Module : Traitement d'images
Filière : ING-4-GLSI
Année universitaire : 2024-2025

TP 0 : INITIATION À MATLAB

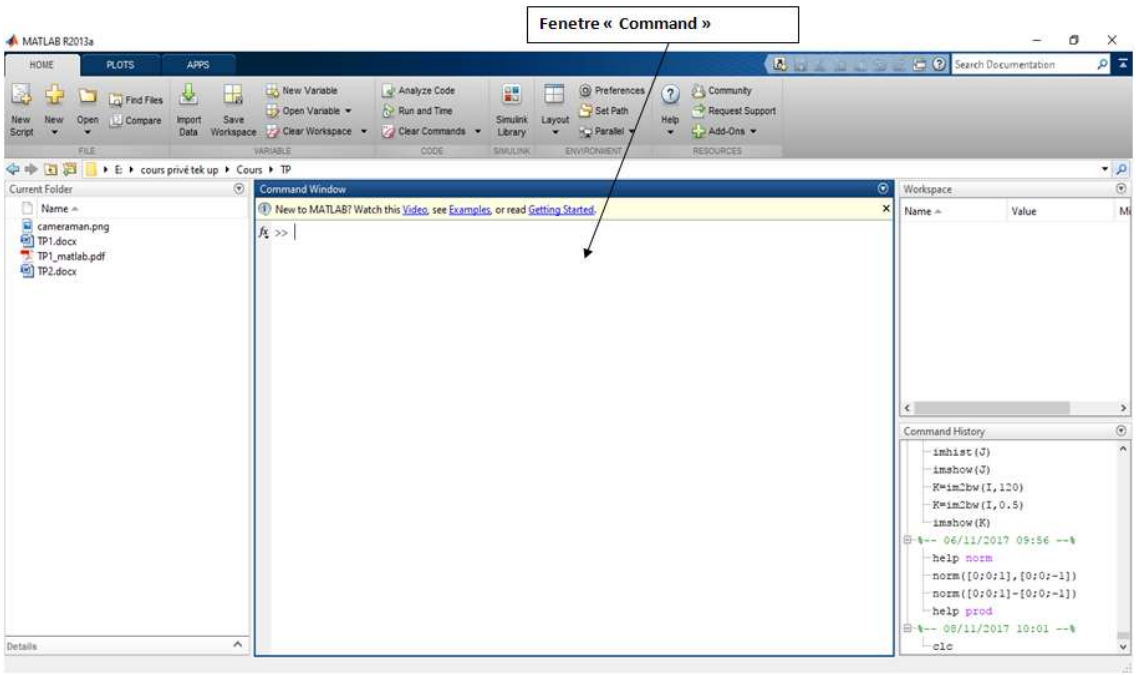
1. Introduction

Matlab (abréviation de " **MAT**rix **LAB**oratory ") est un puissant logiciel de calcul numérique. C'est un environnement informatique conçu pour le calcul matriciel. L'élément de base est une matrice dont la dimension n'a pas à être fixée. Matlab est un outil puissant qui permet la résolution de nombreux problèmes en beaucoup moins de temps qu'il n'en faudrait pour les formuler en C ou en Pascal. La programmation au même titre que C, Pascal ou Basic. Par contre, sa singularité est qu'il s'agit d'un langage interprété, c'est-à-dire que les instructions sont exécutées immédiatement après avoir été tapées.

Le but de ce TP est de vous familiariser avec l'utilisation de MATLAB.

1.1. Lancement

Au lancement de Matlab, la fenêtre "command" apparaît. C'est dans cette fenêtre que l'on peut taper les instructions Matlab (à la suite des chevrons >>).



1.2. Langage interprété

Puisque Matlab est un langage interprété. Il n'est pas nécessaire de compiler un programme avant de l'exécuter. Toute commande tapée dans la fenêtre de commande est immédiatement exécutée.

```
>> 2+2
ans = 4
>> 5^2
ans = 25
```

La réponse est affichée si on ne met pas de point-virgule en fin de ligne. Elle est de plus stockée dans une variable nommée **ans** (**ans**wer). La plupart des fonctions mathématiques usuelles sont définies dans Matlab, et ceci sous une forme naturelle (sin, cos, exp, ...). C'est également le cas de certaines constantes comme pi par exemple :

```
>> 2*sin(pi/4)
ans = 1.4142
```

1.3. Les variables

On peut évidemment indiquer le nom de la variable dans laquelle le résultat doit être stocké (ce nom doit commencer par une lettre).

Remarque : Matlab distingue les minuscules des majuscules.

```
>> x = pi/4
x = 0.7854
```

Le nom de la variable ainsi que le résultat sont alors affichés. On peut taper plusieurs commandes par ligne si elles sont séparées par un point virgule.

```
>> x = pi/2; y = sin(x);
```

1.4. Les vecteurs, les matrices et leur manipulation

En fait, toute variable de Matlab est une matrice (scalaire : matrice 1x1, vecteur : matrice 1xN ou Nx1). On peut spécifier directement une matrice sous la forme d'un tableau avec des crochets, l'espace ou la virgule sépare deux éléments d'une même ligne, les points virgules séparent les éléments de lignes distinctes.

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

Les éléments d'une matrice peuvent être n'importe quelle expression de Matlab :

```
>> x = [ -1.3, sqrt(3), (1+2+3)*4/5 ]
x = -1.3000 1.7321 4.8000
```

Les éléments d'une matrice peuvent ensuite être référencés par leurs indices, on utilise alors des parenthèses et non des crochets. Le mot-clé **end** peut être utilisé en indice pour signifier le dernier élément.

```
>>x(2)
ans = 1.7321
>>x(5) = abs(x(1))
x = -1.3000 1.7321 4.8000 0.0000 1.3000
>> x(end-1)
ans =0
```

On peut remarquer que la taille du vecteur x a été ajustée en remplissant les éléments non précisés par 0. On peut aussi créer des matrices avec les fonctions zeros, ones et eye, ces fonctions créent des matrices de la taille précisée, respectivement remplies de zéros, de un, et de un sur la diagonale et de zéros ailleurs (eye =prononciation anglaise de I comme identité).

```
>> eye(2,3)
ans = 1 0 0
 0 1 0
>> ones(1,5)
ans = 1 1 1 1 1
```

On peut avoir des informations sur la taille d'une matrice :

```
>> size(x)
ans = 1 5
>> length(x)
ans = 5
```

On peut ajouter des lignes et des colonnes à des matrices déjà existantes (attention, les dimensions doivent toutefois être compatibles ...).

```
>> r1 = [10, 11, 12];
>> A = [A ; r1]
A =  1  2  3
     4  5  6
     7  8  9
     10 11 12
>> r2 = zeros(4,1);
>> A = [A, r2]
A =   1   2  3  0
      4   5  6  0
      7   8  9  0
      10 11 12  0
```

1.5. L'opérateur " : "

L'opérateur " : ", sous Matlab, peut être considéré comme l'opérateur d'énumération. Sa syntaxe usuelle est :« deb:pas:fin ». Il construit un vecteur dont le premier élément est deb puis deb+pas, deb+2*pas... jusqu'à deb+n*pas tel que deb+n*pas < fin < deb+(n+1)*pas. « : » est très utile pour construire des signaux.

```
>> x = 0.5:0.1:0.85
x = 0.5000 0.6000 0.7000 0.8000
```

Le pas d'incrémentation peut être omis, un pas de 1 est alors pris par défaut :

```
>> x =1:5
x = 1 2 3 4 5
```

On peut aussi utiliser le " : " pour sélectionner des éléments d'un vecteur ou d'une matrice :

```
>> A(1,3)   % Troisième élément de la première ligne de A
>> A(1,1:3) % Premier, deuxième et troisième éléments de
            % la première ligne de A
>> A(1,:)   % Tous les éléments de la première ligne
>> A(:,3)   % Tous les éléments de la troisième colonne
>> A(:)     % Vecteur colonne contenant tous les éléments
            % de A lus colonne par colonne.
```

1.6. Opérations matricielles

Les opérations usuelles sont définies de façon naturelle pour les matrices :

```
>> 2*A % Produit par un scalaire
>> A*B % Produit de deux matrices (de dimensions cohérentes)
>> A^p % Elève la matrice carrée A à la puissance p
>> inv(A) % Inversion d'une matrice carrée inversible (message d'alerte éventuel)
>> A.*B % Produit élément par élément de deux matrices / IMPORTANT
>> X = A\B % Donne la solution de A*X = B (équivalent à X = inv(A)*B)
>> X = B/A % Donne la solution de X*A = B (équivalent à X = B*inv(A))
>> X = A./B % Division éléments par éléments
```

Autres fonctions utiles pour les opérations matricielles

```
>> max(A)
```

renverra un vecteur ligne dont chaque élément sera le maximum de la colonne correspondante. De la même façon est définie la fonction min().

2. Affichages graphiques et alphanumériques

2.1. Affichage alphanumérique

On peut afficher des chaînes de caractères dans la fenêtre de commande :

```
>> message = 'bienvenue sur Matlab';
>> disp(message)
bienvenue sur Matlab
```

Les fonctions sprintf et fprintf existent également

```
>> fprintf('pi vaut %f\n',pi)
pi vaut 3.141593
```

On peut aussi demander des valeurs à l'utilisateur :

```
>> rep = input('Nombre d''itération de l''algorithme : ');
```

Matlab affichera la chaîne de caractère entrée en paramètre et attendra une réponse de l'utilisateur.

2.2. Affichages graphiques de courbes 1D

Matlab permet un grand nombre de types d'affichage 1D et 2D, seuls les plus courant seront décrits ici. La commande plot permet l'affichage d'une courbe 1D :

```
>> x = 0:0.1:2; y = sin(x*pi);
>> plot(x*pi,y) % plot(abscisse,ordonnée)
```

On peut ajouter un titre aux figures ainsi que des labels aux axes avec les commandes title, xlabel, ylabel:

```
>> title('Courbe y = sinus(pi*x)')
>> xlabel('x'); ylabel('y')
```

2.3. Affichage de plusieurs courbes

On peut bien évidemment vouloir afficher plusieurs courbes à l'écran. Pour cela deux solutions s'offrent à nous :On peut effectuer plusieurs affichages sur une même figure en utilisant la commande subplot qui subdivise la enêtre graphique en plusieurs sous figures. Sa syntaxe est :

«subplot(nombre_lignes,nombre_colonnes,numéro_subdivision)».Les subdivisions sont numérotés de 1 à nombre_lignes*nombre_colonnes, de la gauche vers la droite puis de haut en bas.

```
>> subplot(3,2,1)
>> plot(x,y)
>> subplot(3,2,2)
>> plot(x,y.^2)
```

On peut aussi ouvrir une deuxième fenêtre graphique à l'aide de la commande figure. Le passage d'une fenêtre graphique à une autre pourra alors se faire à la souris. **NB** : on peut également superposer plusieurs courbes sur le même référentiel, en utilisant la commande hold on.

3. Environnement de travail, scripts et fonctions

3.1. Répertoire de travail

Il est indispensable de travailler dans votre propre répertoire et non dans le répertoire de Matlab. De même dans votre répertoire vous pouvez organiser vos fichiers dans des sous répertoires

3.2. Scripts

Il est parfois (souvent) souhaitable, pour ne pas avoir à taper plusieurs fois une même séquence d'instructions, de la stocker dans un fichier. Ainsi on pourra réutiliser cette séquence dans une autre session de travail. Un tel fichier est dénommé script. Sous windows, il suffit d'ouvrir un fichier avec le menu file, new, de taper la séquence de commande et de sauver le fichier avec une extension " .m " (nom_fich.m). En tapant le nom du fichier sous Matlab, la suite d'instructions s'exécute. Les variables définies dans l'espace de travail sont accessibles pour le

script. De même les variables définies (ou modifiées) dans le script sont accessibles dans l'espace de travail. On peut (doit) mettre des commentaires à l'aide du caractère pour-cent " % ". Toute commande située après " % " n'est pas prise en compte par Matlab, jusqu'à la ligne suivante.

3.3. Fonctions

On a parfois (souvent) besoin de fonctions qui ne sont pas fournies par Matlab. On peut alors créer de telles fonctions dans un fichier séparé et les appeler de la même façon que les fonctions préexistantes. La première ligne (hormis les lignes de commentaires) d'une fonction doit impérativement avoir la syntaxe suivante :

```
function [ var de sorties, ...] = nom_fonction( var d'entrée, ...)
```

Exemple de fonction :

```
function y = sinuscardinal(x)
z = sin(x); % Variable de stockage
y = z./x; % Résultat de la fonction % Il faudra se méfier de la division pour x=0
```

Cette fonction pourra être appelée par :

```
>> sincpi = sinuscardinal(pi);
```

Exemple de fonction à plusieurs variables de sortie :

```
function [mini, maxi] = minetmax(x)
mini = min(x); % Première variable de sortie
maxi = max(x); % Deuxième variable de sortie
```

Cette fonction pourra être appelée par :

```
>> [miny, maxy] = minetmax(y);
```

Le nom de la fonction doit impérativement être le même que le nom du fichier dans lequel elle est stockée (sinon Matlab ne prendra pas en compte ce nom mais uniquement celui du fichier). Les nombres d'arguments en entrée et en sortie ne sont

pas fixes. Cela nous permet donc de faire des fonctions Matlab pouvant dépendre d'un nombre variable de paramètres. Les variables de l'espace de travail ne sont pas accessibles à la fonction sauf si elles sont entrées comme variables d'entrée. De même les variables définies dans une fonction ne sont pas accessibles dans l'espace de travail.

4. Boucles et contrôles

Comme de nombreux autres langages de programmation, Matlab possède trois types d'instructions de contrôles et de boucles : for, if et while.

4.1. Contrôle : " if "

```
« if (expression logique) »
sute d'instructions 1;
else
suite d'instructions 2;
end
```

La condition est exprimée au travers d'une expression logique. Les expressions logiques sont des expressions quelconques pour lesquelles on considère uniquement le fait qu'elles soient nulles (expression fausse) ou non (expression vraie). On peut aussi construire ces expressions à l'aide des opérateurs logiques et (&), ou (|), égal (==), supérieur (>,>=), inférieur(<,<=), non.

4.2. Boucle : « while »

```
while (expression logique)
suite d'instructions
end
```

La suite d'instructions est répétée un nombre indéfini de fois jusqu'à ce que l'expression logique soit fausse.

4.3. Boucle : "for"

La boucle for a pour syntaxe :

```
for i=1:n
suite d'instructions;
end
```

Cette structure de contrôle est très utile, mais il est recommandé d'éviter au maximum les boucles dans Matlab, car elles sont très lentes par rapport à un traitement matriciel. Par exemple, la boucle suivante :

```
for i=1:N, x(i) = i; end
```