Laboratoire 1 – Introduction à Matlab, PRTools et DRTools

L'objectif de ce laboratoire session est de se familiariser au logiciel Matlab et à la boite à outil (*Toolbox*) PRTools qui seront les outils principaux de ce laboratoire. Dans ce qui suit vous trouverez un aperçu rapide sur Matlab, mais il est fortement conseillé de consulter les références citées dans la dernière section (spécialement pour ceux qui n'ont pas beaucoup d'expérience avec Matlab). Concernant PRTools, vous trouverez aussi un lien pour le manuel (à consulter) dans la section 1.8, ainsi qu'un script (exp1_data.m) à explorer.

MATLAB (*MATrix LABoratory*) est un langage de calcul scientifique basé sur le calcul matriciel. Contrairement aux autres langages de programmation, il s'occupe de l'allocation mémoire. L'utilisation de MATLAB peut se faire de façon interactive (exécution directe des commandes) ou par des scripts (des programmes et des fonctions). Les boites à outils (*toolboxes*) sont des ensembles de fonctions dédiées à chacun des divers domaines scientifiques tels que le traitementde signal (*Signal Processing Toolbox*), la reconnaissance de formes (*Pattern Recognition Toolbox* ou *PRTools*), etc.

1.1 Démarrer, quitter et description de la fenêtre du Matlab

Pour démarrer Matlab, sous Windows il suffit de cliquer sur l'icône de Matlab (ou bien le chercher dans le menu Démarrer), alors que sous Unix/Linux il suffit de taper `matlab &` dans une fenêtre du terminal (il se peut qu'il soit nécessaire de taper le chemin d'accès avant la commande, *e.g.* `/opt/matlab/bin/matlab`). Généralement, on obtient la fenêtre suivante (ça peut changer selon les versions du logiciel) :

A) La fenêtre commande (*Command Window*) est la partie essentielle utilisée pour définir les variables et lancer les commandes où les scripts (m-files, voir section 1.2). Avec la touche Haut (↑) on peut réutiliser des commandes tapées précédemment. Après avoir édité les commandes, on les exécute avec la touche Enter.

B) L'éditeur Matlab (*Editor*) offre des fonctionnalités d'édition et de débogage standard, comme la définition de points d'arrêt et l'exécution en mode pas à pas.

C) Matlab utilise le répertoire courant (*Current Directory*) et le chemin d'accès (*Search Path*) comme points de référence. Tout fichier qu'on veut utiliser ou exécuter (en particulier les M-files) doit être contenu soit dans le *Current Directo*ry, soit dans un dossier appartenant au *Search Path* (pour définir ces dossiers il faut suivre `File` → `Set Path`). Une façon rapide pour changer de répertoire et de créer, modifier ou effacer des fichiers c'est d'utiliser le champ *Current Directory*, qui fonctionne comme un gestionnaire de fichiers graphique classique. Alternativement, on peut utiliser des commandes telles que `dir`, `cd` ou `delete` dans le *Command Window*.

D) L'espace de travail (*Workspace*) affiche des informations sur toutes les variables affectées. Pour voir une représentation graphique des variables il suffit de double-cliquer sur celle choisie. Elle est ainsi affichée dans le *Array Editor* où on peut aussi l'éditer.

E) Toutes les commandes entrées dans le *Command Window* sont mémorisées et affichées dans le champ *Command History*. Chaque session possède sa propre arborescence et commence

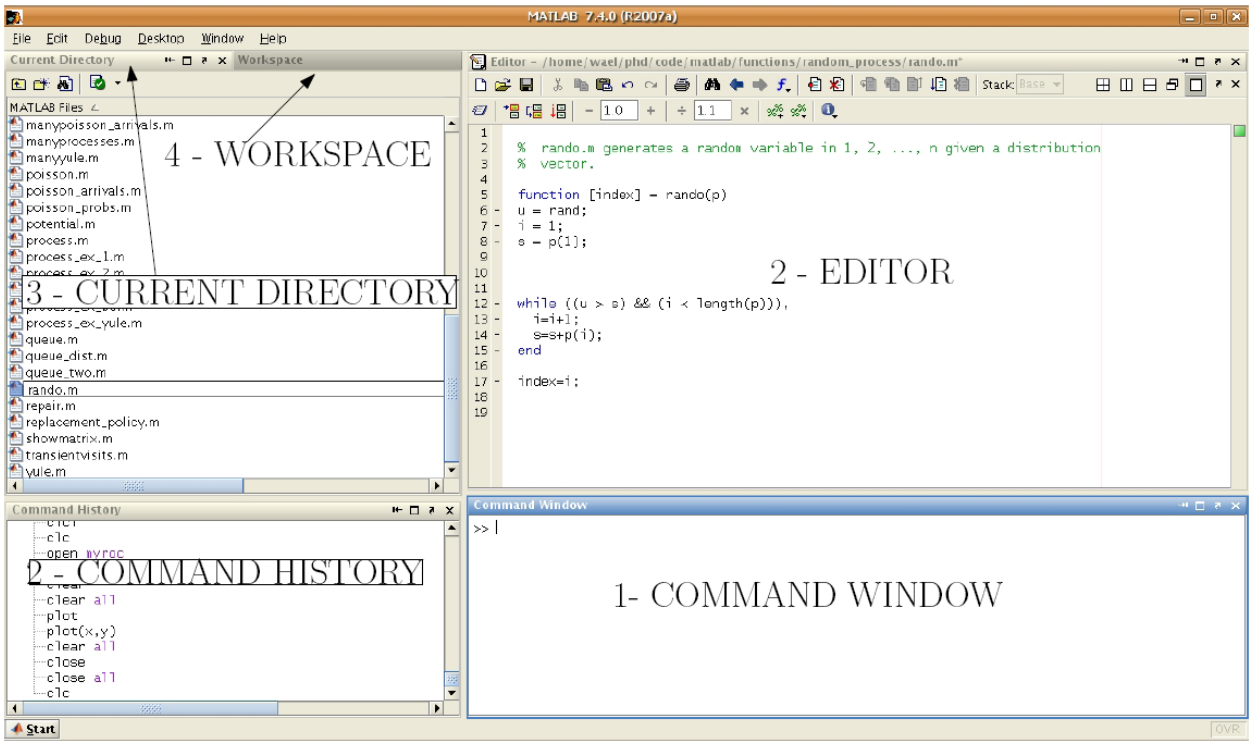


Figure 1 – Interface graphique de Matlab

par une ligne avec date et horaire qui est affichée en vert. En double-cliquant sur une commande affichée, elle est automatiquement exécutée dans le Command Window. En sélectionnant une ou plusieurs commande et ensuite en utilisant la touche droite de la souris, on peut créer directement un M-file qui contient les commandes sélectionnées.

- La fenêtre Matlab peut être personnalisée en utilisant : `File` → `Preferences`
- la commande demo (où bien help → demo) fait apparaître une fenêtre présentant des démonstrations du langage ainsi que des boites à outils.
- `who` liste les variables qui se trouvent dans *workspace*
- `what` retourne le *current directory* et les fichiers (*m-files*) qui sont dans cet répertoire

On quitte Matlab en tapant `quit` dans la fenêtre de commande ou en sélectionnant quit dans le menu File.

1.2 Les fichiers Matlab

Les fichiers Matlab sont des fichiers texte dont l'extension est .m (*m-files*). Ces fichiers sont classés par thèmes et constituent des boîtes à outils (*Toolboxes*). Comme les bibliothèques doivent être accessibles de n'importe quel point où l'on se place dans l'arborescence du disque, une variable PATH doit être configurée pour définir les chemins d'accès. L'ensemble des liens vers des bibliothèques existantes est disponible en tapant dans la fenêtre de commandes : `path`. Pour charger une librairie qui n'est pas dans le répertoire courant mais dans un autre chemin, on va utiliser la commande `addpath('nom_chemin')` pour ajouter le chemin dans la liste des répertoires que Matlab va consulter à l'exécution d'une commande.

Il existe néanmoins deux types fichiers :

Les librairies : Un fichier de librairie est dédié à la définition d'une fonction. Il est préférable de définir une seule fonction par fichier et pour des raisons pratiques, le fichier possède le même nom que la fonction et l'entête suit la syntaxe suivante :

```
function [y1, y2, . . . , yn] = nom_de_la_fct(x1, x2, . . . , xp)
```

Les scripts : Ils sont exécutés par Matlab en tapant simplement leur nom et contrairement aux fonctions, ils ne prennent pas d'arguments en entrée. Ces fichiers contiennent une suite d'instructions. Ces instructions pourraient tout aussi bien être exécutés directement dans la fenetre de commande Matlab . On peut donc y effectuer des opérations d'entrées/sorties, des calculs, exécuter des commandes et des fonctions.

En général, chaque programme débute par un commentaire de quelques lignes décrivant l'objectif du programme, les entrées, les sorties, etc. Pour avoir de l'aide, on peut taper les commandes suivantes :

`help` sans arguments donne une liste thématique (essayer : `help help`)

`help nom_fonction` donne la définition de la fonction désignée et des exemples d'utilisation

doc similaire à help mais l'affichage en est format HTML dans le navigateur de l'aide

lookfor "sujet" donne une liste des rubriques de l'aide en ligne en relation avec le "sujet" indiqué

- **Exemple :** `lookfor random`, donne une liste des fonctions ayant le mot `random` dans leur texte de description.

Autres commandes utiles d'interaction avec l'environnement (idem aux commandes sous linux) :

which nom_fonction : pour localiser la fonction sur le disque dur

pwd indique le répertoire de travail

cd permet le déplacement dans l'arborescence du disque

ls description du contenu du répertoire

1.3 Les types de données

1.3.1 Constantes prédéfinis

NaN (*Not-A-Number*) représente l'élément non défini (*e.g.*, `x = 0/0`)

Inf l'infini (*e.g.*, `x = 1/0`)

eps le zéro machine

pi le nombre $\pi = 3.146...$

i, j i (ou j) représente le nombre imaginaire unité ($\sqrt{-1}$)

realmin : plus petit nombre réel positif

realmax : plus grand nombre réel positif

Les fonctions `isnan` et `isinf` permettent de savoir si dans une matrice les éléments `NaN` et `Inf` sont présents.

NB : si une valeur a été assignée à une de ces constantes, ou encore à une fonction existante de Matlab, la constante (ou fonction) peut être libérée à l'aide de la commande `clear` :

– **Exemple :** après la commande `pi = 18;` qui assigne la valeur 18 à la variable `pi`, la commande `clear pi` libère cette variable qui reprend sa valeur par défaut (`pi` $= \pi$).

1.3.2 Les scalaires

Dans Matlab, il n'y pas de différence entre une matrice, un vecteur et un scalaire. Tous sont des tableaux, seules les dimensions changent. On peut assigner explicitement une valeur fixe à une variable (un tableau de dimension 1×1) :
$>>$ `a` $=$ `3`
Ou bien comme une fonction d'autres variables :
$>>$ `b` $=$ `a` $+$ `2`
Pour supprimer l'affichage de l'opération, ajouter un « ;» en fin de ligne :
$>>$ `b` $=$ `b` $*$ `2;`

La valeur d'une variable est toujours affichée dans le champ *workspace*, avec un clic droit on peut la supprimer, la renommer, etc., avec un double-clic elle est ainsi affichée dans le *Array Editor* où on peut aussi l'éditer. Alternativement, on peut utiliser les commandes :

`who` donne la liste des variables présentes dans l'espace de travail.

`whos` retourne une information plus complète comportant pour chaque variable, la dimension du tableau qui lui est associé, la quantité de mémoire utilisée et la classe à laquelle il appartient.

`disp` pour l'afficher dans la fenêtre de commande (on peut également taper le nom de la variable)

`ans` contient la dernière réponse en date

`clear` nom_variable : supprime la variable indiquée ;

clear all : supprime toutes les `variables` de l'espace du travail (Revient à faire `Edit` $\rightarrow$ `Clear Workspace`).

clc efface tous ce qui est déjà écrit dans la fenêtre commande (sans toutefois perdre les données des variables)

NB : Dans Matlab les variables ne sont ni déclarées ni typées, il ne distingue pas entre réels et entiers, ils sont écrits sous les formes décimales ou scientifiques usuelles (voir `help format`).

1.3.3 Les vecteurs

Pour initialiser des vecteurs il existe plusieurs méthodes :

A) L'énumération explicite (la virgule peut être remplacer par un espace)
$>>$ `v` $= [$`1, 2, 4,` $-$`6`$];$

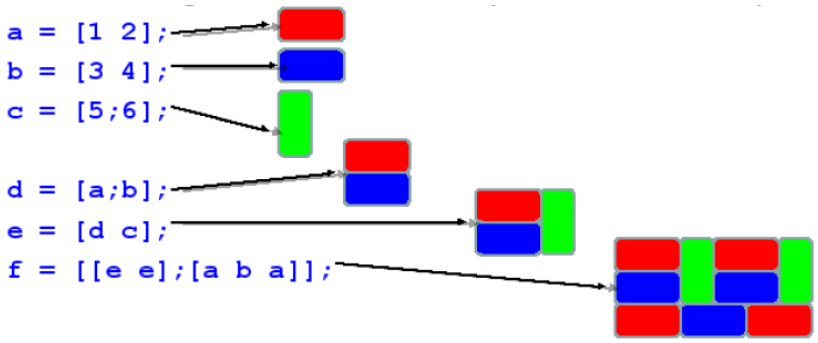
B) La progression régulière d'une unité ou avec incrément quelconque
$>>$ `v` $=$ `1 : 5;`
$>>$ `v` $=$ `1 : 3 : 10; % incrément de valeur 3` (la partie de la ligne qui suit le symbole % est un commentaire)

C) La progression régulière, linéaire ou logarithmique, avec bornes de départ et d'arrivée, ainsi que le nombre de points spécifiés :
$>>$ `v` $=$ `linespace`$($`1, 2, 4`$);$
$>>$ `v` $=$ `logspace`$($`1, 2, 4`$);$

Pour avoir des vecteurs colonne il faut utiliser des points-virgules au lieu des virgules ou bien transposer les vecteurs lignes en utilisant l'opérateur apostrophe ′ :
>> `vecteur_colonne` $= [1; 2; 4; -6]$;
>> `vecteur_colonne` $=$ `vecteur_ligne`$'$;

1.3.4 Les matrices

La création de matrices est similaire à celle des vecteurs. La seule contrainte est que toutes les lignes d'une matrice doivent avoir le même nombre d'éléments (c'est également vrai pour les colonnes d'une matrice). L'initialisation de matrice par énumération explicite se fait comme suit :
>> `A` $= [1, 2, 3; 5, 6, 7]$;
On ne peut pas insérer simplement une colonne ou une ligne à l'intérieur d'une matrice, mais on peut concaténer deux ou plusieurs vecteurs (ligne ou colonnes) ou également des matrices à conditions que les dimensions soit égales :



Autres fonctions d'initialisation pour des matrices particulières (de dimension `m` $\times$ `n`) :

```
>> a = zeros(m,n); \% crée une matrice dont tous les éléments sont nuls.
>> b = ones(m,n);  \% crée une matrice dont tous les éléments ont la
                   \% valeur 1.
>> c = nan(m,n);   \% crée une matrice dont tous les éléments ont la
                   \% valeur NaN.
>> d = eye(m,n);   \% crée une matrice identité
>> e = rand(m,n);  \% crée une matrice remplie de nombres aléatoires
                   \% uniformément distribués.
```

Accès au contenu des matrices : Soit la ligne i et la colonne j d'une matrice A.

- Lecture d'un élément : `A(i,j)`.
- Modification d'un élément : `A(i,j)=d`
- Lecture d'une ligne, d'une colonne, ou d'un vecteur d'indice :
  - `A(i,:)` renvoie la ième ligne de A.
  - `A(:,j)` renvoie la jème colonne de A.
  - `A(x:y,j)` renvoie la matrice contenant toutes les colonnes et les lignes d'indice x à y.
- Suppression de la ième ligne (les numéros de lignes seront décalés) : `A(i,:)=[]` ;

Tailles de matrices : `length` retourne la plus grande des dimensions de son argument.

size retourne le nombre de lignes (1ère sortie) et de colonnes (2ème sortie).

Exemple à copier dans un script :

```
A = rand(10,5)              \%-- Matrice 10 lignes et 5 colonnes
                            \%   d'éléments.
                            \%   aléatoire selon une distribution
                            \%   aléatoire U[0,1].
[x,y] = size(A)             \%-- x=10 et y=5
long = length(A)            \%-- long=10
a = A(5,5)                  \%-- 5e ligne et 5e colonne de A
B = A(1,:)                  \%-- 1ère ligne de A
C = A(:,end)                \%-- Dernière colonne de A
D = A(1:3:end,[3 4])        \%-- Sous-matrice de A avec les lignes 1, 4,
                            \%   7 et 10, et les colonnes 3 et 4.
A(1,:) = []                 \%-- 1ère ligne supprimée
clear all                   \%-- Libère toutes les variables
clc                         \%-- Nettoie la fenêtre de commande (Command
                            \%   Window)
A = [1 2 3; 4 5 6; 7 8 9]; \%-- Nouvelle matrice A
a = A(3,2);                 \%-- 3e ligne et 2e colonne de A
b = A(6);                   \%-- Indice linéaire (voir help ind2sub)
disp('     a     b'),       \%-- Affiche a et b. Voir également la
                            \%   fonction fprintf.
A(:)                        \%-- Crée un vecteur colonne (colonne par
                            \%   colonne de la matrice A)
B = A([3 2 1],:)            \%-- échange de colonnes 1 et 3
```

Le tableau 1) présente les principales opérations matricielles. Pour que les opérateurs de produit (*), division (/), et puissance (^) agissent sur les éléments à l'intérieur des matrices (*i.e.* élément par élément), plutôt que sur les matrices, il faut ajouter un point (.) à gauche de l'opérateur (à condition que les matrices soient de même taille). Dans le cas des scalaires, il n'y a pas de distinction. Voici quelques exemples :

```
>> a=[1;2;3];  b=[4;5;6];  c=[a'; b'];  d=c(:,1:2);
>> p1 = a'*b;     \%-- Produit scalaire
>> p2 = a.*b;     \%-- Multiplication élément par élément
>> p3 = d*c;      \%-- Produit matriciel
>> a1 = a.^2;     \%-- Puissance élément par élément
>> ic = c > 2;    \%-- Les éléments de ic ont une valeur de 1  si
                  \%   ou les éléments de c sont plus grand que 2,
                  \%   et 0 ailleurs
```

Tableau 1 – Opérations matricielles (arithmétiques et logiques)

symbole	définition
[]	définition matricielle et concaténation
;	séparateur de colonne
()	extraction et insertion d'un élément
′	transposition
+	addition
−	soustraction
*	produit matriciel
\	division à gauche
/	division à droite
^	puissance
==	égal à
~=	différent de
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à
&	et
|	ou
~	non

1.4 Structures algorithmiques

Il existe deux types de boucles en Matlab : les boucles for et les boucles while. Pour effectuer un test, on peut utiliser la combinaison classique if-then-else ou bien le switch-case. Les syntaxes sont :

```
for i=deb:pas:fin
   ...
end

%-- Si v est
%   un vecteur
for i = v
   ...
end
```

```
while expression
   ...
end
```

```
if condition
   ...
elseif
   ...
else
   ...
end
```

```
switch var
  case val1
     ...
  case val2
     ...
  otherwise
     ...
end
```

1.5 Affichage des courbes 2D

Matlab possède un grand nombre de fonctions permettant de produire des courbes 2D. Chacune accepte en entrée des formes vectorielles ou matricielles automatiquement mises à l'échelle suivant les axes définis par ces données. La fonction `plot` est la fonction de base pour l'affichage.

plot génère l'affichage des éléments d'un vecteur ou des colonnes d'une matrice. Les couleurs et les symboles associés aux données sont paramétrables (voir `help plot`)

`plot(y)` produit un affichage linéaire des éléments de vecteur **y** suivant l'index du vecteur.

`plot(x,y)` produit l'affichage des éléments de **y** suivant ceux de **x**.

subplot(n,p,r) permet de gérer le nombre de graphes que l'on crée sur une même figure. Elle partitionne la figure comme une matrice de n ligne et p colonnes, où chaque élément de la matrice est un graphiques. L'emplacement d'un graphique (colonne par colonne) est indiqué par r.

figure permet d'ouvrir autant de fenêtres graphiques que désiré. Chaque figure porte un numéro qui permet de la référencer et ainsi de savoir où l'on envoie les sorties graphiques. Les figures sont indexées automatiquement.

close permet de fermer une figure à partir de l'espace de travail (`close all` ferme toutes les figures).

title création d'un titre

xlabel commentaire sur x

ylabel commentaire sur y

grid création d'une grille

text commentaire sur graphe

axis gestion des axes (zoom)

hold mode surimpression

Pour la visualisation des données en 3D, voir le help pour les fonctions `plot3`, `mesh` et `surf`.

1.6 Importation et exportation de données

Les échanges de données entre applications utilisent généralement des fichiers. Matlab possède deux types de fichiers. Le premier type est constitué par les fichiers créés à un format propriétaire à Matlab (.mat). Le second type regroupe les fichiers binaires et ASCII communs à tous les langages de programmation.

1.6.1 Fichier Matlab

save enregistre la totalité de l'espace de travail dans le fichier matlab.mat

save nom_fic.mat var1 var2 ... permet de sauvegarder des variables (au format numérique ou texte, des variables réelles ou complexes) comme des matrices sous le système de fichiers propres à Matlab.

save nom_fic.mat var ... -ascii sauvegarde une matrice `var` au format ASCII (.txt)

load nom_fic permet la relecture des variables sauvegardées avec la commande `save` (.mat ou .txt)

1.6.2 Fichiers binaires et ASCII

Ce type de fichiers permet de communiquer avec l'extérieur et en particulier de rapatrier des données collectées par d'autres logiciels. Les commandes nécessaires à leur manipulation sont les suivantes :

fopen ouvre et donne des informations sur le fichier que l'on désire manipuler.

fwrite permet l'écriture en format binaire de données contenues dans une matrice.

fread permet la lecture d'un fichier de données binaires.

fprintf permet d'écrire une chaîne de caractère ASCII (avec caractères de formatage) dans le fichier.

fscanf permet la lecture de données ASCII

fclose ferme un ou de plusieurs fichiers

1.7 PRTools et DRTools

Installation des PRTools et DRTools :

- Téléchargez les boîte à outils sur les site :
  - PRTools : `http://www.prtools.org/`
  - DRTools : `http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html`
- Décompressez les fichiers dans leur répertoire respectif.
- Faites `File` → `Set Path ...` et ajoutez les répertoires et sous-répertoires de PRTools et DRTools dans les répertoires connus de Matlab à l'aide du bouton *Add with Subfolders...*. La commande `addpath` peut également être utilisée, mais il faut s'assurer que les répertoires de PRTools et DRTools soient en haut des répertoires connus de Matlab. Ceci donne priorité au fonctions de PRTools et DRTools en cas de conflit avec d'autres boîtes à outils.
- Testez l'installation à l'aide des commandes `help prtools` et `help drtool`.

Une fois PRTools et DRTools installées, téléchargez le script exp1_data.m à partir du site du sys828 dans votre répertoire de travail et excutez le script pour vous familiariser avec les commandes de bases. Au besoin, référez vous au manuel d'utilisation.

1.8 Ressources Utiles (en anglais)

- Matlab
  - Matlab Tutorial by the University of New Hampshire : `http://www.cyclismo.org/tutorial/matlab/`
  - Getting started in Matlab (from the Mathworks) : `http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf`
  - Matlab central (repository for exchanging matlab softwares) : `http://www.mathworks.com/matlabcentral/`
- PRTools
  - PRTools manual : `http://prtools.org/files/PRTools4.1.pdf`