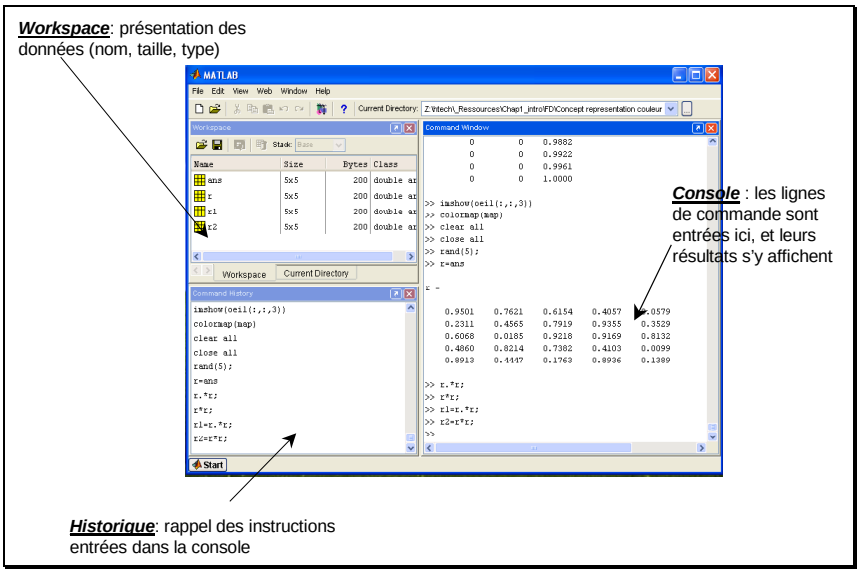
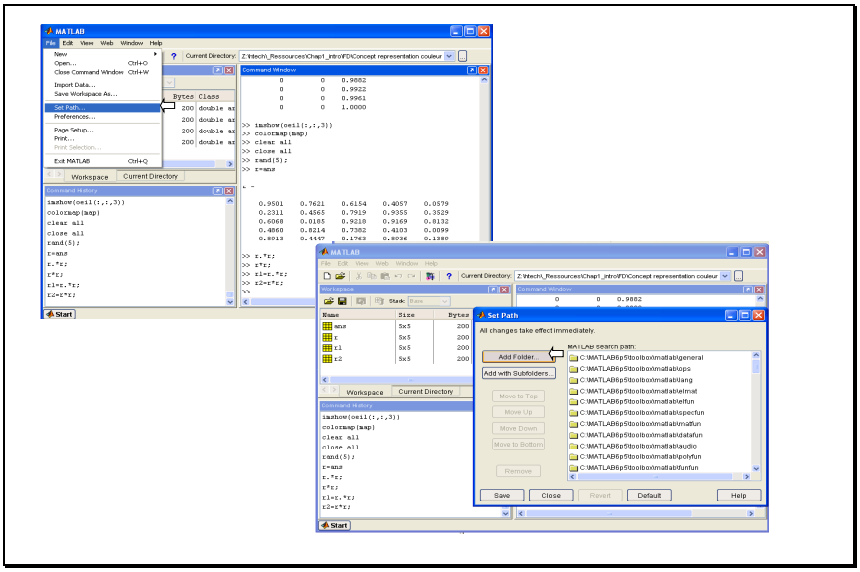
Exercice Chapitre 1– Prise en main du logiciel Matlab

MATLAB est un langage de calcul scientifique de haut niveau et un environnement interactif pour le développement d'algorithmes, la visualisation et l'analyse de données, ou encore le calcul numérique.

Lorsque vous lancez Matlab, vous disposez par défaut d'une console, d'un espace de travail (workspace) et d'un historique de commandes. Un éditeur est également accessible par la simple commande ***edit***.



ACTION : Lancez Matlab et ajoutez à la liste des chemins dans le navigateur de chemin (path browser) vos propres répertoires de travail (ne pas oublier d'ajouter les sous-répertoires). Les commandes Unix ***pwd, cd, ls*** sont disponibles.



*Remarque : lorsqu'un utilisateur lance une commande dans la console, Matlab va chercher dans le répertoire - que vous avez indiqué à travers le path browser - si une **fonction** ou un **script** correspond à cette commande, c'est alors le premier trouvé qui est utilisé (attention donc à l'ordre des répertoires et aux noms des scripts et des fonctions).*

Exercice 1 – Les opérateurs
Entre chaque exercice, il est conseillé de lancer les commandes ***clear*** et ***close all*** de manière à vider le workspace et à fermer toutes les figures.

1.1 – Entrez la commande *a=[1 4 5 ;4 8 9]*, à quoi cette commande correspond elle ?
Entrez dans la console la commande *a=rand(5)*. Puis entrez la commande ***help rand*** pour obtenir une description de l'opération effectuée par la fonction ***rand***. Enfin entrez la commande : *a=rand(5)* suivie d'un point virgule « *; »*
Quelle différence observe t'on dans la console avec la commande précédente *a=rand(5)* ? En déduire le rôle sous Matlab du « ; ».

1.2 – Sous Matlab, l'opérateur « : » est très utile. Il permet entre autres de balayer les éléments d'une ligne ou d'une colonne d'une matrice.
Mise en garde : L'indice 0 en Matlab n'existe pas. Le premier élément d'une matrice s'accède par l'indice 1. Par exemple, *matrice(1,1)* permet pour les images d'accéder à la valeur du pixel (1ère ligne, 1ère colonne). Le premier indice est pour les lignes, le second indice pour les colonnes.
Afin de bien assimiler ces concepts essayez les commandes :

- *a(2,1)*
- *a(1:3,1)*
- *a(:,2)*
- *a(1, :)*
- *a([1 4], :)*
- *a(1 :2 :5, :)* (la commande *1:2:5* permet de balayer l'intervalle [1, 5] par pas de 2)

Attention cependant à ne pas mettre de « ; » en fin de ligne pour observer les résultats obtenus dans la console.

1.3 – Matlab est un outil très intéressant pour le calcul matriciel. Les différentes opérations matricielles classiques (addition, multiplication, valeurs propres, …) y sont bien sur présentes, mais des opérations « élément par élément » très utiles en traitement d'images sont également disponibles par l'intermédiaire d'un opérateur pointé (exemple : « ***.*** », « ***./*** »).
Entrez les commandes suivantes (sans « ; » en fin de ligne pour observer les résultats) :

- *a = [0 0 0; 1 1 1; 2 2 2]*
- *b = a + 4*
- *c = a * b*
- *e = a .* b*

Expliquez la différence entre « ***c*** » et « ***e*** ».

ACTION : Créez une matrice *A* de taille 4×4 selon la méthode de votre choix (utilisez ***rand*** ou entrez les éléments un à un). Comment accéder à :

- la première ligne de *A* ?
- la quatrième colonne de *A* ?
- les trois premiers éléments de la quatrième ligne de *A* ?

Exercice 2 – L'affichage

Matlab offre aussi des possibilités d'affichage très diverses et pratiques. Il est par exemple possible de tracer des fonctions sur une échelle logarithmique ou de visualiser en image les valeurs d'une matrice.

2.1 – L'affichage d'un vecteur se fait par l'intermédiaire de la commande ***plot***. Entrez la commande ***help plot*** pour obtenir plus d'informations sur cette fonction et sur les fonctions qui ont des propriétés similaires (***subplot***, ***semilogx***, …).
Essayez les commandes suivantes :

- *x=1:3:10*
- *plot(x)* puis *plot(x,'r')*
- *y=rand(4,1)*, puis *plot(y)*, puis *plot(x,y,'g')*. Interprétez la différence de ces deux commandes.

La fonction ***plot*** est donc très utile pour obtenir par exemple les courbes de différentes fonctions planes.

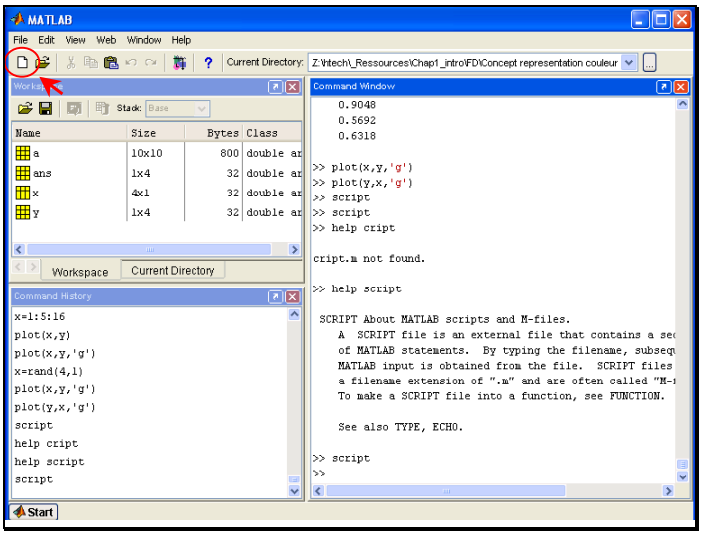
2.2 – L'affichage d'une matrice correspond lui à celui d'une image. Chaque élément (m, n) de la matrice est considéré comme étant la valeur du pixel (m, n) auquel il est associé. Vérifiez ceci en entrant les commandes suivantes :

- *a=rand(10)*10 ;* (pour que les éléments ne soient plus bornés à [0,1]),
- *a=exp(a) ;* (pour obtenir de plus grands écarts entre les éléments du vecteur a),
- *image(a)*

L'affichage des images peut se faire avec les fonctions ***image***, ***imagesc***, et ***imshow***. Essayez de zoomer à l'aide de la souris, quel type d'interpolation est utilisé ?

Exercice 3 – Ecriture de scripts

L'extension classique d'un fichier MATLAB est .m. On peut trouver 2 types de fichiers m : les fichiers de fonctions (abordés plus loin dans cet exercice) et les fichiers de script qui sont un ensemble de commandes pouvant s'enchaîner. L'utilisation de fichiers script permet de sauvegarder vos commandes d'une session Matlab à une autre. Pour ouvrir un fichier script :
- soit vous tapez la commande ***edit***,
- soit vous cliquez : file ⇒ new ⇒ M-file,
- soit vous cliquez directement sur l'icône représentant une page blanche.



Pour exécuter un script :

- soit vous lancez dans la fenêtre de commande la commande ***nom_du_fichier*** (sans l'extension .m), en vous assurant que la liste des chemins est cohérente ;
- soit vous sélectionnez des lignes du fichier .m dans la fenêtre d'édition et vous tapez la touche F9 (pratique pour exécuter une partie isolée du script).

ACTION : Entrez l'ensemble des commandes de la partie 1.2 dans un script que vous sauvegardez (exemple : ***test.m***). Lancez le script dans la console et par l'intermédiaire de la touche F9.

Exercice 4 – Type de données

Durant les séances d'exercices, nous serons amenés à utiliser la **toolbox *Image Processing***. Celle-ci contient un grand nombre de fonctions déjà écrites (n'hésitez pas à consulter l'aide via les commandes ***help*** ou ***helpwin***) ainsi que des démonstrations. Tantôt, nous écrirons nos propres fonctions tantôt nous utiliserons celles de la toolbox. Néanmoins, il faudra faire attention au type des données. En effet, classiquement et par défaut sous Matlab, tout est matrice de ***double***, or la plupart des fonctions de la toolbox *Image Processing* travaille avec le type ***uint8*** pour représenter les valeurs des pixels. Il faudra donc procéder au trans-typage chaque fois que cela est nécessaire (vous pouvez facilement voir le type de vos données dans le workspace).

ACTION : De la banque d'images, téléchargez l'image '*FRUIT_LUMI*' dans votre répertoire de travail. Lisez et affichez le fichier image respectivement à l'aide des commandes ***imread*** et ***imshow*** :

```
fruit = imread('FRUIT_LUMI.bmp');
imshow(fruit);
```

En consultant le workspace, observez la taille et le type de la donnée. De manière à afficher une sous-image correspondant au coin haut gauche de 64×64 pixels, testez la commande :

```
imshow(fruit(1:64,1:64));
```

La matrice *fruit* représente maintenant l'image '*FRUIT_LUMI*'. Essayez d'additionner directement cette matrice avec un nombre quelconque (ex : *fruit+8*). La console retourne le message d'erreur :

Function '+' is not defined for values of class 'uint8'

En effet, l'opérateur '+' est défini uniquement pour des éléments de type « double ». Pour effectuer cette opération, il est donc nécessaire de forcer le type « double » des éléments de la matrice en tapant : *double(fruit)*. Réessayez après ce transtypage.

Exercice 5 – Écriture de vos fonctions

Les fichiers de fonction sont à utiliser comme en programmation impérative classique. Ce sont également des *fichiers.m*. Pour utiliser une fonction il faut l'appeler avec des paramètres d'appels soit dans un script soit dans une autre fonction.

ACTION : Utilisez le fichier ***template.m*** pour écrire votre propre fonction ***max_min*** qui effectue la différence entre le plus grand et le plus petit élément d'une matrice. On pourra utiliser les fonctions Matlab ***max*** et ***min*** **.** Attention, il est nécessaire d'enregistrer le nom de votre fonction (exemple ***max_min*** **)** dans un fichier de même nom (exemple ***max_min.m*** **) .**

Corps du fichier ***template.m*** :

```
function[out1, out2] = template(arg1, arg2)

%-----------------------------------------
%
%    Description:
%
%    Entree :
%          arg1 :
%          arg2 :
%
%    Sortie :
%          out1 :
%          out2 :
%
%    Date :
%    Modif:
%-----------------------------------------
```

Remarque : L'utilisation d'une fonction peut s'effectuer directement au sein de la fenêtre de commandes (sous réserve des bons chemins !) ou encore au sein d'un script ou d'une autre fonction.

Correction de l'exercice sur la prise en main Matlab

Cet exercice a pour premier objectif de vous familiariser avec Matlab dans le cas d'une première prise en main, et de rappeler les concepts fondamentaux du logiciel Matlab à tous ceux qui l'avaient déjà utilisé.

1 – Les opérateurs

1.2 –

La commande *a=[1 4 5 ;4 8 9]* retourne la matrice 2×3 : $\begin{pmatrix} 1 & 4 & 5 \\ 1 & 8 & 9 \end{pmatrix}$.

La commande *a=rand(5)* retourne une matrice 5×5 composée de valeurs aléatoires comprises entre 0 et 1.
Enfin sous Matlab l'opérateur « ; » permet de ne pas afficher le résultat d'une commande dans la console. Cet opérateur est utile par exemple lorsqu'on travaille avec de grandes matrices (comme les images) pour lesquelles l'affichage est long et souvent peu représentatif de la donnée.

1.3 –

Manipulation de l'opérateur « : ».

1.4 –

Sous Matlab deux types d'opérations sur les matrices existent avec les opérateurs « * » et « / » :

- multiplication et division matricielle,
- multiplication et division élément par élément (utilisation conjointe avec l'opérateur « . ».

La commande *c=a*b* fera la multiplication matricielle de la matrice « a » par la matrice « b » : $c_{ij}=\sum_k a_{ik}.b_{kj}$.

La commande *e=a.*b* fera la multiplication élément par élément de la matrice « a » par la matrice « b » : $e_{ij}=a_{ij}.b_{ij}$. Les matrices doivent donc être de même taille.

ACTION :

Créons une matrice *A* de taille 4×4 en entrant directement les coefficients un à un : A=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]

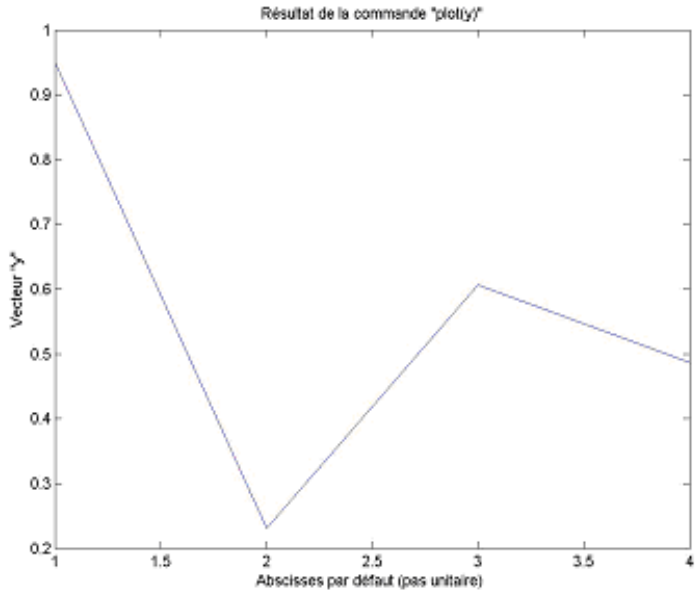
$$\mathbf{A}=\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

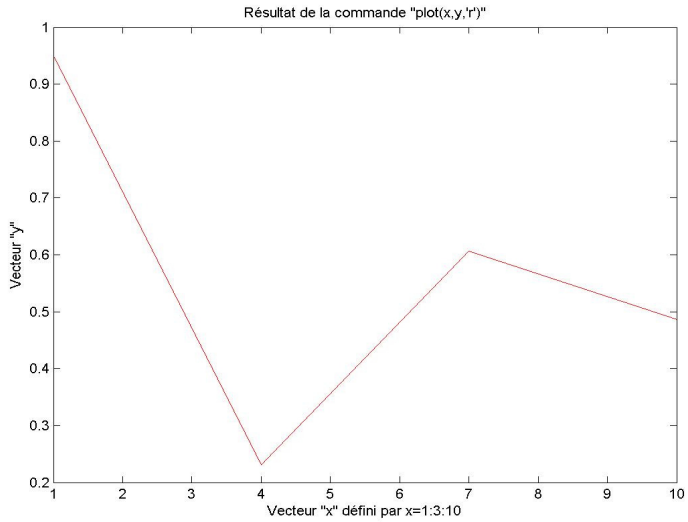
La 1ère ligne de A est donnée par la commande : *A(1,:)*.
La 4ème colonne de A est donnée par la commande : *A(:,4)*.
Les 3 premiers éléments de la 4ème ligne de A sont donnés par la commande : *A(4,1:3)*.

2 – L’affichage

2.1 –

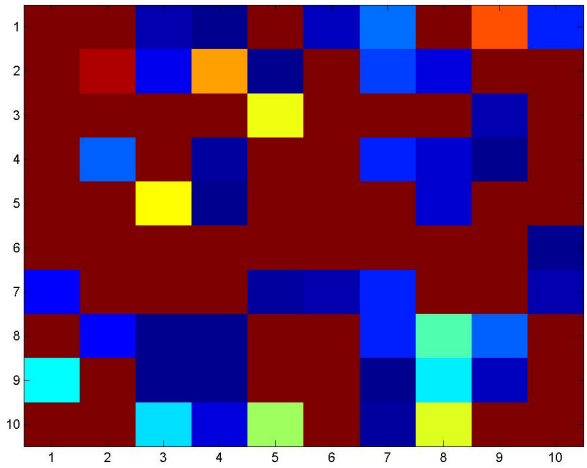
La commande *y=rand(4,1)* renvoie un vecteur de 4 éléments. En tapant *plot(y)*, une courbe apparaît. Cette courbe représente l’évolution du vecteur « y » en fonction des indices des éléments du vecteur. Il est cependant possible de modifier les abscisses en créant un vecteur d’abscisse « x » dans une unité voulue de même taille que le vecteur « y » : vous tracez y en fonction de x par la commande *plot(x,y)*.



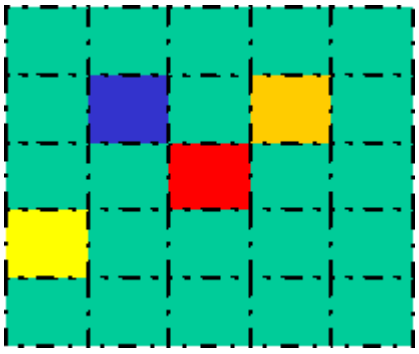


2.2 –

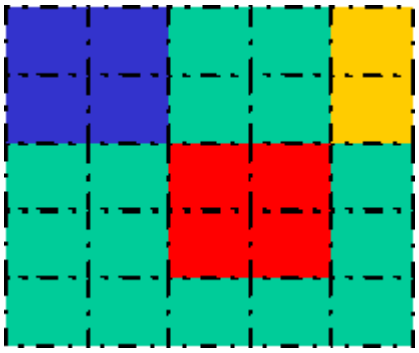
Voici un exemple de résultat obtenu en tapant les commandes indiquées avec un affichage par *image(a)*.



La matrice de taille 10×10 est ici représentée par 10×10 pixels carrés de couleurs différentes. Les couleurs correspondent aux valeurs des différents éléments de la matrice.
Le zoom utilise une interpolation d'ordre 0 dite du plus proche voisin. Il s'agit en fait d'une simple recopie d'un pixel image sur plusieurs pixels de l'écran d'affichage.
Considérons par exemple un écran de 5×5 pixels, sur lequel est représenté une image 5×5. Les pixels écran correspondent au quadrillage représenté.



La figure ci dessous présente un zoom ×2 autour du pixel rouge situé au centre dans le cas d'une interpolation du plus proche voisin.



Le pixel rouge est simplement « dupliqué » deux fois en hauteur et en largeur sur les pixels écran.

3 – Ecriture de scripts

Manipulation d’un script.

4 – Type de données

Manipulation de données uint8.

5 – Écriture de vos fonctions

La fonction « ***max*** » de matlab (resp. la fonction « ***min*** ») retourne pour une matrice M×N en entrée, un vecteur de taille N dont chaque élément e_k est l’élément maximal (resp. l’élément minimal) sur la colonne k de la matrice.

Voici la fonction solution :

```
function[out] = max_min(A)

%-------------------------------------------
%     Description: Différence des éléments max et min
%     d'une matrice A correspondant à une image monochrome
%
%     Entree :
%           A : la matrice sur laquelle s'effectue
%           la recherche
%
%     Sortie :
%           out : la valeur de la différence
%-------------------------------------------

out = max(max(A))-min(min(A));
```