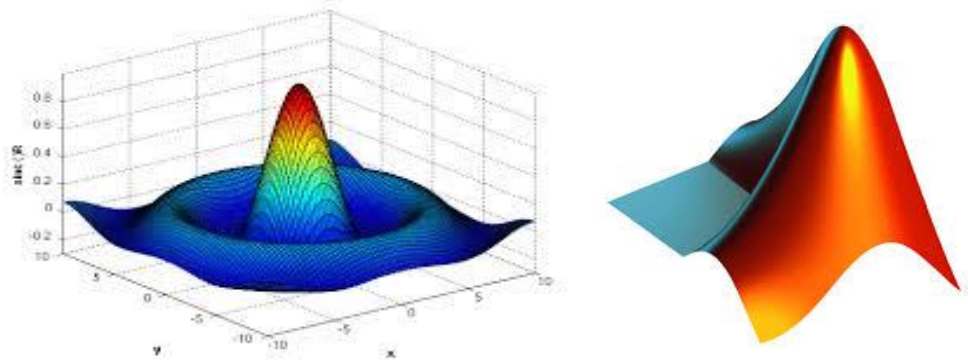
قسم هندسة النقل

Master 1 Hygiène et sécurité industrielle

Matière
Programmation Matlab



Chapitre 1: Introduction à Matlab

Plan

1. Informations sur la matière
2. Introduction à Matlab
3. Etude de l’environnement de Matlab
4. Aide contextuelle
5. Type de données
6. Vecteurs: création, concaténation,
7. Opérations sur les vecteurs
8. Opération sur les poylnômes
9. Matrices et tableaux prédéfinis

Informations sur la matière

- **Matière:** **Programmation Matlab**
- **Enseignante:** **Meriem Bensouyad**
- **Volume Horaire Semestriel**: 45H soit 1h30 Cours + 1h30 TD/TP / semaine
- **Crédits:**4
- **Coefficient:** 2
- **Objectif de l'enseignement**

  Maitrise de la programmation sous **Matlab/simulink**
- **Connaissances préalables**
  - Les principe de base de l'informatique
  - Les algorithmes
  - La programmation en langages structurés
- **Mode d'évaluation:** **Contrôle continu 40% Examen 60%**

Introduction à Matlab

Qu’est ce que MATLAB ?

- MATLAB **MAT**trix **LAB**oratory est un logiciel scientifique de calcul numérique créé en 1984,
- **SIMULINK** est une plateforme de modélisation et de simulation de systèmes dynamiques.
  - **SIMULINK** offre un environnement de développement graphique et une bibliothèque de blocs qui permettent de simuler divers systèmes de contrôle, communication, traitement de signaux.
  - SIMULINK est entièrement intégré à MATLAB, ce qui procure une grande souplesse d’utilisation.

Introduction à Matlab

- MATLAB est une application qui a été conçue afin de fournir un environnement de calcul matriciel simple, efficace, interactif et portable, permettant la mise en

œuvre des algorithmes développés dans le cadre des projets linpack et eispack.

- MATLAB est constitué d'un noyau relativement réduit, capable d'interpréter puis d'évaluer les expressions numériques matricielles qui lui sont adressées :
- soit directement au clavier depuis une fenêtre de commande ;
- –soit sous forme de séquences d'expressions ou scripts enregistrées dans des fichiers texte appelés m-files (ou fichiers .m) et exécutées depuis la fenêtre de commande ;

Pourquoi utiliser Matlab ? (1)

- C'est un logiciel de programmation facile à utiliser
  - Plusieurs fonctions prédéfinies pour analyser et représenter des données : on peut faire des choses élaborées avec très peu de code
- Particulièrement adapté à l'analyse du signal
  - Il existe un module spécialisé d'analyse du signal (et de l'image)
  - Plusieurs fonctions prédéfinies (filtrage par exemple.)
- Création de belles figures
  - Figures stables (cf. Excel…) et esthétiques
  - Automatisation de la création de figures
- Création d'interfaces pour analyser des données variées

Environnement de Matlab (1)

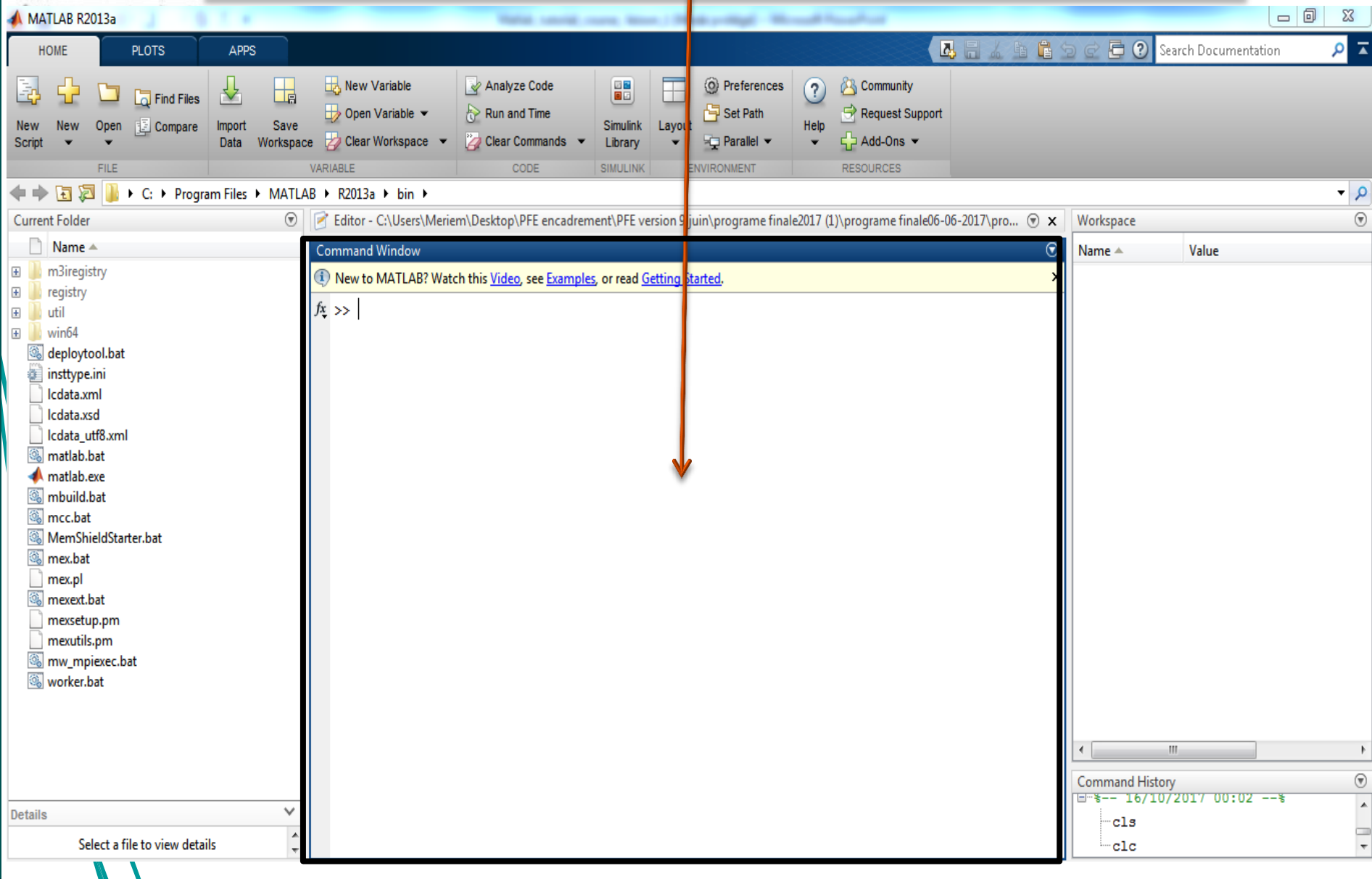
L'environnement de MATLAB possède 4 fenêtres :

- Invite de commande (command window).
- le contenu de l'espace courant de travail (workspace).
- la liste des fichiers du répertoire courant (current folder).
- l'historique des commandes tapées (command history).

- MATLAB nous offre la possibilité d'entrer des commandes dans la fenêtre de commandes avec le **prompt** « >> ».
- Toutes les commandes sont en minuscules et en anglais.
- Lorsque l'on entre une commande, MATLAB affiche systématiquement le résultat de cette commande dans cette même fenêtre.

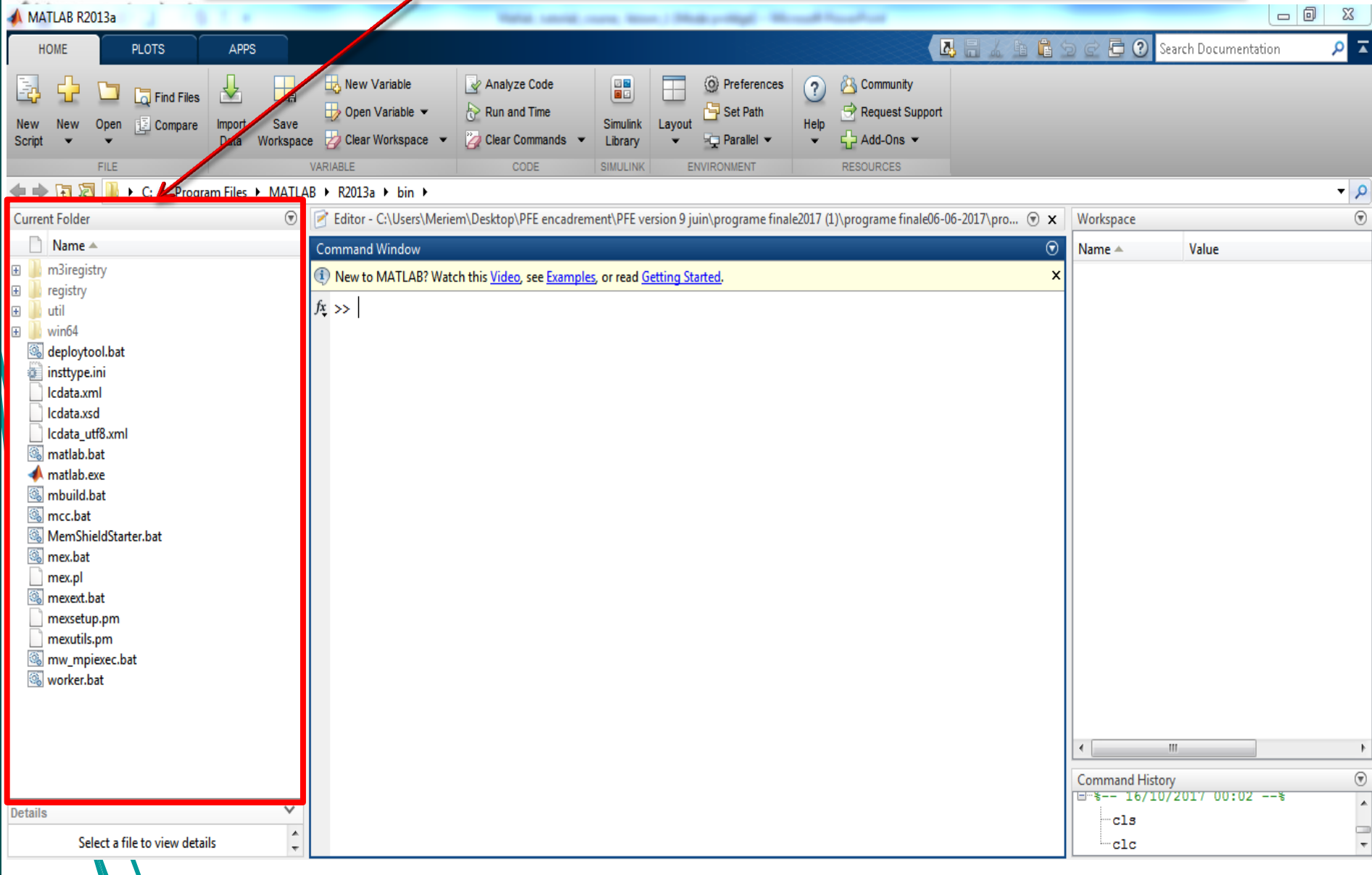
Environnement de Matlab (2)

Invite de commande



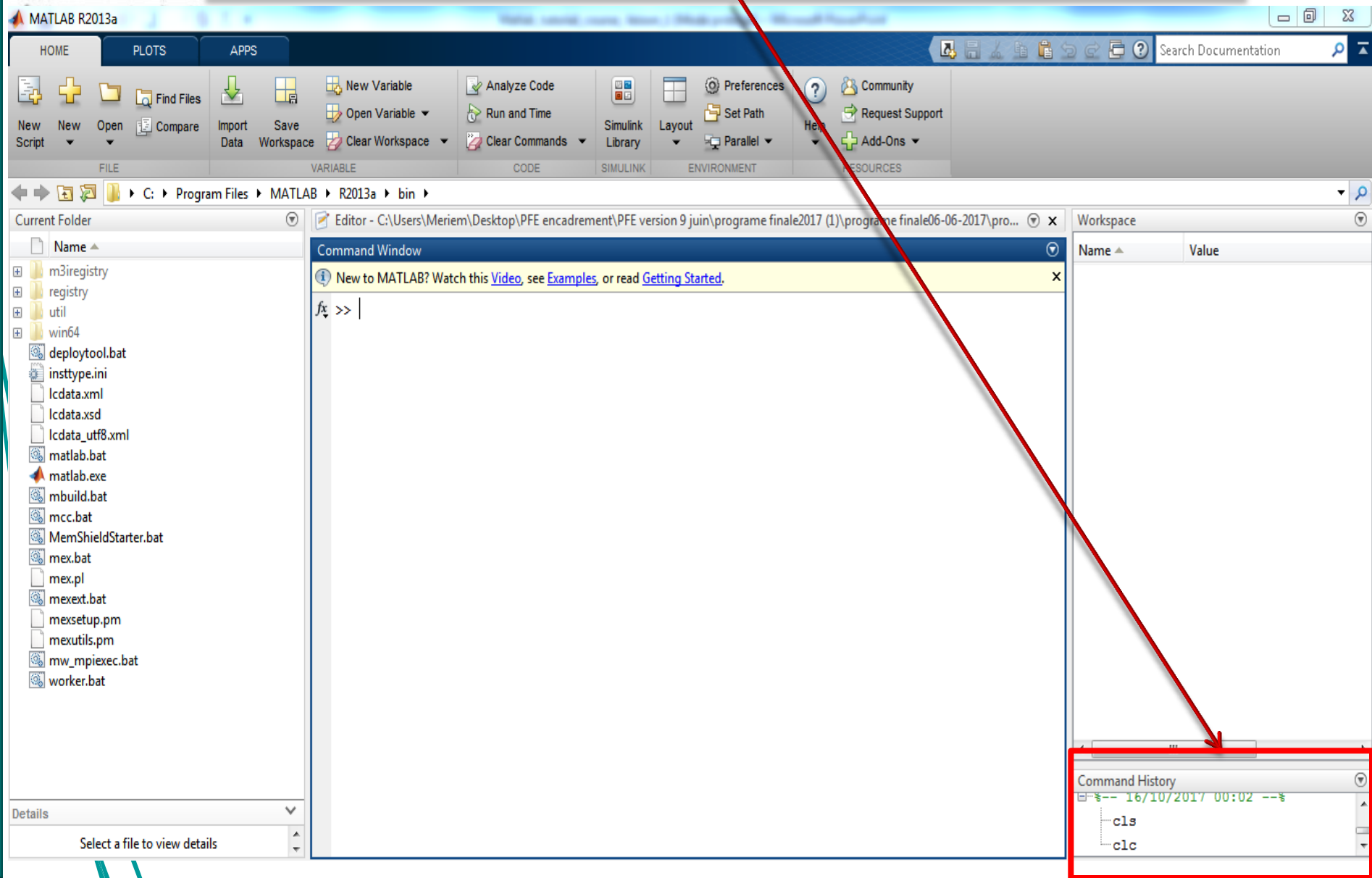
Environnement de Matlab (3)

Répertoire courant



Environnement de Matlab (4)

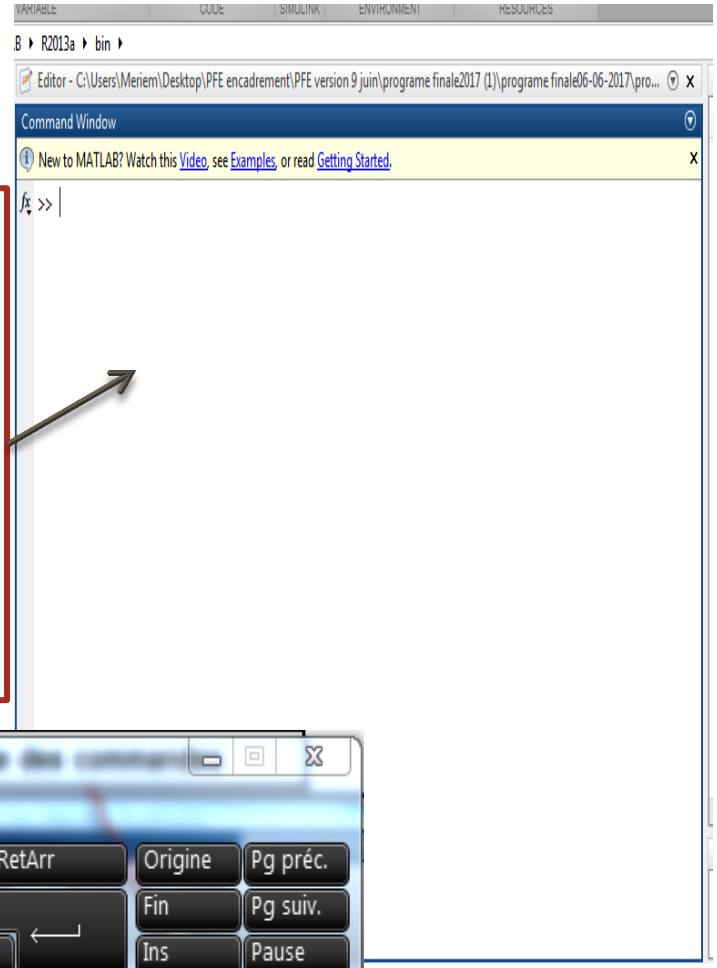
Historique des commandes



Environnement de Matlab (5)

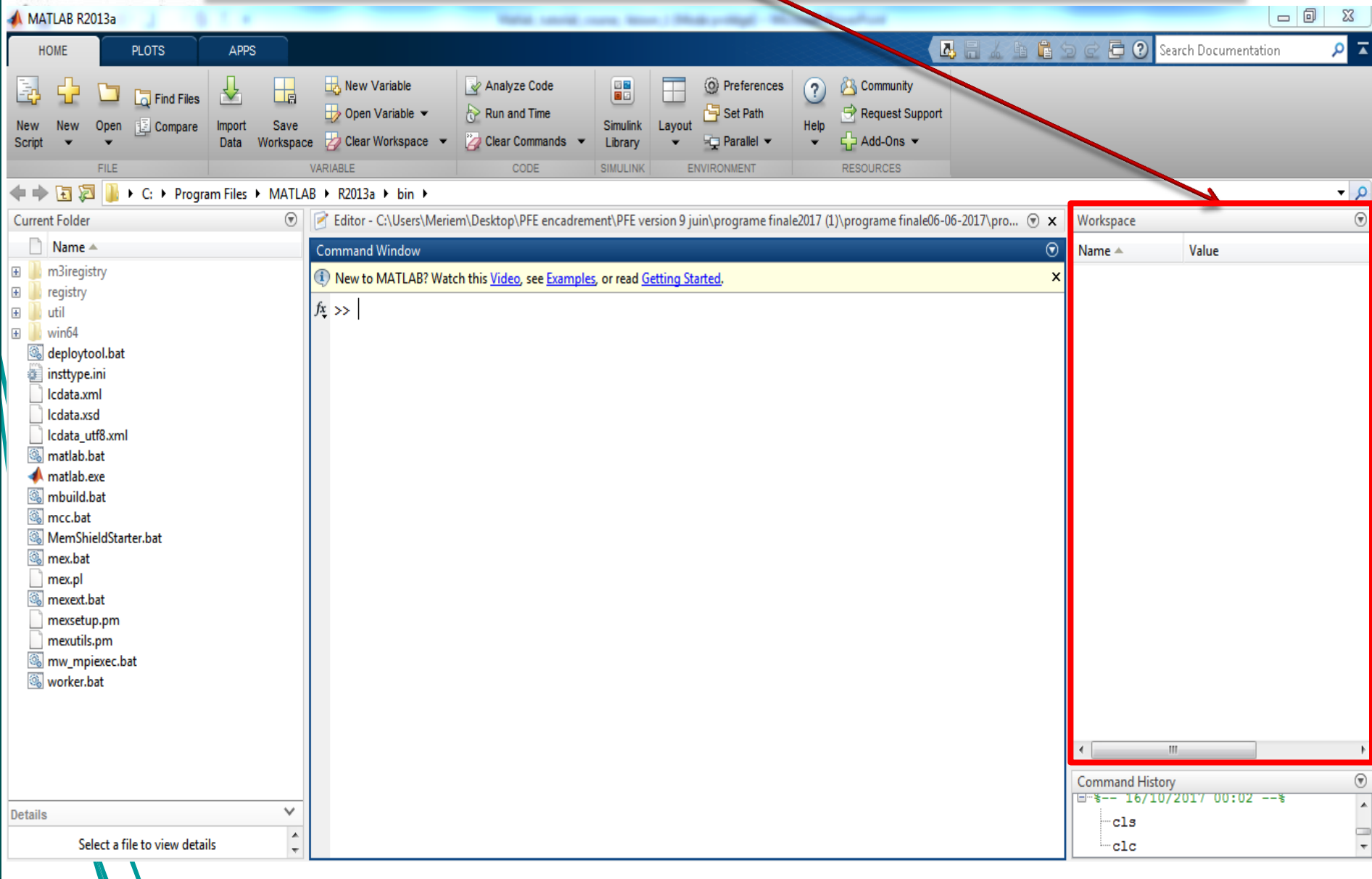
Information utile:

MATLAB conserve l'historique des commandes. Il est donc possible à l'aides des flèches du clavier de remonter dans la liste des instructions déjà entrées pour retrouver une instruction particulière
pour la réutiliser et éventuellement la modifier avant de l'utiliser à nouveau.



Environnement de Matlab (6)

Espace des travail: voir les variables de notre programme

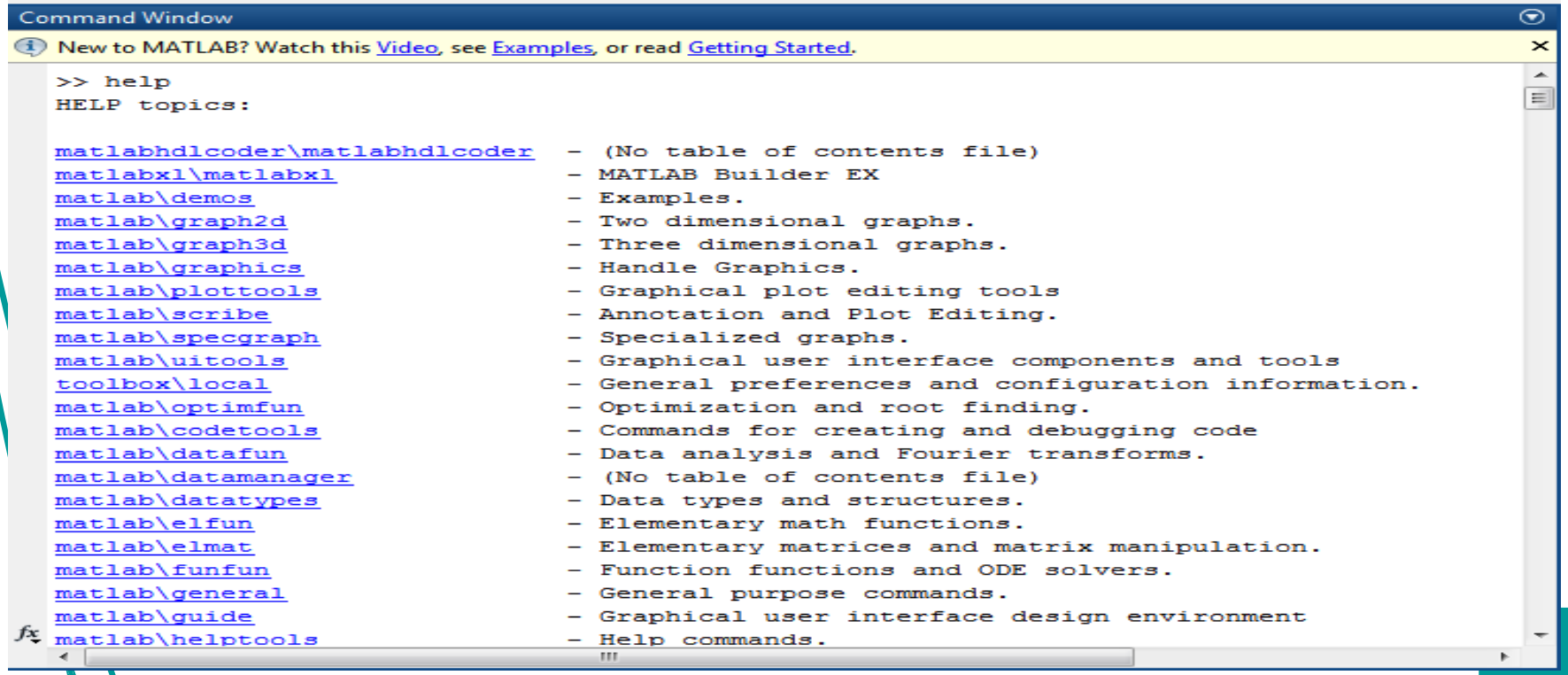


Aide de Matlab (Help)

- Le nombre de fonctions de matlab étant énorme, vous devrez utiliser l'aide quasiment en permanence.
- Deux méthodes sont possibles:
  - En mode **texte**
  - Via **l'interface graphique**

Aide de Matlab :Aide contextuelle

- Saisir des commandes d'aide ,
- Commande **help** vous donne un aperçu des commandes disponibles :



Aide de Matlab :Aide contextuelle

- Pour obtenir les informations concernant une section particulière, entrez help section :

```
>> help elfun
  Elementary math functions.

  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.
    atan2d        - Four quadrant inverse tangent, result in degrees.
    atanh         - Inverse hyperbolic tangent.
    sec           - Secant.
```

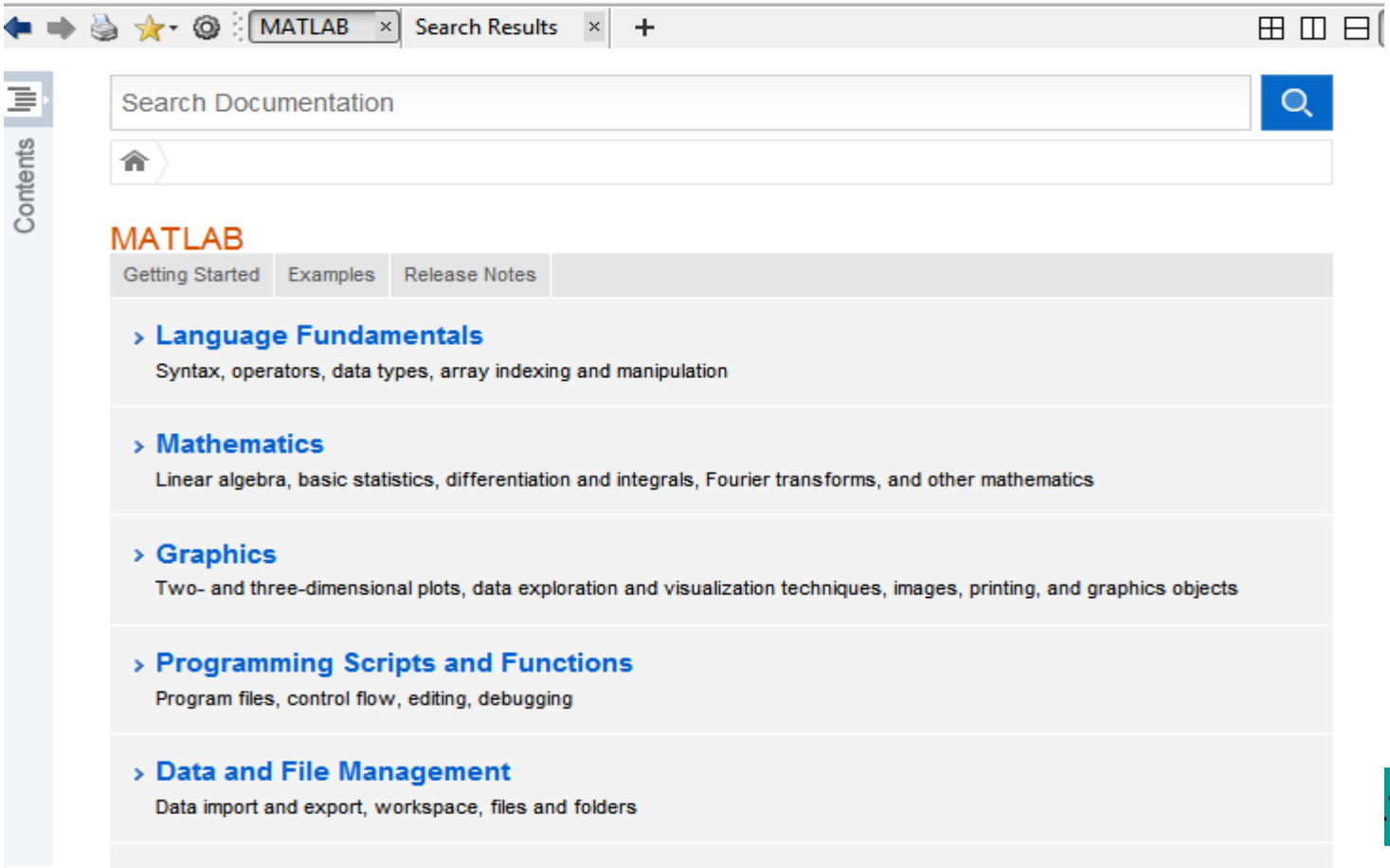
Aide de Matlab :Aide contextuelle

- Pour avoir de l'aide directement sur une commande, entrez help commande :

```
>> help cos
 cos    Cosine of argument in radians.
    cos(X) is the cosine of the elements of X.

    See also acos, cosd.

    Overloaded methods:
       codistributed/cos
       gpuArray/cos

    Reference page in Help browser
       doc cos

fx >>
```

Aide de Matlab: interface graphique

- Vous pouvez également accéder à l'aide via l'interface graphique et ses onglets index et recherche :



Interaction avec le système d'exploitation

- Les commandes et fonctions suivantes permettent à MATLAB d'interagir avec le système d'exploitation de la machine sur laquelle il est utilisé :
- **addpath path** : ajoute le chemin d'accès (path) à la liste des chemins d'accès connus de MATLAB (MATLABPATH) ;
- **cd ou pwd** : affiche le chemin d'accès au répertoire de travail actuel ;
- **cd path** : fixe le répertoire de chemin d'accès path comme répertoire de travail ;

Opérations Arithmétiques

- MATLAB fournit une série des opérations arithmétiques de base :
- a **+** b addition ou somme matricielle
- a **–** b soustraction ou soustraction matricielle
- a / b division ou division matricielle
- a * b multiplication ou produit matriciel
- a^b ou power(a,b) mettre *a* à la puissance de *b*
- **NB:** La racine carrée s'obtient par la fonction **sqrt**

Opérations logiques

- a & b ou and (a,b**)** **et logique**
- a | b ou or(a,b) **ou logique**
- ~a ou not(a) **négation logique**
- xor(a,b) **ou exclusif logique**
- false **valeur logique de faux**
- true **valeur logique de vrai**
- a == b **égalité**
- a ~= b **inégalité**
- a > b **supérieur**
- a >= b **supérieur ou égal**

Commentaires

- Lorsque nous désirons écrire un commentaire dans notre code, le signe de pourcentage (%) permet d'écrire du texte que MATLAB ignorera à l'exécution.

% ceci est un commentaire

x = 3 ; % encore un commentaire

Fonctions prédéfinies

- mod(a, b) le reste de la division entière de a sur b
- abs(a) la valeur absolue
- ceil(a) la valeur entière supérieure
- floor(a) la valeur entière inférieure
- round(a) la valeur entière la plus proche de a
- fix(a) la valeur entière

Types de données: manipulation des variables

- MATLAB gère les **nombres entiers, réels, complexes, les chaînes de caractères** ainsi que les tableaux de nombres de façon transparente : Il est **inutile de déclarer préalablement le type de la variable** que l'on manipule, même pour **les tableaux** et les **matrices**, il suffit simplement d'assigner une valeur au nom de la variable avec l'instruction **« = »**.

```
>> a = 10

a =

     10
```

Types de données: manipulation des variables

- Lorsqu'on n'utilise pas des variables, le résultat de la commande est automatiquement affecté à la variable *ans* qui peut être par la suite utilisée comme une variable normale :

```
>> 5 + 5
ans =
     10
>> a = a + ans
a =
     20
```

Afin de cacher le résultat d'une commande, mettez un point-virgule « ; » à la fin de la commande :

```
>> a = 10 ;
```

Les scalaires

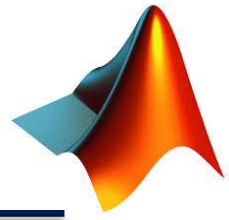
- Le type de scalaire manipulé est transparent pour l'utilisateur.
- Ce type peut être
  - Entier
  - Réel
  - Complexe.

```
>> a = 1
a =
     1
>> b = 1.02
b =
    1.0200
>> x = 1.45e4
x =
    14500
>> c = 1 + 2.4i
c =
   1.0000 + 2.4000i
```

Les scalaires

- MATLAB fournit un ensemble des constants prédéfinis :
  - i, j le nombre imaginaire (racine carré de -1)
  - pi 3.1415...
  - eps 2.2204e-016
  - Inf nombre infini
  - realmax la valeur maximale absolue des réels
  - realmin la valeur minimale absolue des réels

Exemples



```
>> i
ans =
     0 + 1.0000i
>> pi
ans =
     3.1416
>> eps
ans =
     2.2204e-16
>> inf
ans =
   Inf
>> realmax
ans =
     1.7977e+308
>> realmin
ans =
     2.2251e-308
```

Tableaux: vecteurs/Matrices

MATLAB traite un seul type d'objet : les matrices !

- Les scalaires sont des matrices **1 x 1**.
- les vecteurs **lignes** sont des matrices **1 x n**.
- les vecteurs **colonnes** sont des matrices **n x 1**.

Tableaux: vecteurs/Matrices

Syntaxe

un tableau est délimité par **des crochets** ;

- les éléments sont entrés **ligne** par **ligne** ;
- les éléments appartenant à la même ligne sont séparés par des **espaces** (ou par des **virgules**) ;
- les différentes lignes qui doivent posséder le **même nombre d'éléments**, sont séparées par des **points-virgule.**

Tableaux: vecteurs/Matrices

Les tableaux :

$$\begin{array}{|cccc|}\hline 1 & 2 & 3 & 4 \\ \hline \end{array} \qquad \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 3 \\ \hline 4 \\ \hline \end{array} \qquad \begin{array}{|cccc|}\hline 1 & 2 & 0 & 0 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 2 & 2 \\ \hline \end{array}$$

s'écrivent sous la forme `[1 2 3 4 ]` `[1; 2; 3; 4 ]` `[1 2 0 0 ; 0 2 3 1 ; 0 0 2 2 ]` :

```
>>  [1 2 3 4 ]
  ans =
    1   2   3   4

>> [1; 2; 3; 4 ]
  ans =
    1
    2
    3
    4

>> [1 2 0 0 ; 0 2 3 1 ; 0 0 2 2 ]
  ans =
    1   2   0   0
    0   2   3   1
    0   0   2   2
```

Opérateur d'incrémentation

- Le double point (:) est l'opérateur d'incrémentation dans MATLAB.
- Cet opérateur permet de créer des vecteurs de manière automatique.
- Il permet de créer des séquences de nombres en indiquant en option l'intervalle entre ces nombres :

```
>> 1:4

ans =

1 2 3 4

>> 1:1.5:6

ans =

1.0000 2.5000 4.0000 5.5000
```

a = [0:**1**:**5**]; a = [0, 1, 2, 3, 4, 5]

b = [0:**0.2**:1]; b = [0, 0.2, 0.4, 0.6, 0.8, 1]

- Si le pas d'incrémentation n'est pas précisé, il est de **1** par défaut :
- a = [0:5]; a = [0, 1, 2, 3, 4, 5]

Caractères et chaînes de caractères

- On écrit les caractères et les chaînes de caractères entre apostrophes : ’a’, ’toto’
- MATLAB les considère comme des chaînes de caractères de longueur **un**.
- pour MATLAB, les chaînes de caractères et les liste de caractères sont des objets de même nature :

Exemple :

- La liste de caractères [’a’ ’b’ ’c’ ’d’ ’e’] est identique à la chaînes de caractères [’abcde’] :

```
>> [’a’ ’b’ ’c’ ’d’ ’e’]

ans =

abcde
```

’abcde’ ; [’abc’ ’de’] est identique à ’abcde’ :

Caractères et chaînes de caractères

>>[’abc’ ’de’]

ans =

abcde

[ ] sont donc le symbole de l’opérateur de concaténation

Exemple:

mot1 = 'bonjour';

mot2 = 'monde';

phrase = [mot1, ' ', mot2] phrase = ‘bonjour monde’

Caractères et chaînes de caractères

- Il n'est pas possible de mélanger directement les variables numériques et les variables textes.
- Pour afficher une valeur numérique dans un texte, il est nécessaire de la transformer en texte (**string**) à l'aide de la fonction num2str() :

```
a = 52;

b = a/2;

str = ['La moitié de ', num2str(a), ' est ', num2str(b)];
```

Adressage et Indexage

Pour accéder aux éléments d'un vecteur, il faut indiquer l'index de leur emplacement entre parenthèses :

a = [5, 23, 57, 89, 111, 4, 23];

a(2) = 0; a = [5, 0, 57, 89, 111, 4, 23]

a(3) = a(4) + a(5); a = [5, 0, 200, 89, 111, 4, 23]

Si nous désirons accéder à plusieurs éléments pour travailler sur un sous-vecteur, les deux points (:) sont utilisés de cette manière :

b = a(1:3); b = [5, 0, 200]

b(1:2) = 1; b = [1, 1, 200]

Adressage et Indexage: Exemples

```
>> v = [1:1:5]*2
v =
     2     4     6     8    10
>> v(3)    % 3ème élément
ans =
     6
>> v(2:4)    % sous-vecteur du 2ème  au 4ème élément
ans =
     4     6     8
>> v(2:end)   %sous-vecteur du 2ème au dernier élément
ans =
     4     6     8    10
>> v([5 3 1])   % les éléments v(5), v(3), et v(1)
ans =
    10     6     2
```

Taille d'un vecteur

- La fonction size() permet de connaître la taille d'un vecteur ou d'une matrice, selon les deux dimensions :

```
vl = [7, 2, 45, 67, 3]

vc = [6; 45; 67; 2]

size(vl);    [1, 5]

size(vc);   [4, 1]
```

- Puisque les vecteurs n'ont qu'une dimension, la fonction **length()** est plus simple, car elle retourne la plus longue dimension :

```
length(vl);    5

length(vc);   4
```

Les fonctions linspace et logspace

- La fonction linspace(i, j, k) génère un vecteur de k éléments allant de i à j
- La fonction logspace(i, j, k) génère un vecteur de k éléments allant de 10 ^i à 10^j :

```
v = logspace(0, 2, 5)

v =

    1.0000    3.1623   10.0000   31.6228  100.0000

  % 10^0,    10^1.5,  10^1,     10^1.5,  10^2
```

La fonction rand

- La fonction rand initialise des vecteurs lignes (rand(1, k)) ou colonnes (rand(1, k)) par des valeurs aléatoires entre 0 et 1 :

```
>> v1 = rand(1, 5) % 5 valeurs aléatoires (0–1)

v1 =

    0.0975    0.2785    0.5469    0.9575    0.9649

>> v2 = rand(1, 5) % 5 autres valeurs aléatoires (0-1)

v2 =

    0.1576    0.9706    0.9572    0.4854    0.8003

>> v3 = 10 * rand(1, 5) % 5 valeurs aléatoires (0-10)

v3 =

    1.4189    4.2176    9.1574    7.9221    9.5949
```

Opérations sur Les Vecteurs

```
>> v = [1 2 3 4]; w = [2 4 6 8];
>> v + w    % addition
ans =
     3     6     9    12
>> v – w    % soustraction
ans =
    -1    -2    -3    -4
>> v * w'   % produit scalaire
ans =
    60
>> v .* w   % produit terme-à-terme
ans =
     2     8    18    32
```

Opérations Statistiques

```
>> x = [2 4 5 6 7];
>> max(x)   % maximum
ans =
     7
>> min(x)   % minimum
ans =
     2
>> sum(x)   % somme des valeurs
ans =
    24
>> prod(x)   % produit des valeurs
ans =
        1680
>> mean(x)   % moyenne des valeurs
ans =
    4.8000
>> median(x)  % la valeur médiane (après le tri)
```

Opérations Statistiques

```
ans =
     5
>> var(x)   % la variance des valeurs
ans =
    3.7000
>> std(x)   % l'écart type des valeurs
ans =
    1.9235
```

Tri d'un vecteur

- Deux opérations particulièrement utiles à effectuer sur des vecteurs sont les opérations de **tri** et de recherche.
- La commande **sort(A)** retourne un vecteur trié en ordre croissant depuis un vecteur d'entrée A.
- la commande **sort** retourne deux vecteurs:
  - vecteur des valeurs triées
  - vecteur de leurs indices avant le tri.

```
>> A = fix(100*rand(1, 10))    %10 valeurs entre 0 et 100
A =
   85   62   35   51   40    7   23   12   18   23
>> sort(A)
ans =
    7   12   18   23   23   35   40   51   62   85
```

Tri d'un vecteur

```
ans =
    7    12    18    23    23    35    40    51    62    85
```

```
>> [svecteur ivecteur] = sort(A)
svecteur =
    7    12    18    23    23    35    40    51    62    85
ivecteur =
    6     8     9     7    10     3     5     4     2     1
```

Tri d'un vecteur

Les commandes sort(A, 'ascend') et sort(A, 'descend') retournent un vecteur trié en ordre croissant (décroissant respectivement).

```
>> sort(A, 'ascend')

ans =

     7    12    18    23    23    35    40    51    62    85


>> sort(A, 'descend')
ans =

    85    62    51    40    35    23    23    18    12     7
```

Recherche d'un Elément dans un vecteur

- La commande **find(A)** retourne un vecteur des indices des valeurs non-nulles d'un vecteur A.
- **find(cond A)** retourne les indices des valeurs satisfaisantes d'une condition sur le vecteur A.

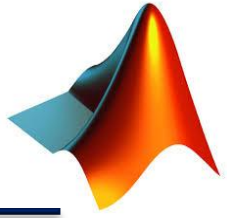
Exemple

```
>> A = fix(100*rand(1, 10))-50        % 10 valeurs entre -50 et 50

A =
   -27   -15    32   -49   -46   -34    14    23    14    -5
>> A(2) = 0; A(6) = 0  % rendre quelques valeurs nulles
A =
   -27     0    32   -49   -46     0    14    23    14    -5
>> j = find(A)    % sauvegarder les indices
ans =
     1     3     4     5     7     8     9    10
```

Recherche d'un Elément dans un vecteur

```
>> A(j)        % afficher les valeurs non nulles
ans =
   -27    32   -49   -46    14    23    14    -5
>> [Lin Col B] = find(A)          % i, j, A[i, j]
Lin =
     1     1     1     1     1     1     1     1
Col =
     1     3     4     5     7     8     9    10
B =
   -27    32   -49   -46    14    23    14    -5
>> find(A>=0)
ans =
     2     3     6     7     8     9
```

Recherche d'un Elément dans un vecteur



find(A<0)

```
ans =

     1     4     5    10

>> find(A>30)

ans =

     3

>> find(A==30)

ans =

   Empty matrix: 1-by-0


>> find(A==14)

ans =

     7     9
```

Recherche d'un Elément dans un vecteur

- La commande **find(A, k)** ou **find(A, k, 'first')** retourne au plus un vecteur des k premiers indices des valeurs non-nulles d'un vecteur A.
- La commande **find(A, k, 'last')** retourne au plus un vecteur des k derniers indices des valeurs non-nulles d'un vecteur A.
- Les commandes **find(cond A, k, 'first')** et **find(cond A, k, 'first')** filtrent le vecteur A selon la condition introduite et prennent les k premiers ou k derniers indices.

Recherche d'un Elément dans un vecteur

```
find(A>=0)
ans =
     2     3     6     7     8     9
>> find(A>=0, 2)
ans =
     2     3
>> find(A>=0, 2, 'first')
ans =
     2     3
>> find(A>=0, 2, 'last')
ans =
     8     9
```

Modification, suppression et insertion, au niveau des vecteurs

Modification d'un élément du vecteur

La modification d'un élément (ou plusieurs éléments) dans un vecteur peut être effectuée par de son index (ou leurs index) comme suit:

A(i_1): l'élément i_1 du vecteur A.

A(i_1 : i_2): un sous-vecteur de A à partir de l'indice i_1 jusqu'à l'indice i_2 .

A(i_1 : **pas**: i_2): un sous-vecteur de A à partir de l'index i_1 jusqu'à l'index i_2 avec un pas.

A([i_1 i_2 … i_n]): un sous-vecteur de A en prendre en considération les indices i_1 , i_2 , …, i_n .

Modification d'un élément du vecteur
Exemples

```
>> A = zeros(1, 10)
A =
     0     0     0     0     0     0     0     0     0     0
>> B = ones(1, 10)
B =
     1     1     1     1     1     1     1     1     1     1
>> A(4) = B(7)    % un seul élement
A =
     0     0     0     1     0     0     0     0     0     0
```

```
>> A(1:3) = B(5:7)  % un sous-vecteur contigu
A =
     1     1     1     1     0     0     0     0     0     0
>> A(5:2:9) = B(1:2:5) % un sous vecteur non contigu
A =
     1     1     1     1     1     0     1     0     1     0
>> A([6 8 10]) = B(8:10) % des éléments bien déterminés
A =
     1     1     1     1     1     1     1     1     1     1
```

Suppression d'un élément du vecteur

- La suppression d'un ou plusieurs éléments d'un vecteur peut être effectuée en utilisant le vecteur vide []:

```
>> A(5) = []

A =

     1     2     3     4     6     7     8     9    10

>> A(1:3) = []

A =

     4     6     7     8     9    10
```

L'insertion d'un nouvel élément dans un vecteur

- L'insertion d'un nouvel élément dans un vecteur nécessite l'index exact de l'élément afin de concaténer la partie gauche, le nouvel élément et la partie droite, en construisant le nouveau vecteur:

```
>> A = [2 4 8 10 12]   % vecteur de 5 éléments

A =

    2    4    8   10   12

>> x = 6; i = 3;    % élément à insérer et son indice
```

A= **[A(1:i-1) , x , A(i:end)]**

```
>> A = [A(1:i-1) x A(i:end)] % concaténer partie gauche + élément + partie droite

A =

    2    4    6    8   10   12
```

Références

- Ce cours est principalement basé sur le cours de **Dr. Mohammed Omari**, de la matière: Outils de Programmation pour les Mathématiques
- http://wp.unil.ch/risk/files/2015/12/1.-Introduction-%C3%A0-Matlab.pdf

Merci