Ministère de l'Education Nationale,
de l'Enseignement Supérieur, de la Formation
des Cadres et de la Recherche Scientifique

Fondation ACADEMIA

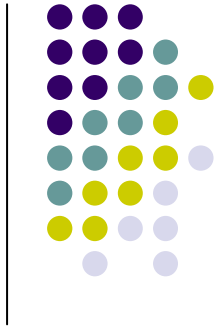
Les Journées Informatiques des Classes Préparatoires aux Grandes Ecoles
Rabat, 29 Octobre-2 Novembre 2009

Matlab©

une introduction

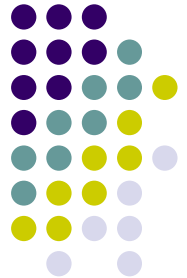
Said. EL HAJJI
PhD en Mathématiques Appliquées, Université Laval, Canada.
Professeur de l'enseignement supérieur
Faculté des Sciences de Rabat
http://www.fsr.ac.ma/mia/elhajji.htm
elhajji@fsr.ac.ma

MATLAB :



- **MAT** (rix) **LAB** (oratory) est un logiciel puissant doté à la fois d'un langage de programmation haut niveau et d'outils dédiés au calcul numérique et à la visualisation numérique.
- Développé en C par la société Mathworks (http://www.mathworks.com/).
- Matlab était initialement destiné à faire du calcul matriciel simplement.

Actuellement, Matlab recouvre d'autres domaines d'applications de l'informatique scientifique :



- **visualisation graphique 2D et 3D**
- **résolution d'équations aux dérivées partielles**
- **optimisation**
- **traitement du signal**
- **traitement de l'image**
- **logique floue**
- **réseaux de neurones**
- **…**

Les système Matlab se divise en deux parties :

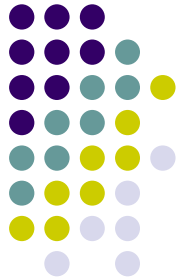
1) Le noyau

Il comprend :

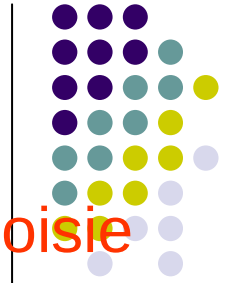
- **l'environnement de travail offrant plusieurs facilités pour la manipulation des données.**
- **son interpréteur permet de tester rapidement ses propres programmes Matlab.**
- **le système graphique Matlab (interfaces homme-machine, graphiques, images, animations).**
- **le langage de programmation Matlab.**
- **une librairie de fonctions mathématiques Matlab.**
- **un système d'interfaçage faciltant l'exécution de programmes C ou Fortran ou sous Matlab.**

2) Des Toolboxes (boîtes à outils)

Ils regroupent un ensemble de fonctions spécifiques à un thème.



INSTRUCTION DE BASE



>>8/10 **«Entrée»**

ans =

0.8000

variable temporaire choisie par Matlab

4 décimales par omission

>>r = 8/10 **«Entrée»**

r =

0.8000

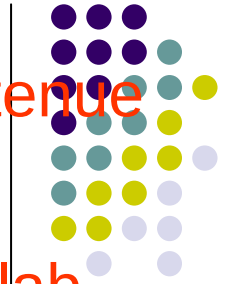
variable choisie par l'utilisateur

>>r **«Entrée»**

r =

0.8000

r conserve la dernière valeur calculée



valeur de r retenue

```
>>s=10*r          «Entrée»
s =
        8
```

fonction dans Matlab

```
>>v=sin(s)        «Entrée»
v =
        0.9894
```

; ne pas imprimer la réponse

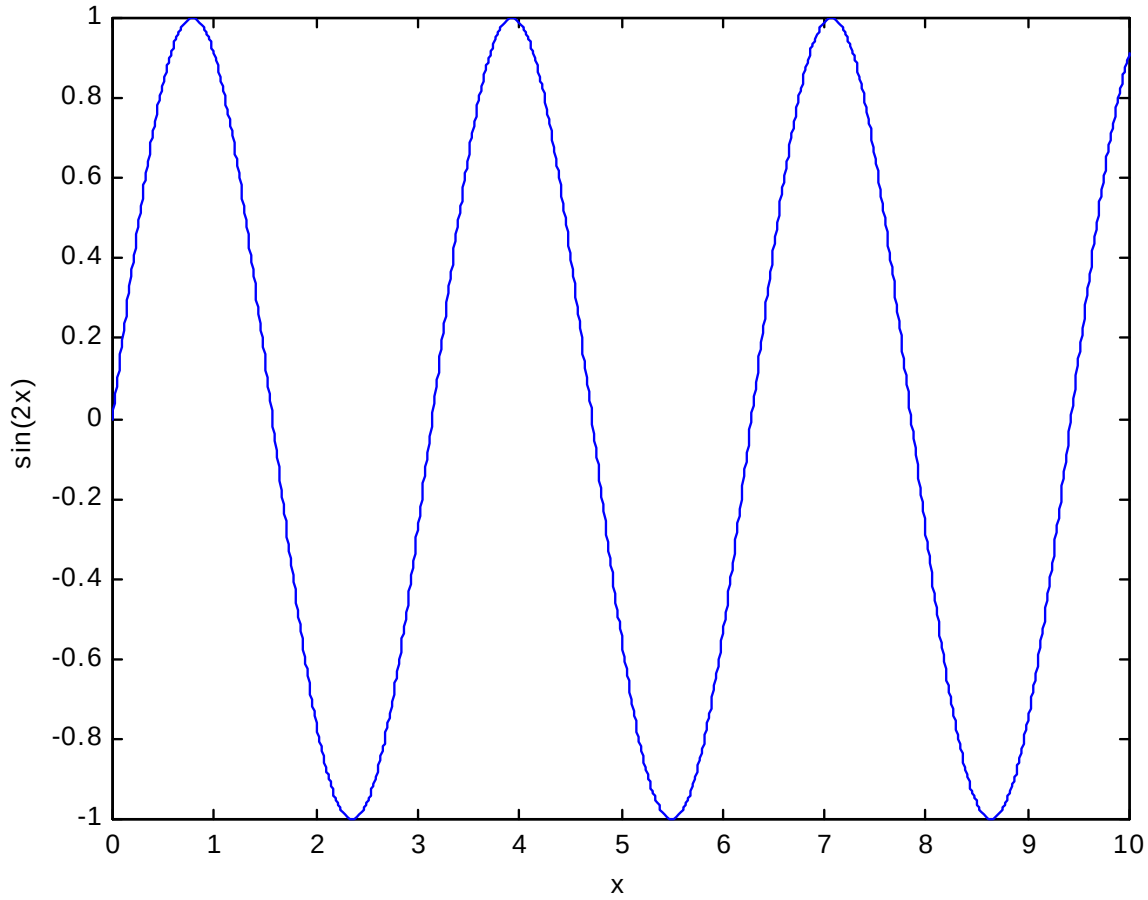
```
>>u=[0:.1:10];    «Entrée»
>>z=sin(u);
```

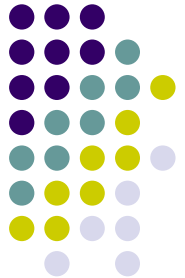
faire varier u de 0 à 10 par saut de 0.1

```
>>u(7)            «Entrée»
ans =
        0.6000
```

u est une variable indicée dont on veut la 7ième valeur

```
>>x=[0:.01:10];
>>y=sin(2*x);
>>plot(x,y),xlabel('x'),ylabel('sin(2x)')
```



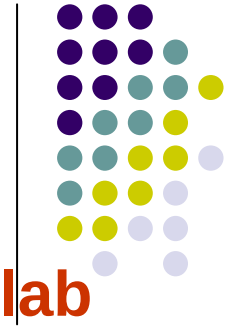


Opérations de base

Vecteurs

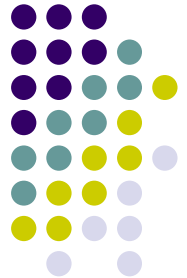
Matrices

Opérateurs arithmétiques :



Symbole	Opération	Forme Matlab
^	exponentiation : $\mathbf{a^b}$	a^b
*	multiplication : **ab**	a*b
/	div. vers la droite : **a/b**	a/b
\	div. vers la gauche : a/b	b\a
+	addition : **a + b**	a+b
-	soustraction : **a – b**	a-b

Variables spéciales :



ans	variable temporaire contenant la réponse la plus récente
eps	spécifie la précision d'un nombre en point flottant
i, j	le nombre imaginaire $(-1)^{1/2}$
Inf	l'infini
NaN	indique un résultat numérique non défini
pi	le nombre π

Commandes pour la gestion d'une session :

clc	nettoie la fenêtre dite Command
clear	enlève les variables de la mémoire
exist ('*nom*')	détermine si un fichier existant ou une variable a le nom '*nom*'
help *nom*	recherche en ligne pour le sujet *nom*
lookfor *nom*	recherche l'aide pour le mot-clé *nom*
quit	arrête Matlab
who	énumère les variables courantes en mémoire
whos	énumère les variables actuelles et leur dimension

Création de vecteurs et de matrices

Vecteurs :

```
>>r = [2,4,10]                    «Entrée»

r =

     2.0000    4.0000   10.0000

>>s = [2 4 10]                     «Entrée»

s =

     2.0000    4.0000   10.0000
```

Crochets carrés

Séparateurs

vecteurs lignes

Vecteurs (suite)

Vecteurs colonnes

```
>>g = [3;7;9]
g =
     3
     7
     9
```

«Entrée»

Séparateur pour les lignes

ou

```
>>g = [3,7,9]´
g =
     3
     7
     9
```

«Entrée»

Transposée

Création d'une matrice :

```
>>A = [2,4,10;16,3,7]
A =
     2   4   10
    16   3    7
```

«Entrée»

Séparateurs

```
>>a = [1,3,5];
```

«Entrée»

```
>>b = [7,9,11];
```

«Entrée»

vecteur

```
>>c = [a b]
c =
     1   3   5   7   9   11
```

«Entrée»

2 vecteurs séparés par , ou un espace

```
>>D = [a;b]
D =
     1   3    5
     7   9   11
```

«Entrée»

2 vecteurs lignes avec séparateur de lignes

Transposée d'une matrice :

Matrice transposée : matrice dont on remplace les lignes par les colonnes

$$\mathbf{A} = \begin{bmatrix} -2 & 6 \\ -3 & 5 \end{bmatrix}$$

$$\mathbf{A}^{\mathbf{T}} = \begin{bmatrix} -2 & -3 \\ 6 & 5 \end{bmatrix}$$

>>A= [-2,6;-3,5]' ← **apostrophe**

A =

-2	**-3**
6	**5**

Jeu des indices :

v(:) tous les éléments du vecteur v

v(2:5) du 2e au 5e élément

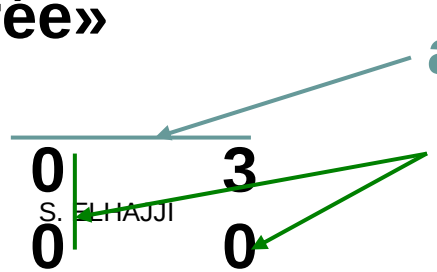
A(:, 3) tous les éléments de la 3e colonne

A(:,2:5) tous les éléments de la 2e à la 5e colonne

A(2:3,1:3) tous les éléments dans la 2e et 3e ligne et qui sont aussi dans la 1ère à la 3e colonne

A([1 4]) = [ ] élimine les colonnes 1 et 4

```
>>A= [6, 9, 4; 1, 5, 7];
>>A(1,5) = 3          «Entrée»
A =
     6     9     4     0     3
     1     5     7     0     0
```



Commandes pour les vecteurs et matrices :

find(x) — **Fournir une variable indicée contenant les indices des éléments non nuls de x**

[u, v, w] = find (A) — **Fournir des variables indicées contenant les indices des lignes et colonnes des éléments non nuls de la matrice A**

length(A) — **Fournir le nombre d'éléments de A si A est un vecteur ou la valeur max de *m* ou *n* si A est une matrice *m x n***

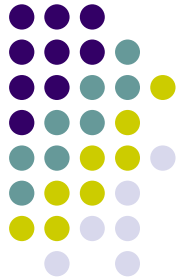
max(A) — **Fournir l'élément ayant la valeur algébrique max si A est un vecteur ou un vecteur contenant l'élément max dans chaque colonne de la matrice A**

[x, k] = max(A) — **Même définition que max(A) sauf que le stockage des valeurs max s'effectue dans le vecteur ligne x et leurs indices dans le vecteur ligne k**

... suite à la diapositive suivante

Commandes pour les vecteurs et matrices :

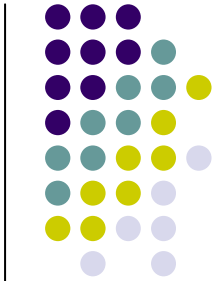
(... suite)



min(A) **[x, k] = min(A)**	**Même signification sauf qu'il s'agit des *valeurs min***
size(A)	**Fournir un vecteur ligne contenant les dimensions *m x n* de la variable indicée A**
sort(A)	**Classer chaque colonne de la variable indicée A par *ordre croissant* des grandeurs et fournir une variable indicée ayant les mêmes dimensions que A**
sum(A)	**Effectuer la *somme* des éléments de chaque colonne de A et fournir le résultat dans un vecteur ligne**

(Fin)

Opérations élément par élément :



Définitions à l'aide d'un exemple :

```
>>A = [6 , 3];
>>B = [4 , 8];
>>c = 2;
>>d = 5
```

addition d'un scalaire

```
>>A + c
ans
        8       5
```

soustraction d'un scalaire

```
>>A – d
ans
        1       -2
```


(... suite à la diapositive suivante)

(... suite)

>>A + B ***addition matricielle***

ans

10 11

>>A – B ***soustraction matricielle***

ans

2 -5

>>A .* B

ans

24 24

multiplication élément par élément

>>A ./ B

ans

6/4 3/8

division par la droite

(... suite à la diapositive suivante)

(... suite)

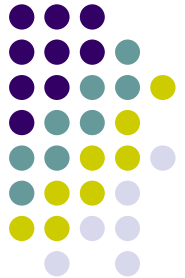
```
>>A .\ B
ans
        0.6667        2.6667

>>A .^ c
ans
        36     9
```

division par la gauche

exponentiation

(Fin)

Opérations matricielles (multiplication) :



Posons que

A : matrice m x p

B : matrice p x n

Produit matriciel **C** = **AB** $\Rightarrow$ **C =** matrice m x n

IMPORTANT :

Nombre de colonnes de A = nombre de lignes de B

(... suite à la diapositive suivante)

(... suite)

Multiplication de 2 vecteurs :

```
>>u = [1, 2];
>>v = [-2, -1];

>>w = u * v'
w =
        -4
```

1 x n par n x 1

vecteur colonne

vecteur ligne

scalaire

produit scalaire
ou produit intérieur

(... suite à la diapositive suivante)

(... suite)

```
>>y = u' * v
y
    -2    -1
    -4    -2
```

produit extérieur

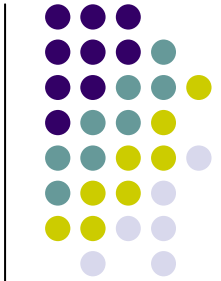
vecteur ligne 1 x n

vecteur colonne n x 1

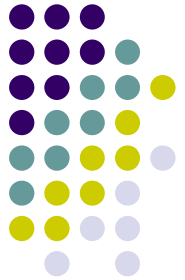
matrice n x n

(Fin)

Création de matrices spéciales :



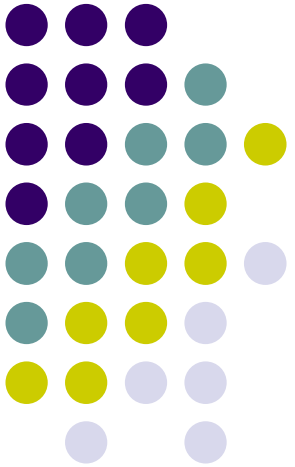
Commande	Description
eye(n)	Créer une matrice identité I *n x n*
eye(size(n))	Créer une matrice I de même dimension que A
ones(n)	Créer une matrice *n x n* remplie de 1
ones(m,n)	Créer une matrice *m x n* remplie de 1
ones(size(A))	Créer une matrice remplie de 1 et de même dimension que A
zeros(n)	Créer une matrice *n x n* remplie de 0
zeros(m,n)	Même signification qu'avec ones(m,n)
zeros(size(A))	et ones(size(A)) sauf 1 remplacé par 0



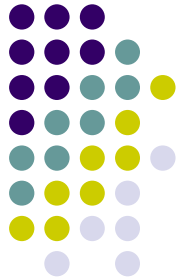
Programmer sous MATLAB

Programmer sous MATLAB

- Scripts et fonctions.
- Opérateurs de comparaison .
- Opérateurs logiques .
- Instructions de contrôle .

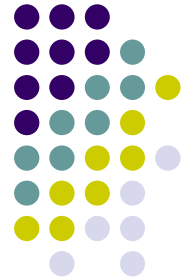


Scripts et fonctions :



- Un ***script*** est un ensemble d'instruction MATLAB qui joue le rôle de programme principal. Si le *script* est écrit dans le fichier de *nom* ***nom.m*** on l'exécute dans la fenêtre MATLAB en tapant après « **>>** » nom .

Fonctions



function [vars1 ,…,varsm] = fonc(vare1,…varen)

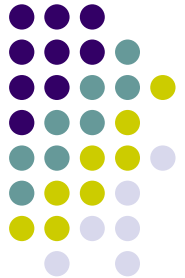
Séquence d'instructions

Où : *vars1 ,…,varsm* sont les variables de sortie de la fonction

vare1,…varen sont les variables d'entrée de la fonction

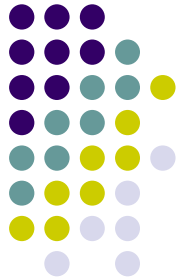
Séquence d'instructions est le corps de la fonction.

Scripts et fonctions



- Il est impératif que la fonction ayant pour nom **fonc** soit enregistrée dans un fichier de nom fonc.m sans quoi cette fonction ne sera pas « visible » par MATLAB.

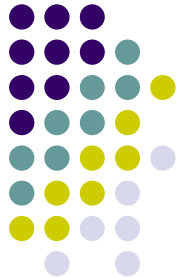
Opérateurs de comparaison



- Les opérateurs de comparaison sont :

1. = = : égal à (x= =y)
2. > : stictement plus grand que (x>y)
3. < : strictement plus petit que (x<y)
4. >= : plus grand ou égal à (x>=y)
5. <= : plus petit ou égal à (x<=y)
6. ~ = : différent de (x~ =y)

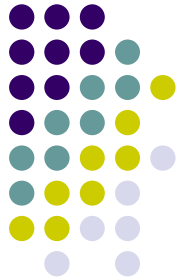
Opérateurs logiques



- Les opérateurs logiques sont :

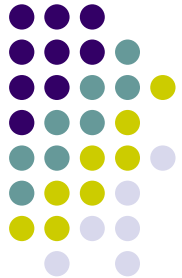
1. & : et (x & y)
2. | : ou (x | y)
3. ~ : non (~ x)

Instructions de contrôle



- **Boucle *FOR* (parcours d'un intervalle)**
- **Boucle *WHILE* (tant que…faire)**
- **L'instruction conditionnée IF**
- **Choix ventilé , l'instruction switch**

Boucle *FOR*



- **Syntaxe :**

  for indice = borne_inf : borne_sup

  Séquence d'instructions

  end

Où

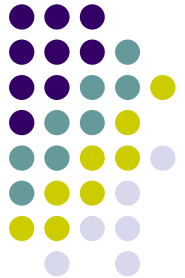
indice est une variable appelée l'indice de la boucle

Borne_inf et borne_sup sont deux constantes

On peut utiliser un incrément (pas) autre que 1.

La syntaxe est alors Borne_inf : pas : borne_sup

Boucle *WHILE*



- ***Syntaxe :***

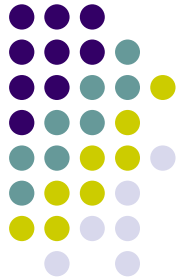
  while expression logique

  Séquence d'instructions

  end

- *expression logique* est une expression dont le résultat peut être vrai ou faux
- *séquence d'instructions* est le traitement à effectuer tant que expression logique est vraie.

L’instruction conditionnée IF



- ***Syntaxe :***

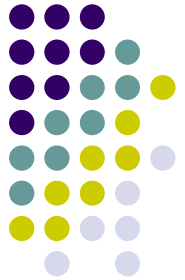
  if expression logique

  séquence d’instructions

  end

- *expression logique* est une expression dont le résultat peut être vrai ou faux
- **Il n’y a pas de mot clé « then »**

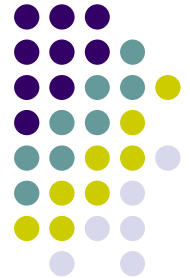
L’instruction conditionnée IF



- ***Syntaxe :***

```
if expression logique
        séquence d’instructions 1
else
        séquence d’instructions 2
end
```

L’instruction conditionnée IF

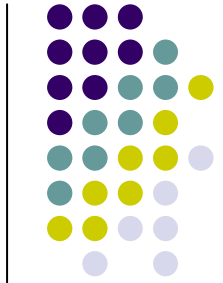


- Il est possible d’effectuer un choix en cascade :

Syntaxe :

```
if   expression logique 1
                séquence d’instructions 1
        elseif  expression logique 2
                séquence d’instructions 2
                …
        elseif   expression logique N
                séquence d’instructions N
        else  séquence d’instructions par défaut
end
```

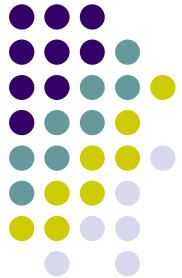
L’instruction switch



- ***Syntaxe :***

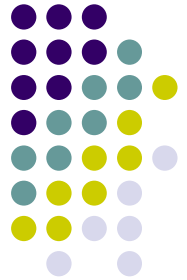
switch var
case cst1,
séquence d’instructions 1
case cst2,
séquence d’instructions 2
…
case cstN,
séquence d’instructions N
otherwise séquence d’instructions par défaut
end

L’instruction switch



- *var* est une variable numérique ou une variable chaîne de caractères
- *cst1,…,cstN*, sont des constantes numérique ou des constantes chaîne de caractères
- *séquence d’instructions i* est une séquence d’instructions à exécuter si le contenu de la variable *var* est égal à la constante *csti* (*var*= =*csti*).

L’instruction switch



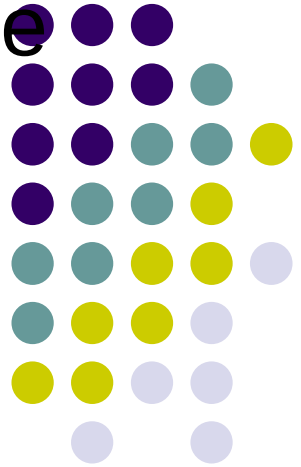
- *Il est possible de regrouper plusieurs « cas » si la séquence d’instructions à exécuter est la même pour ces différents cas. La syntaxe est alors :*

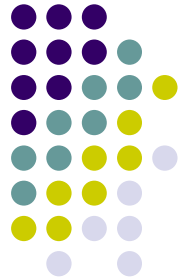
Case*{ cst1,…,cstN}*

Séquence d’instructions commune

Graphisme

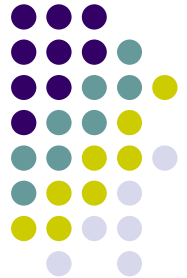
1. Gestion des fenêtres graphiques
2. Graphisme 2D
3. Améliorer la lisibilité d'une figure





Gestion des fenêtres graphiques

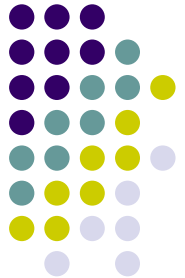
- Une instruction graphique ouvre une fenêtre dans laquelle est affiché le résultat de cette commande.
- Par défaut, une nouvelle instruction graphique sera affichée dans la même fenêtre et écrasera la figure précédente.
- On peut ouvrir une nouvelle fenêtre graphique par la commande *figure*.



Gestion des fenêtres graphiques

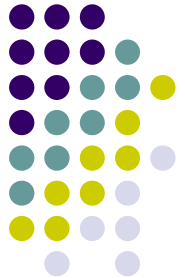
- Chaque fenêtre se voit affecter un numéro *n*.
- Ce numéro est visible dans le bandeau de la fenêtre sous forme d'un titre.
- Le résultat d'une instruction graphique est par défaut affiché dans la dernière fenêtre graphique ouverte
- On rend active une fenêtre graphique précédemment ouverte en exécutant la commande *figure(n)*

Gestion des fenêtres graphiques



- La commande *close* permet de fermer la fenêtre graphique active.
- On ferme une fenêtre graphique précédemment ouverte en exécutant la commande ***close(n)***
- Il est également possible de fermer toutes les fenêtres graphiques en tapant *close all*.

Graphisme 2D
la commande fplot



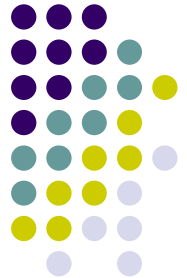
>>fplot('nomf', [xmin , xmax])

où :

nomf est le nom d'une fonction MATLAB incorporée, soit une expression définissant une fonction de la variable x, soit le nom d'une fonction utilisateur.

[xmin , xmax] est l'intervalle pour lequel est tracé le graphe de la fonction.

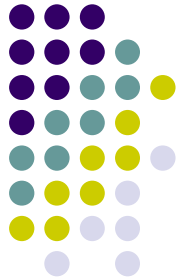
Graphisme 2D
la commande fplot



>>fplot('sin',[-2*pi 2*pi])

- Pour tracer le graphe de la fonction *h(x) = x sin(x)* on peut définir la fonction utilisateur *h* dans le fichier *h.m* de la manière suivante :

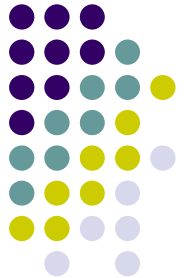
Graphisme 2D
la commande fplot



- *function y=h(x)*
- *y=x.*sin(x);*
- On obtient alors le graphe de la fonction *h* par l'instruction :

>>fplot('h',[-2*pi 2*pi])

Graphisme 2D
la commande fplot

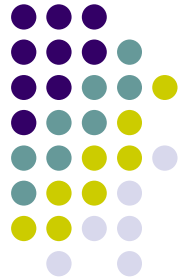


- L'autre façon de procéder est d'exécuter l'instruction :

>>fplot(' x* sin(x) ',[-2*pi 2*pi])

- Il est possible de tracer plusieurs fonctions sur la même figure:

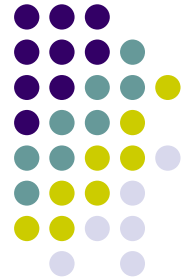
Graphisme 2D
la commande fplot



fplot('[nomf_1 , nomf_2 , nomf_3]', [xmin,xmax])

- *nomf_1 , nomf_2 , nomf_3* est le nom d'une fonction MATLAB incorporée, soit une expression définissant une fonction de la variable x, soit le nom d'une fonction utilisateur.

Graphisme 2D
la commande fplot



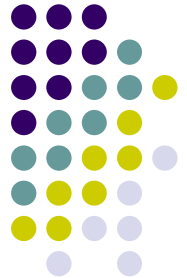
- **Pour limiter le graphe aux ordonnées comprises entre les valeurs *ymin* et *ymax* on passera comme second argument de la commande *fplot* le tableau *[xmin,xmax,ymin,ymax]*.**

- **Une autre possibilité pour gérer les bornes des valeurs en ordonnées est d'utiliser la commande *axis* après utilisation de la commande *fplot*.**

- **La syntaxe est axis([xmin, xmax, ymin, ymax]).**

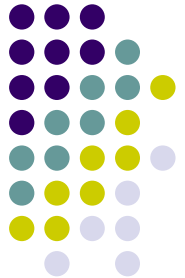
```
>> fplot('[sin(x)/x , cos(x)/x]', [-5, 5, -1, 1])
```

Graphisme 2D
la commande plot



- La commande *plot* permet de tracer un ensemble de points de coordonnées (*xi* , *yi*), i=1,…,N.
- La syntaxe est ***plot(x,y)*** où *x* est le vecteur contenant les valeurs *xi* en abscisse et *y* est le vecteur contenant les valeurs *yi* en ordonnée.

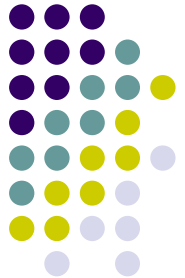
Graphisme 2D
la commande plot



- Les vecteurs *x* et *y* doivent être de même dimension mais il peut s'agir de vecteurs lignes ou colonnes.

- Par défaut, les points (*xi* , *yi*) sont reliés entre eux par des segments de droites.

Graphisme 2D
la commande plot

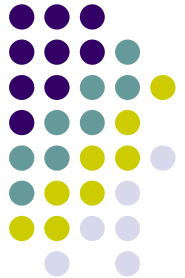


- Pour tracer le graphe de la fonction

$$h(x)=x \sin(x)$$

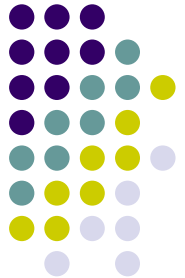
```
>> x=[-2*pi:0.01:2*pi]; y = x.*sin(x);
>> plot(x,y)
>> x=[-2*pi:1:2*pi]; y = x.*sin(x);
>> plot(x,y)
```

Graphisme 2D
la commande plot



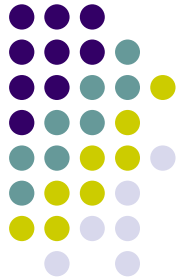
- On peut spécifier la couleur d'une courbe, le style de trait et/ou le symbole à chaque point *(xi , yi).*
- On donne un troisième paramètre d'entrée à la commande plot qui est une chaîne de 3 caractères de la forme *'cst'* :
  - ***c* désignant la *couleur* du trait**
  - ***s* le *symbole* du point**
  - ***t* le *type* de trait.**

Graphisme 2D
la commande plot



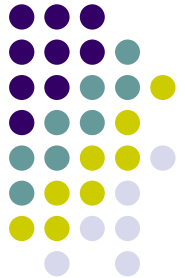
y -jaune	*.* point	- trait plein
m magenta	*o* cercle	*:* pointillé court
c cyan	*x* marque	x - pointillé long
r rouge	+ plus	-. pointillé mixte
g vert	* étoile	< triangle (gauche)
b bleu	*s* carré	> triangle (droit)
w blanc	*d* losange	*p* pentagone
k Noir	*v* triangle (bas)	^ triangle (haut)

Graphisme 2D
la commande plot



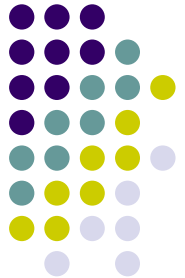
- Les valeurs par défaut sont *c* = *b*, *s* = . et *t* = - ce qui correspond à:
  - **Un trait plein**
  - **Bleu**
- Il n'est pas obligatoire de spécifier chacun des trois caractères
- La commande *grid* permet d'obtenir un quadrillage de la figure

Graphisme 2D
la commande **plot**



- Il est possible de tracer plusieurs courbes sur la même figure en spécifiant plusieurs tableaux *x1, y1, x2, y2, ...,* comme paramètres de la commande plot.
- Si l'on souhaite que les courbes aient une apparence différente, on utilisera des options de couleurs et/ou de styles de traits distincts après chaque couple de vecteurs *x, y.*

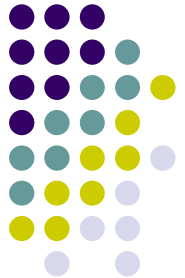
Graphisme 2D
la commande **plot**



- On trace sur l'intervalle *[-5, 5]* la fonction x^2 *cos(x)* en trait plein bleu et la fonction *x cos(x)* en trait pointillé rouge.

```
>> x = [-5:0.01:5];
>> y = x.^2.*cos(x); z = x.*cos(x);
>> plot(x,y,'b-',x,z,'r:');
```

Graphisme 2D
la commande loglog



- **la commande *loglog(x,y)* permet d'afficher le vecteur *log(x)* contre le vecteur *log(y).***
- **La commande *loglog* s'utilise de la même manière que la commande *plot*.**

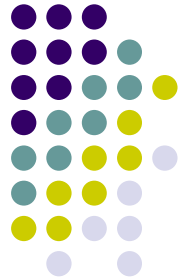
>> *x = [1:10:1000]; y = x.^3;*

>> loglog(x,y)

Semologx = graphisme avec échelle log sur l'axe des x seul

Semology = graphisme avec échelle log sur l'axe des y seul

Améliorer la lisibilité d'une figure



Maquillage (habillage, légendes) d'une figure :

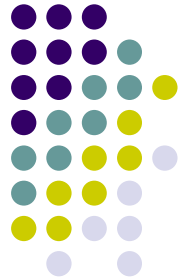
- La commande *xlabel* permet de mettre un texte en légende sous l'axe des abscisses.

>>xlabel(' légende ')

- La commande *ylabel* fait de même pour l'axe des ordonnées. La commande *title* permet de donner un titre à la figure.

>> title('le titre')

Améliorer la lisibilité d'une figure



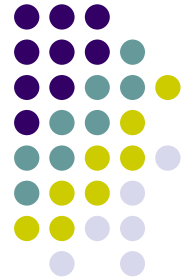
- On peut écrire un texte donné à une position précise sur la figure grâce à la commande *text.*

text(posx , posy,' un texte ')

- *posx* et *posy* sont les coordonnées du point.
- La commande *gtext* permet quant à elle de placer le texte à une position choisie sur la figure à l'aide de la souris.

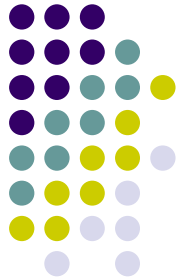
gtext('un texte ')

Améliorer la lisibilité d'une figure



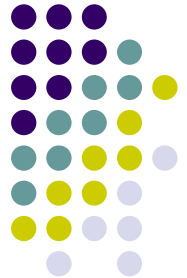
```
>> P = 5;
>> t = [0:.01:2];
>> c = 12*exp(-2*t) - 8*exp(-6*t);
>> plot(t,c); grid
>> xlabel('temps en minutes')
>> ylabel('concentation en gramme par litre')
>> title(['evolution de la concentration du
   produit ', num2str(P), ...    ' au cours du
   temps ']))
>> gtext('concentration maximale')
```

Afficher plusieurs courbes dans une même fenêtre



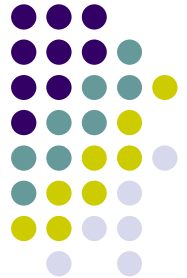
- la commande ***hold on*** permet d'afficher plusieurs courbes dans une même fenêtre
- Pour rétablir la situation antérieure (le résultat d'une nouvelle instruction graphique remplace dans la fenêtre graphique le dessin précédent) on tapera ***hold off***.

Afficher plusieurs courbes dans une même fenêtre



```
>> e = exp(1);
>> figure
>> hold on
>> fplot('exp',[-1 1])
>> fplot('log',[1/e e])
>> plot([-1:0.01:e],[-1:0.01:e])
>> grid
>> hold off
```

Afficher plusieurs courbes dans une même fenêtre



- la commande *subplot*. décompose une fenêtre en sous-fenêtres et d'afficher une figure différente sur chacune de ces sous-fenêtres

>>subplot(m , n , i)

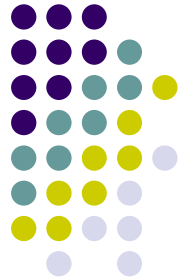
où

m est le nombre de sous-fenêtres verticalement

n est le nombre de sous-fenêtres horizontalement;

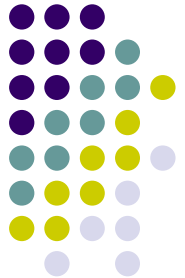
i sert à spécifier dans quelle sous-fenêtre doit s'effectuer l'affichage.

Afficher plusieurs courbes dans une même fenêtre



```
>> figure
>> subplot(2,3,1), fplot('cos',[0 4*pi]),
   title('cosinus'), grid
>> subplot(2,3,2), fplot('sin',[0 4*pi]), title('sinus'),
   grid
>> subplot(2,3,3), fplot('tan',[-pi/3 pi/3]),
   title('tangente'), grid
>> subplot(2,3,4), fplot('acos',[-1 1]), title('arc-
   cosinus'), grid
>> subplot(2,3,5), fplot('asin',[-1 1]),
    title('arc-sinus'), grid
>> subplot(2,3,6), fplot('atan',[-sqrt(3) sqrt(3)]),
   title('arc-tangente'), grid
```

Sauvegarder une figure



- La commande *print* permet de sauvegarder la figure d'une fenêtre graphique dans un fichier sous divers formats d'images.

>>print –f<num> -d<format> <nomfic>

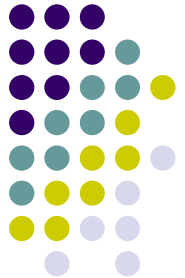
<*num*> désigne le numéro de la fenêtre graphique.

<*nomfic*> est le nom du fichier dans lequel est sauvegardée la figure.

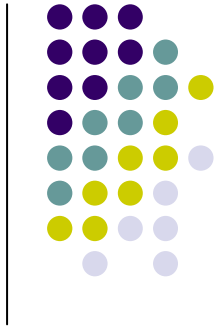
<*format*> est le format de sauvegarde de la figure.

Ces formats sont nombreux. On pourra obtenir la liste complète en tapant help plot.

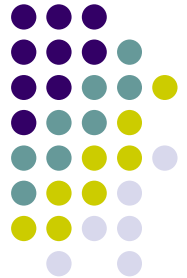
Sauvegarder une figure



- ps : PostScript noir et blanc
- psc : PostScript couleur
- eps : PostScript Encapsulé noir et blanc
- epsc : PostScript Encapsulé couleur
- jpeg : Format d'image JPEG
- tiff : Format d'image TIFF



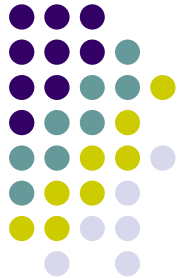
Les entrées – sorties



Les formats d'affichage des réels

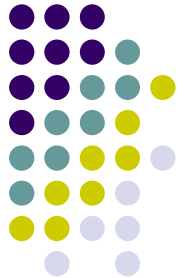
- format long : format long à 15 chiffres.
- format short e : format court à 5 chiffres avec notation en virgule flottante.
- format long e:format long à 15 chiffres avec notation en virgule flottante.

Les formats d'affichage des réels



```
>> pians =    3.1416

>> format long>> pi

>>  format short e

>> pi^3

>> format short g

>> pi^3

>> format short
```

Lecture

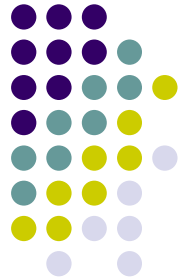


- La commande input permet de demander à l'utilisateur d'un programme de fournir des données.

>>var = input(' une phrase ')

- Une phrase est affichée et MATLAB attend que l'utilisateur saisisse une donnée au clavier.
- Cette donnée peut être une valeur numérique ou une instruction MATLAB.
- Un retour chariot provoque la fin de la saisie.
- Une valeur numérique est directement affectée à la variable var
- Une instruction MATLAB est évaluée et le résultat est affecté à la variable var.

Lecture



- Il est possible de provoquer des sauts de ligne pour aérer le présentation en utilisant le symbole \n

```
>>var = input('\n une phrase : \n ')
```

- Pensez à mettre un point virgule (;) à la fin de l'instruction si vous ne souhaitez pas voir s'afficher var = .
- Pour saisir une réponse de type chaîne de caractères

```
>>var = input(' une phrase ','s')
```

Signalons qu'un retour chariot (sans autre chose) initialise la variable var au tableau vide

```
rep = input(' Affichage du resultat ? o/n [o]
    ','s');
if isempty(rep), rep = 'o'; end
if rep == 'o' |  rep == 'y'
  disp(['Le resultat vaut ', num2str(res)])
end
```