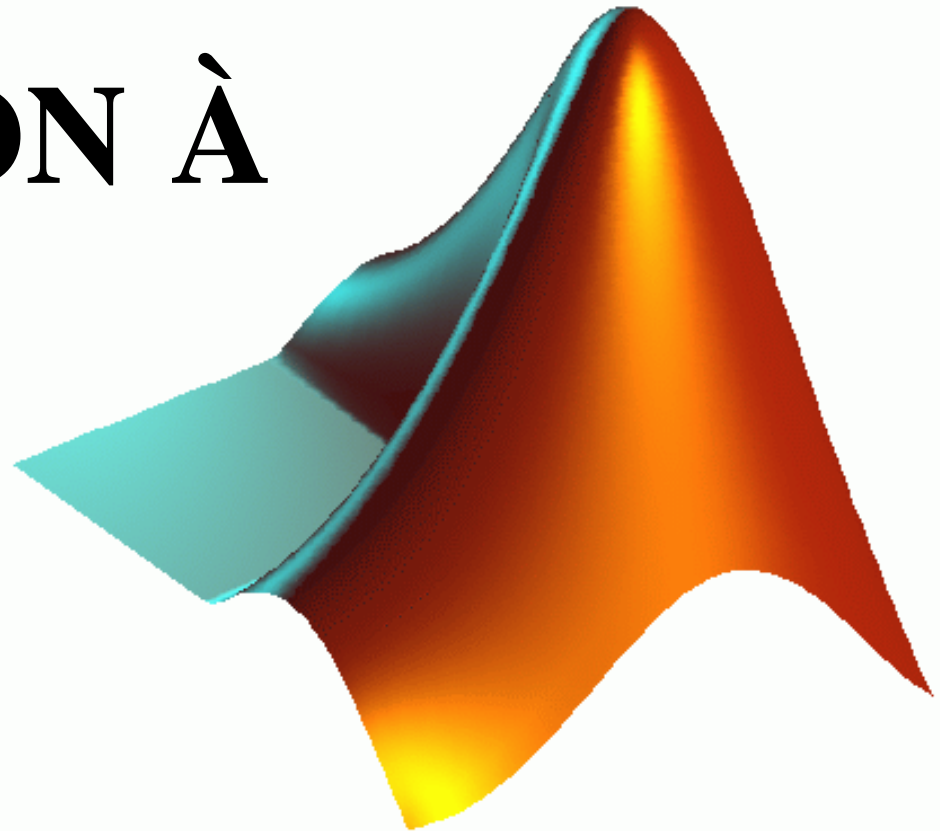
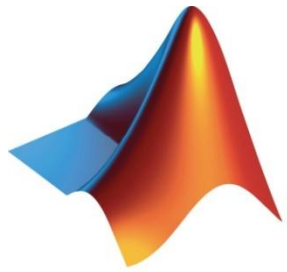


INITIATION À MATLAB





MathWorks®

Accelerating the pace of engineering and science

Matlab, ses boîtes à outils et Simulink sont des produits développés par la société The MathWorks, Inc. Matlab® et Simulink® sont des marques déposés par cette même société.

La distribution utilisée en séance est sous licence *classroom*, c'est-à-dire qu'elle est réservée à un usage académique éducatif. Toute utilisation à but commercial ou recherche est interdite.

Interface de MATLAB

The screenshot displays the MATLAB 7.12.0 (R2011a) interface. The main window is titled "MATLAB 7.12.0 (R2011a)" and has a menu bar with "File", "Edit", "Debug", "Desktop", "Window", and "Help". The "Current Folder" is set to "D:\Données Yassine\yassine.ariba@icam.fr\Matlab".

The interface is divided into several panels:

- Current Folder:** Shows a list of files and folders in the current directory, including "carte NI USB-6008", "distributed delay", "Documentation", "Manuel", "Matlab et C", "Mémo Simulink", "prepa tp bacdo", "prepa tp i47", "prepa tp ts", "slprj", "toolboxes", and various example files like "exemple_fft.asv", "exemple_fft.m", "exemple_fft2.asv", "exemple_fft2.m", "exemple_sos.asv", "exemple_sos.m", "lieu-nyquist-exemple-stabilite.fig", "mafonction.asv", "mafonction.m", "monequadiff.asv", "monequadiff.m", "monfichier.asv", "monfichier.m", "startup.m", "test.asv", "test.m", "testaerotherme.mdl", and "untitled.mdl.autosave".
- Command Window:** Contains the MATLAB command prompt with the following input:

```
>>  
>>  
fx >> |
```
- Workspace:** Displays a table of variables in the workspace:

Name	Value	Min
A	5	5
G	<1x1 tf>	
b	[1,2,3;4,5,6]	1
k	1.6000	1.6000
tau	10	10
taud	10	10
- Command History:** Shows the execution history of commands, including:

```
h/3*(sin(0)+4*sin(pi/4)+sin(p  
1/0.004  
%-- 16/11/2011 17:41 --%  
%-- 18/11/2011 17:39 --%  
A=5  
b=[1 2 3;4 5 6]  
G=tf(k,[tau 1])  
k=1.6  
taud=10  
G=tf(k,[tau 1])  
tau=10  
G=tf(k,[tau 1])  
clc
```

Variables et Matrices

```
Command Window
>> A=[3 2 -2; -1 0 1; 1 1 0]
A =
     3     2    -2
    -1     0     1
     1     1     0
>> A=[
3 2 -2
-1 0 1
1 1 0]
A =
     3     2    -2
    -1     0     1
     1     1     0
>> B=[1 2; 3 4];
>> B
B =
     1     2
     3     4
>> a
??? Undefined function or variable 'a'.
```

Deux façons de créer une matrice

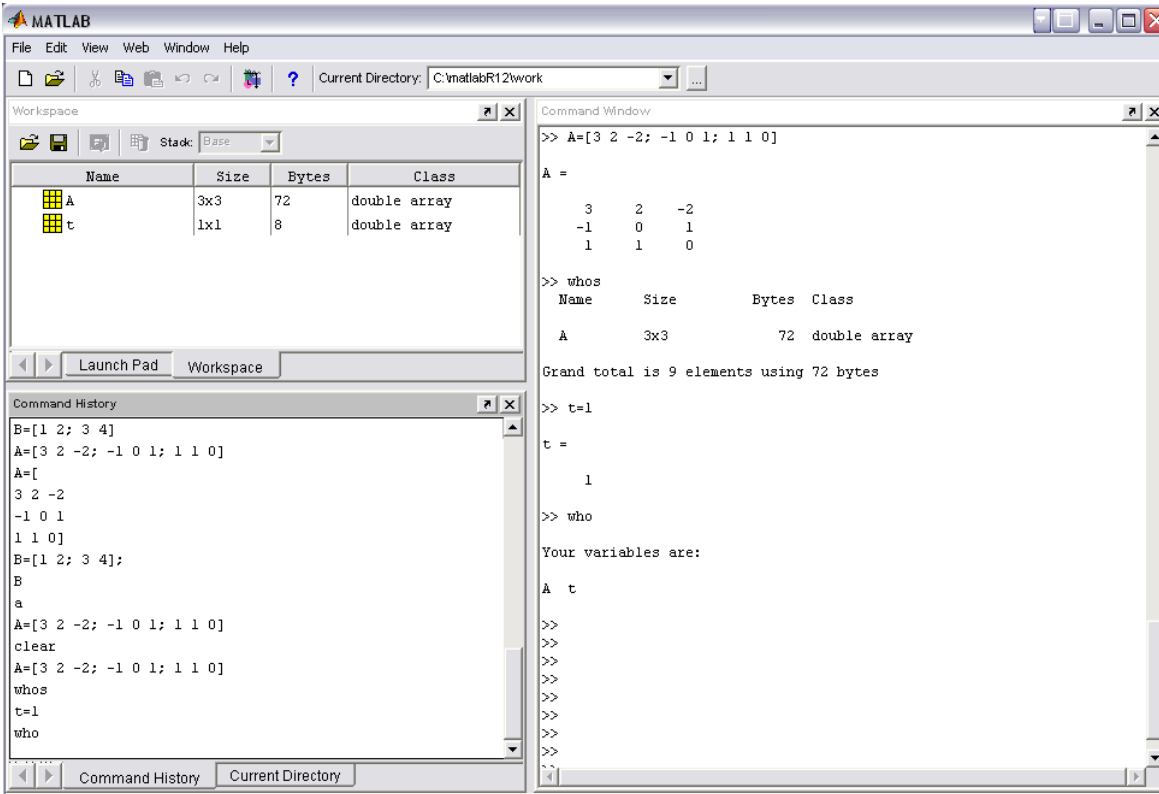
Sans écho à l'écran

MATLAB fait la différence entre minuscules et majuscules

Fenêtre de commandes : création de quelques variables

Variables et Matrices

Information sur les variables



The screenshot shows the MATLAB environment with the following components:

- Workspace:** A table listing variables in the current workspace.

Name	Size	Bytes	Class
A	3x3	72	double array
t	1x1	8	double array

- Command Window:** Shows the execution of several commands and their outputs.

```
>> A=[3 2 -2; -1 0 1; 1 1 0]
A =
     3     2    -2
    -1     0     1
     1     1     0

>> whos
  Name      Size      Bytes  Class
  ----      -
  A         3x3         72  double array

Grand total is 9 elements using 72 bytes

>> t=1
t =
     1

>> whos
Your variables are:
  A      t
  ----  -
  3x3    1
  double double array

>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
```
- Command History:** A scrollable list of previously entered commands, including the creation of variables B, A, and t, and the use of 'clear' and 'whos' commands.

```
1 >> who
2 Your variables are:
3 A t
4 >> clear t
5 >> who
6 Your variables are:
7 A
```

clear all **réinitialise l'environnement**

Variables et Matrices

Déclaration d'un vecteur

```
1 % Vecteur ligne  
2 >> m = [0 1 2 3]
```

```
m =  
    0 1 2 3
```

ou

```
1 >> m = [0:3]
```

La progression entre deux éléments peut être précisée :

```
>> x = [1:-0.25:-1]
```

Déclaration d'un vecteur colonne

```
1 >> n = [0;1;2;3] % Création d'un vecteur colonne
```

ou par transposition

```
1 >> n = m'
```

Variables et Matrices

Déclaration d'une matrice

```
1 >> A=[1 2 3;4 5 6;7 8 9]
```

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

Taille d'une matrice

```
1 >> size(A)
```

```
ans =  
     3     3
```

Accès aux éléments

```
1 >> A(1,2)  
2 >> A([1 3],:)
```

Matrices particulières

```
1 >> eye(n)  
2 >> ones(n,m)  
3 >> zeros(n,m)  
4 >> rand(n,m)
```

n, m sont des entiers

Variables et Matrices

Opérations élémentaires

```
1 >> 3+5
2 >> ans-2
3 >> A + ones(3)
4 >> m + x(1:4)
```

**Dans toutes opérations, il faut faire attention
aux dimensions des opérandes**

Multiplication

```
1 >> 3*4
2 >> b=[1;0;5]
3 >> A*b
4 >> b'*A
5 >> [1,1,2]*b
```

Division

```
1 >> 3/4
```

```
1 >> 3\4
```

```
1 >> B=[1 4 3;2 -1 0;0 3 3]
2 >> y=B\b
```

Élévation à la puissance

```
1 >> 2^3
2 >> sqrt(16)
3 >> A^2
4 >> B^(-1) % idem fonction d'inversion inv(B)
```


Variables et Matrices

Opérations terme à terme

```
1 >> A .^ 2
2 >> diag([1,2,3]) .* A
3 >> c=[1:1:5],d=[0.5:0.5:2.5]
4 >> c./d
5 >> e=[1 -1 0 2 2]
6 >> e.*c
```

Manipulations sur les matrices

Manipulation par blocs

```
1 >> C = [ones(3), rand(3,2);  
2         rand(2,3), eye(2)]
```

```
1.0000 1.0000 1.0000 0.4565 0.4447  
1.0000 1.0000 1.0000 0.0185 0.6154  
1.0000 1.0000 1.0000 0.8214 0.7919  
0.9218 0.1763 0.9355 1.0000 0  
0.7382 0.4057 0.9169 0 1.0000
```

Extraction de sous matrices

```
1 >> C1 = C(:, [3 5])  
2 >> C2 = C(:, [1:3])  
3 >> C3 = C([1 2], [4 5])  
4 >> D = diag(C)  
5 >> D1 = diag(C, 1)  
6 >> D2 = diag(C, -2)  
7 >> T1 = tril(C)  
8 >> T2 = triu(C)
```

C(arg1, arg2) : le premier argument permet de sélectionner un ensemble de lignes, le second un ensemble de colonnes.

Boucles et branchements

Boucles

```
1 for compteur = expression  
2     instructions  
3 end
```

```
1 n = 20;  
2 for k = 1:n  
3     y(k) = k^2;  
4 end
```

```
1 while expression  
2     instructions  
3 end
```

```
1 x = 16;  
2 while x > 1  
3     x = x / 2;  
4 end
```

Boucles et branchements

Conditions

```
1 if      expression 1  
2         instructions 1  
3 elseif expression 2  
4         instructions 2  
5 else  
6         instructions 3  
7 end
```

```
1 switch expression 1  
2         instructions 1  
3 case   expression 2  
4         instructions 2  
5 case   {expression 3, expression 4, ...}  
6         instructions  
7 otherwise  
8         instructions 4  
9 end
```

Opérateurs logiques

opération	Symbole	opération	Symbole
supérieur à	>	supérieur ou égal à	>=
inférieur à	<	inférieur ou égal à	<=
égal à	==	non	~
et	&, &&	ou	,

Il existe aussi xor, any, all.

Aide en ligne

```
1 >> help polyval
2 >> doc polyval
```

```
>> help polyval
```

POLYVAL Evaluate polynomial.

`Y = POLYVAL(P,X)` returns the value of a polynomial `P` evaluated at `X`. `P` is a vector of length `N+1` whose elements are the coefficients of the polynomial in descending powers.

$$Y = P(1)*X^N + P(2)*X^{(N-1)} + \dots + P(N)*X + P(N+1)$$

If `X` is a matrix or vector, the polynomial is evaluated at all points in `X`. See `POLYVALM` for evaluation in a matrix sense.

The screenshot shows the MATLAB Help window with the following content:

- Contents:** MATLAB > Functions > Mathematics > Polynomials > polyval
- polyval**
Polynomial evaluation
- Syntax**

```
y = polyval(p,x)
[y,delta] = polyval(p,x,S)
y = polyval(p,x,[],mu)
[y,delta] = polyval(p,x,S,mu)
```
- Description**

`y = polyval(p,x)` returns the value of a polynomial of degree `n` evaluated at `x`. The input argument `p` is a vector of length `n+1` whose elements are the coefficients in descending powers of the polynomial to be evaluated.

$$y = p_1x^n + p_2x^{n-1} + \dots + p_nx + p_{n+1}$$

`x` can be a matrix or a vector. In either case, `polyval` evaluates `p` at each element of `x`.

`[y,delta] = polyval(p,x,S)` uses the optional output structure `S` generated by `polyfit` to generate error estimates `delta`. `delta` is an estimate of the standard deviation of the error in predicting a future observation at `x` by `p(x)`. If the coefficients in `p` are least squares estimates computed by `polyfit`, and the errors in the data input to `polyfit` are independent, normal, and have constant variance, then `ydelta` contains at least 50% of the predictions of future observations at `x`.

`y = polyval(p,x,[],mu)` or `[y,delta] = polyval(p,x,S,mu)` use `mu` in place of `x`. In this equation, `mu` = `mean(x)` and `mu` = `std(x)`. The

`[Y,DELTA] = POLYVAL(P,X,S)` uses the optional output structure `S` created by `polyfit` to generate prediction error estimates `DELTA`. `DELTA` is an estimate of the standard deviation of the error in predicting a future observation `y` by `P(X)`.

If the coefficients in `P` are least squares estimates computed by `polyfit`, and the errors in the data input to `POLYFIT` are independent, normal, and have constant variance, then `Y +/- DELTA` will contain at least 50% of the observations at `X`.

`[Y,DELTA] = POLYVAL(P,X,S,MU)` uses `XHAT = mu` in place of `X`. The centering and scaling parameters `MU` are computed by `POLYFIT`.

Polynomial `p(x) = 3x^2+2x+1` at `x = 5,7, and 9`:

```
polyval(p,x,S,mu)
```

Fonctions usuelles

Fonctions scalaires

Fonctions usuelles

```
1 sin      exp      abs      round
2 cos      log      sqrt     tanh
3 tan      rem      sign     acosh   ...
```

```
1 >> help elfun
```

Fonctions matricielles

Fonctions usuelles

```
1 max      sum      mean     sort
2 min      prod     ...
```

Fonctions matricielles

```
1 eig      size     norm
2 inv      ...
```

```
1 >> help matfun
```

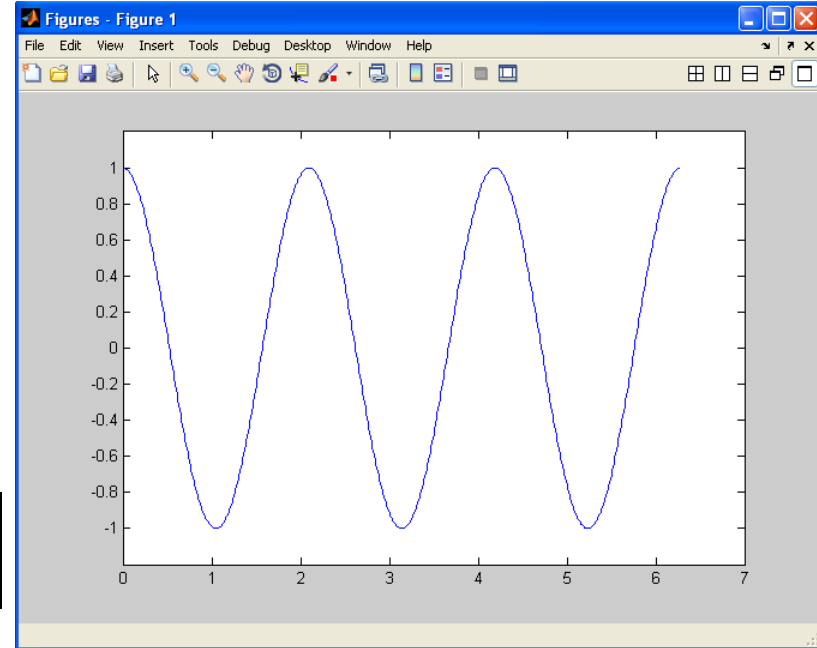
Représentation graphique

Commande de base

```
1 >> x = [0 : 0.01 : 2*pi];  
2 >> y = cos(3*x);  
3 >> plot(x,y)
```

Quelques options

```
1 plot(x,y,s) % s:chaîne de caractères  
2           pour la couleur et le tracé
```



Couleurs	Marqueurs	Tracés
y : jaune	+ : plus	- : trait continu
m : magenta	o : cercles	:: : pointillés
c : cyan	* : étoiles	-. : trait point
r : rouge	x : croix	- : tirets
g : vert	s : carré	
b : bleu	d : diamant	
k : noir	v^<> : triangles ▽ △ ◁ ▷	
w : blanc	h : hexagramme	

```
1 >> plot(x,y,'r*')
```


Représentation graphique

Superposition de tracés

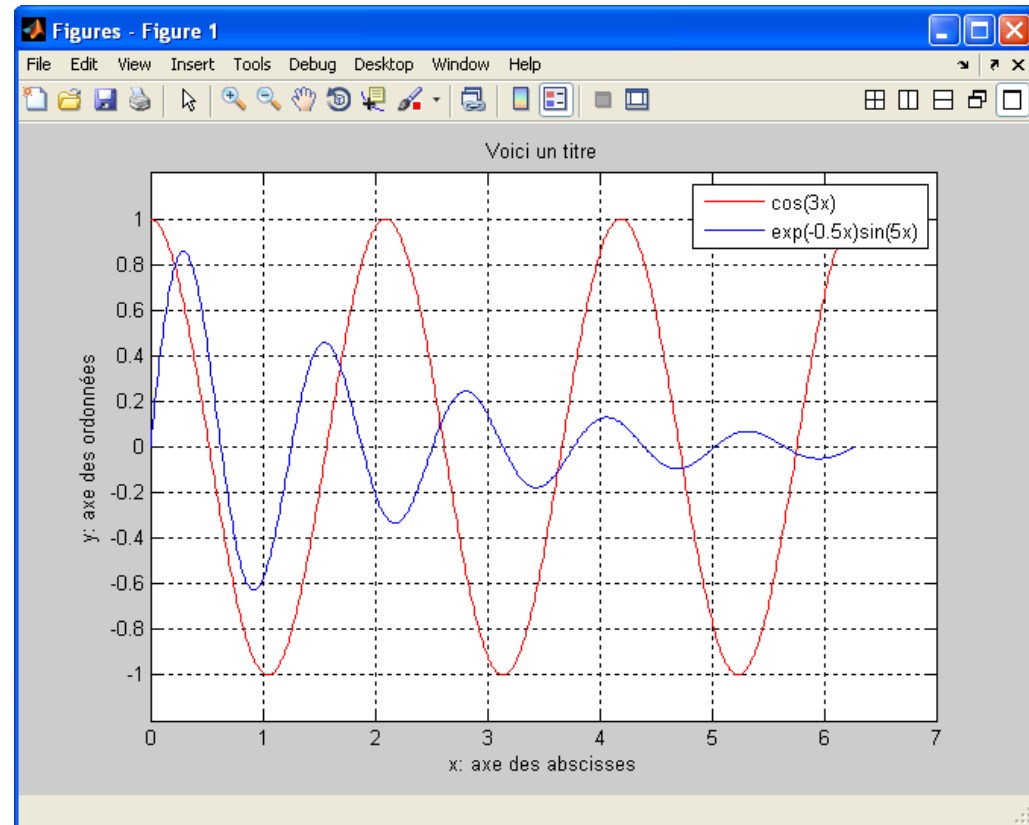
```
1 >> x = [0 : 0.01 : 2*pi];  
2 >> y = cos(3*x);  
3 >> z = exp(-0.5*x).*sin(5*x),  
4 >> plot(x,y,x,z)
```

ou

```
1 >> hold on  
2 >> plot(x,y);  
3 >> plot(x,z)
```

Pour compléter le tracé

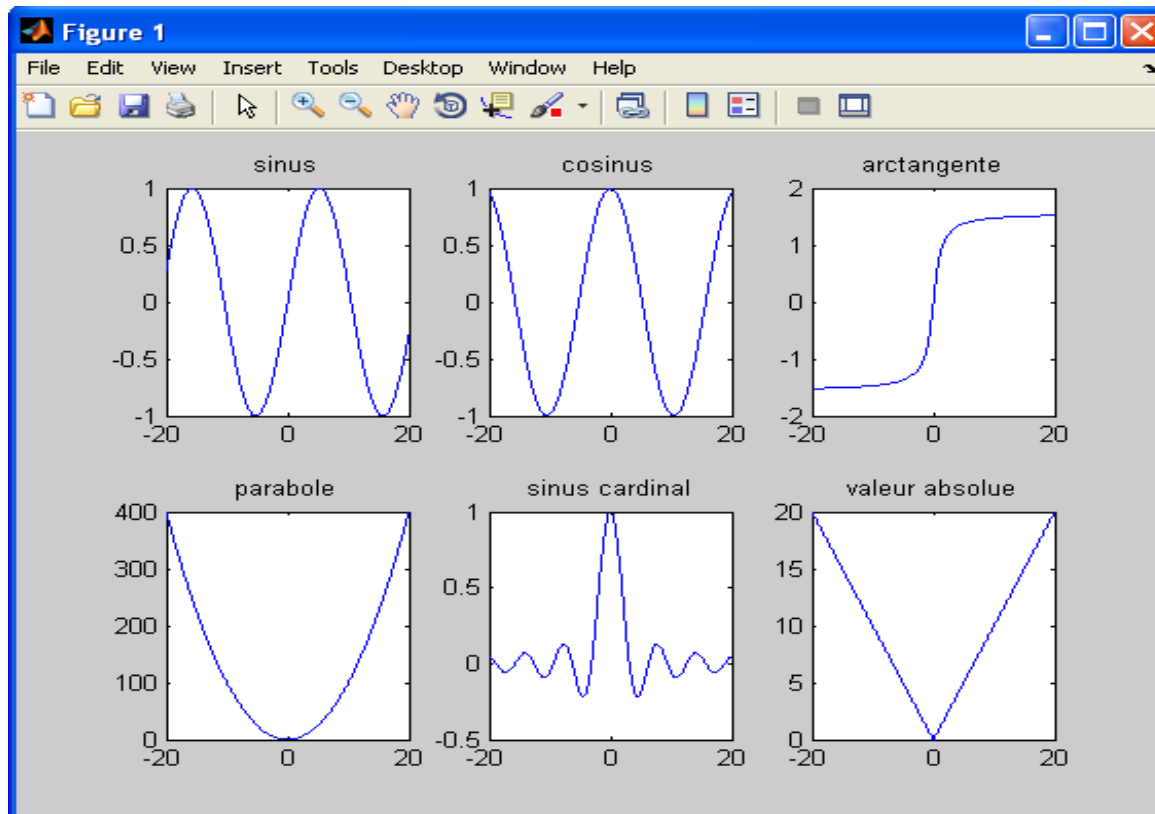
```
1 >> legend('cos(3x)', 'sin(2x)')  
2 >> title('Voici un titre')  
3 >> axis([0 7 -1.2 1.2])  
4 >> xlabel('x: axe des abscisses')  
5 >> ylabel('y: axe des ordonnées')  
6 >> grid on
```



Représentation graphique

Figure avec plusieurs graphiques

```
1 >> x=[-20:0.05:20];  
2 >> subplot(2,3,1) ; plot(x,sin(0.3*x)); title('sinus');  
3 >> subplot(2,3,2) ; plot(x,cos(0.3*x)); title('cosinus');  
4 >> subplot(2,3,3) ; plot(x,atan(x)); title('arctangente');  
5 >> subplot(2,3,4) ; plot(x,x.^2); title('parabole');  
6 >> subplot(2,3,5) ; plot(x,sin(x)./x); title('sinus cardinal');  
7 >> subplot(2,3,6) ; plot(x,abs(x)); title('valeur absolue');
```



Utilisation de m-files

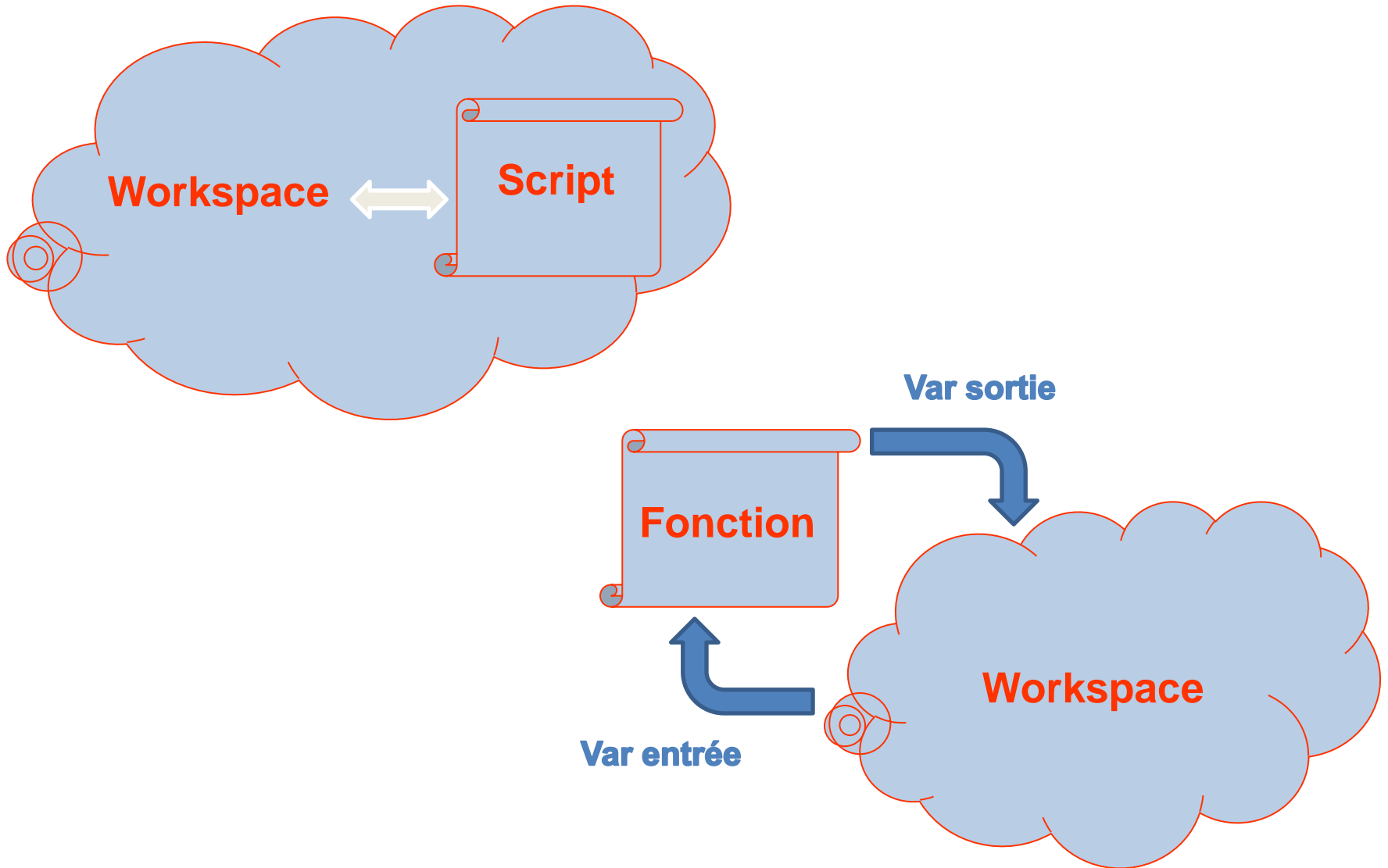
Scripts

- Liste d'instructions.
- Accès à toutes les variables de l'environnement.
- Pas de paramètres d'entrée.
- Ne retourne aucune valeurs.
- Appels d'autres scripts ou fonctions.

Fonctions

- N'a pas accès aux variables de l'environnement.
- Possède des variables d'appel en arguments.
- Variables locales inaccessibles depuis l'extérieur.
- Retourne une ou plusieurs valeurs.

Utilisation de m-files



Utilisation de m-files

Script1.m

```
1 x1=-1 ; x2=1;
2 n=50;
3 x = linspace(x1,x2,n);
4 y = exp(-2*x).*(x.^2-3*x+4);
5 plot(x,y);
6 disp('voir tracé sur figure')
```

```
>> n=0;
>> script1;
??? Undefined function or
    variable 'script1'.
>> Script1;
voir tracé sur figure

>> n
n =
    50
```

Fonc1.m

```
1 function [moy,ecarttype] = Fonc1(x)
2 % Commentaire affiché par le help Fonc1
3     n = length(x);
4     moy = sum(x) / n;
5     ecarttype = sqrt(sum((x - moy).^2)/n);
6
```

```
>> Fonc1;
??? Input argument »x » is undefined
Error in ==> Fonc1 at 3
>>[a,b] = Fonc1([1 2 3 4]);
a =
    2.5000
b =
    1.1180
>> n
??? Undefined function or variable 'n'.
```

Exercices d'application

Énoncé 1

Les équations paramétriques d'une ellipse centrée à l'origine sont:

$$x = A \cos t$$

$$y = B \sin t$$

avec $0 \leq t \leq 2\pi$

1. Tracer cette ellipse pour $A=2$ et $B=1$.
2. Ajouter un titre et des labels abscisses/ordonnées
3. Tracer la courbe pour différentes valeurs de A et B . A quoi correspondent ces 2 paramètres?

Exercices d'application

Solution 1

Exercices d'application

Énoncé 2

Calculer la somme des carrés des 1000 premiers entiers de 3 façons différentes:

- Avec l'instruction for
- Avec l'instruction while
- Avec l'instruction sum

Exercices d'application

Solution 2

Exercices d'application

Énoncé 3

Représenter graphiquement la fonction suivant:

$$y = \frac{1}{\varepsilon} \left(\phi\left(\frac{x-a}{\varepsilon}\right) - \phi\left(\frac{x-b}{\varepsilon}\right) \right)$$

$$\text{avec } \phi(x) = e^{-1-x^2}$$

a, b et ε sont des paramètres constants.

La fenêtre de représentation porte sur $-2 \leq x \leq 3$ et $-1 \leq y \leq 1$

consigne: utiliser une fonction (m-file) pour calculer $\Phi(x)$.

A.N. : On prendra $a=-1$, $b=2$ et $\varepsilon=0.5$.

Exercices d'application

Solution 3

Exercices d'application

Énoncé 4

Recherche d'une racine par dichotomie.

Soit $f(x)$ une fonction continue strictement croissante sur $[a,b]$ telle que:

$$f(a) < 0 \quad \text{et} \quad f(b) > 0$$

L'objectif est de trouver x_0 tel que $f(x_0) = 0$.

- évaluer la fonction en $c=(a+b)/2$;
- si $f(c) < 0$, l'intervalle de recherche devient $[c,b]$;
- si $f(c) > 0$, l'intervalle de recherche devient $[a,c]$.
- Ce processus est ensuite réitéré...

A.N. : Déterminer la racine des fonctions:

- $f(x) = 2x^4 + 2.3x^3 - 16x^2 - 8x - 17.5$ sur l'intervalle $[0,100]$.
- $g(x) = \tan(x^2) - x$ sur l'intervalle $[0.5,\pi/3]$.

Exercices d'application

Solution 4

Exercices d'application

Énoncé 5

Soit un signal créneau $f(t)$ d'amplitude A , de période T et de valeur moyenne nulle. Sa décomposition en série de Fourier est donnée par:

$$f(t) = \sum_{n=1}^{+\infty} a_n \cos(n\omega t) \quad \text{avec}$$

$$a_n = \frac{2A}{n\pi} \sin\left(n\frac{\pi}{2}\right)$$

$$\omega = \frac{2\pi}{T}$$

Représenter le signal $f(t)$ à partir de sa décomposition en série de Fourier jusqu'à l'ordre $n=7$. Tester ensuite pour des valeurs de n supérieures.

A.N. : On prendra $A=2$ et $T=0.5\text{s}$. Echelle temporelle: $0 \leq t \leq 2$ avec un pas de 0.001 .

Exercices d'application

Solution 5

Exercices d'application

Énoncé 6

Manipulation de polynômes. Créer deux fonctions qui permettent:

- d'additionner deux polynômes,
- de multiplier deux polynômes.

Nous définirons un polynôme de la forme (par ex.)

$$P(x) = x^4 + 2x^3 + 3x^2 + 4x + 5$$

par un vecteur ligne contenant ses coefficients: $P=[1 \ 2 \ 3 \ 4 \ 5]$

Notons la fonction `poly2str(P, 'x')`

Exercices d'application

Solution 6