TP1 : Introduction à Matlab

OBJECTIF : L'objectif principal de ces 3 heures de TPs est de vous apprendre les bases en langage Matlab. Vous avez à votre disposition ce document ainsi qu'un feuillet « Mémo Matlab ». Il vous sera demandé un certain nombre de réalisations montrant que vous maîtrisez ce nouvel outil. La section « Problèmes » est constituée d'exercices complémentaires, potentiellement obligatoires, destinés aux curieux et à arrondir la note au niveau de chacun.

1. Partie obligatoire

1.1. Ouverture, découverte et création de votre espace de travail (Mémo partie 1 et 3.1).

Exercice 1.

1. Prenez connaissance de la première partie du fichier mémo et essayez les différentes commandes recommandées.

2. Vous pouvez constater que la commande « *intro* » effectue des opérations les unes après les autres sans vous donner plus de détails. Saisissez « *help intro* » dans votre terminal et cliquez sur « *show demo intro* ». Renseignez-vous sur les actions effectuées, si certaines opérations mathématiques vous sont inconnues (convolution, ...), ne paniquez pas, nous vous les introduirons en temps voulu.

3. Une fois l'exemple compris, vous allez créer votre répertoire de travail. Matlab possède un espace de travail par défaut [1], cependant il ne vous sera pas pratique de travailler que dans ce seul répertoire. Créez donc un nouveau dossier dans le répertoire courant et appelez le « *TP1* ». Vous avez le choix entre plusieurs méthodes :
   - Effectuer un clic droit dans la fenêtre « Current folder » et sélectionner « New Folder ».
   - Aller dans le dossier « $MATLAB$ » à partir du navigateur de fichier et créer le nouveau dossier manuellement.
   - Utiliser la bonne commande Matlab (partie 3 du mémo)

4. Copiez le fichier *patronTP.m* dans le dossier « *TP1* », renommez le « *mainScriptTP1* », puis retournez sur Matlab. Lancez la commande « *mainScriptTP1* » dans le terminal Matlab. Que constatez-vous ?

5. Si l'erreur « *undefined function or variable 'mainScriptTP1'* » s'affiche, c'est parce que votre dossier« *TP1* » ne se trouve pas dans le chemin d'accès de Matlab (*path*). Pour l'ajouter, vous avez le choix entre deux actions :
   - Faire un clic droit sur le dossier en question et sélectionnez « *Add to Path* ».

1. situé dans « $\backslash Documents \backslash MATLAB$ »

— Exécuter la commande « *addpath TP1* » ou « *path = (path, 'TP1')* »

Votre dossier est à présent ajouté dans l'ensemble des dossiers consultables par Matlab. Pour vérifier cela retenter de lancer la commande « *mainScriptTP1* » [2].

Attention : Les modifications des chemins d'accès ne sont pas automatiquement sauvegardées dans Matlab. Pour sauvegarder vos modifications, tapez la commande « *savepath* », ainsi vous n'aurez pas à rajouter le dossier « *TP1* » à chaque nouvelle session. L'ensemble de ces commandes peuvent se retrouver dans la partie 3 du mémo.

1.2. Types, variables et calculs basiques.

Matlab ou *Matrix Laboratory* ne connaît qu'un seul type d'objet : les matrices. Les scalaires sont des matrices 1×1, les vecteurs colonnes des matrices $n \times 1$ et les vecteurs lignes sont des matrices $1 \times n$. Une matrice en Matlab ne peut être constitué que d'un seul type de données.

1.2.1. Les types de données

Matlab est composé de 16 classes fondamentales qui vont définir les types de données disponibles :

— La classe numérique : permet de construire les réels et les entiers avec un certain niveau de précision.
  — Les entiers (8 classes) : int* et uint*
  — Les réels (2 classes) : single (32-bits), double (64-bits)
— La classe logique : permet de représenter les états vrais et faux en utilisant les nombres 1 et 0 respectivement.
— La classe caractère : permet de stocker des caractères ou des chaînes de caractères.
— Les classes table, cell, struct et function handle : qui ne vous seront pas présentées dans ce module.

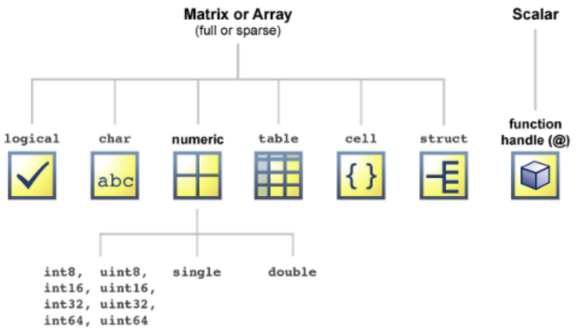

FIGURE 1 – Diagramme représentant l'organisation des classes en Matlab - tiré de la documentation officielle MathWorks® [3]

2. **NB :** Il est possible de taper sur '↑' dans le terminal pour retrouver une ancienne commande.

3. `https://fr.mathworks.com/help/matlab/matlab_prog/fundamental-matlab-classes.html`

1.2.2. Les variables (Mémo partie 3.5 et 4).

1. Première variable
   - Dans le terminal Matlab, saisissez la commande « *11* » et appuyez sur « *Entrée* », que constatez-vous ? Quel nouvel élément est apparu dans la fenêtre en haut à droite ?
   - Dans le terminal saisissez « *ans* ». Saisissez ensuite la commande « *7* » puis de nouveau « *ans* ». Que constatez-vous ?
   - « *ans* » ou « *answer* » est une variable par défaut de Matlab qui retourne le dernier résultat. Cette variable n'est donc pas pratique pour effectuer de plus grands calculs, il va falloir créer les vôtres.
   - Saisissez dans le terminal « *a=11* » et remarquez les changements dans la fenêtre en haut à droite. Vous venez de créer votre première variable. Recommencez avec une variable « *b* » mais cette fois-ci ajoutez un semi-colon « ; » à la fin de l'opération. Que constatez-vous ?
2. Construction explicite vs rapide
   - Dans le terminal saisissez la variable « *x = [1,2,3,4,5]* », de quel type de vecteur s'agit-il ? Créez la variable « *y* » en remplaçant les « , » par des « ; ». Que constatez-vous ?
   - À présent, saisissez « *x1 = [1 : 5]* » et « *y2 = [1 : 5]'* » . Que peut-on en déduire ?
3. Construction de matrices :
   - Saisissez la commande « *A = [1 2 3 ; 2 3 1 ; 3 1 2]* ». Vous venez de créer une matrice de manière explicite.
   - Saisissez la commande « *A2 = [A ; A]* ». Puis « *A3 = [A ; 2 3]* ». Que constatez-vous ? Que pouvez-vous en déduire sur la construction de matrice en générale ? Cela vous semble-t-il logique ?
4. Index :
   - Pour accéder aux valeurs présentes dans les vecteurs, vous devez saisir « *x(i)* » avec i la valeur de l'index. Pour les matrices, vous devez saisir « *A(i, j)* » avec i l'index de la ligne et j l'index de la colonne. Ainsi, si « *B = [1, 2, 3 ; 4, 5, 6]* », taper « *B(2,2)* » retournera la valeur 5.
   - Vous pouvez accéder à plus de valeurs en fournissant des vecteurs comme valeurs d'index. Ainsi, « *B(1 : 2, 2 : 3)* » vous retournera la sous-matrice « *[2 3 ; 5 6]* ».
   - Entrainez-vous à manipuler les index en vous inspirant du mémo et répondez aux questions de l'exercice 2.

L'ensemble des réponses aux exercices sont à noter dans le fichier « *mainScriptTP1* » ! N'hésitez pas à donner des noms de variables significatifs, pour faciliter la correction.

Exercice 2.

1. À partir du mémo, proposez une commande qui nous permettrait d'obtenir un vecteur ligne « *x3* » contenant les nombres de 1 à 5 avec un pas de 0.25.
2. Simulez un carré magique de taille 5 et sélectionnez le nombre se trouvant dans la troisième colonne de la cinquième ligne.

3. Créez un vecteur ligne de longueur 50 contenant des coefficients suivant une loi normale de moyenne 0 et de variance 1.
4. À l'aide de la fonction « *repmat* », créez une matrice de taille 5×10 contenant que des 1 sur la première ligne, des 2 sur la deuxième, etc... N'hésitez pas à utiliser la fonction « *help* ».

1.2.3. Calculer avec Matlab (Mémo partie 3.2 et 5).

Matlab est comme une grosse calculatrice très couteuse, vous pouvez donc y effectuer les opérations élémentaires. La partie 3.2 du mémo, vous montre qu'il y a trois façons de calculer avec Matlab, tout dépend de ce que vous souhaitez faire. Ainsi, inutile de déclarer une variable si vous souhaitez juste connaître une valeur ou d'afficher un résultat interminable comme une matrice (15000×500). Nous vous invitons à lire cette session ainsi que la partie 5 consacrée aux fonctions avant de passer aux exercices.

Exercice 3.

1. Créez deux matrices de tailles 2×2 et un vecteur ligne de taille 2. Effectuez la multiplication entre ces deux matrices, puis entre une des matrice et le vecteur ligne. Multipliez terme à terme les deux matrices (n'oubliez pas à noter chaque opération dans le fichier « *mainScriptTP1* »).
2. Créez la matrice « *A4 = [2 1 -3 ; 1 -2 1]* » et calculez son rang, calculez A5 qui est égal à A4 multipliée par sa transposée, puis calculez le déterminant et la trace de A5. Refaites ces opérations pour la transposée de A4, que ce passe-t-il ?
3. Créez le vecteur « *xrand = rand(1,10)* » et calculez sa moyenne, son écart-type, sa médiane et triez le dans l'ordre croissant (une ligne par opération). Selon vous la valeur de la moyenne est-elle correcte ?

1.3. L'aide avec Matlab (Mémo partie 2).

Vous avez déjà vu l'utilité de la fonction « *help* » dans les parties précédentes. Sachez qu'il existe d'autres fonctions permettant de vous aider dans vos réalisations, tel que « *lookfor* » qui vous aidera à trouver des fonctions à l'aide d'un mot clé ou « *helpwin* » qui vous donnera la liste des bibliothèques installées.

1.4. Outils graphiques (Mémo partie 7).

Les résultats obtenus ne sont parfois pas directement exploitables. Vous allez avoir recours à une représentation graphique pour les analyser, les interpréter et conclure sur vos approches. Matlab fournit un ensemble d'outil pour réaliser ces graphiques : vous pouvez tracer les résultats dans une figure, ajouter des légendes, modifier les axes,...

Le but de cette partie est de vous donner un avant-goût des possibilités de Matlab en termes de rendu graphique.

Exercice 4.

1. Créez la variable « *zrand = 2*randn(1,8000) + 4* ».
2. À l'aide des commandes « *plot* », « *figure* », « *hist* » et éventuellement « *bar* » : afficher les caractéristiques de zrand sur deux figures différentes (N'hésitez pas à utiliser la fonction help pour comprendre le rôle de chacune de ces fonctions et d'augmenter le nombre de bar de l'histogramme à 100).
3. Lors de la question précédente, vous avez tracé l'histogramme de zrand. Quelles caractéristiques peut-on retirer de cette représentation d'un point de vue statistique ?
4. Utilisez la commande « *clf* » pour effacer l'ensemble des figures.
5. Ouvrez une nouvelle figure « *'figure(1)'* » et regardez la documentation de la fonction « *subplot* ».
6. Nous allons créer une figure contenant 2 sous-figures comme ci-dessous :

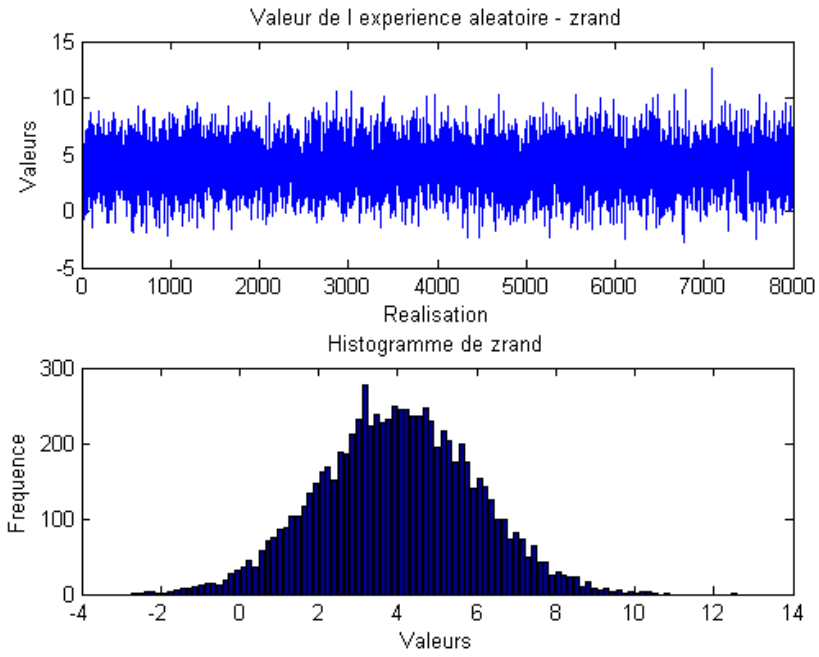

FIGURE 2 – Figure à réaliser

7. Premièrement : ouvrez une nouvelle figure, sélectionnez la première figure à l'aide de la fonction « *subplot* » et affichez « *zrand* »
8. Ensuite : sélectionnez la seconde figure et affichez-y l'histogramme de zrand (n'hésitez pas à conserver la valeur des centres des bars de l'histogramme pour un meilleur rendu visuel)
9. Utilisez les fonctions *title*, *xlabel*, *ylabel* pour obtenir la même figure que la figure à réaliser.

1.5. Programmer sous Matlab (Mémo partie 8).

Vous avez appris à déclarer des variables, à effectuer un certain nombre d'opérations mathématiques, à réaliser des figures à partir du terminal et utiliser des fonctions présentent dans le logiciel. Cette pratique présente quelques limites :

1. Si vous fermez Matlab, les résultats et les variables seront effacés ; ainsi, vous devrez tout recommencer depuis le début entre chaque session de TP.
2. Vous êtes limité par les implémentations des fonctions Matlab.

Ainsi, vous devez apprendre à créer des **scripts** pour automatiser certaines actions comme charger des données, créer des variables, effectuer certains calculs, ... et à créer des **fonctions** pour compléter les outils à votre disposition.

Exercice 5.

1. Prenez connaissance des points 8 et 9 du mémo ;
2. Créez une fonction « *xBin* » et « *pyramEtoile* » à l'aide de boucles *while* et *for*, tout en respectant les signatures suivantes :

```
function [ xBinaire ] = xBin( x )
%
% Fonction qui prend un entier x et le divise
% consecutivement par 2 pour obtenir son ecriture
    binaire
%
% * Entree :
%       -> x - Int - L'entier a binariser
%
% * Sortie :
%       -> xBinaire - [char] - Chaine de caracteres
%       representant l equivalent binaire de x
```

```
function [ pyramide ] = pyramEtoile( nombDeLigne )
%
% Fonction qui dessine une pyramide d'etoile (exemple
    pour 3):
%  *..
%  **.
%  ***
%
% * Entree :
%       -> nombDeLigne - Int - Un nombre de ligne a
    tracer
%
% * Sortie :
%       -> pyramide - [char] - nombDeLigne x
    nombDeLigne - la pyramide
```

3. Réalisez un script qui selon la valeur d'un chiffre aléatoire (« *rand* ») affiche pile ou face avec une probabilité de 0,5.

1.6. Préparer son compte rendu.

L'une des meilleures pratiques que l'on puisse avoir en programmation est celle de commenter son code. D'une part, pour vous souvenir des actions effectuées d'une séance de TP à l'autre. D'autre part pour ceux qui vont reprendre votre code : que ce soit les correcteurs dans le cadre de vos études ou les collègues dans le cadre professionnel. Généralement, la forme exacte est définie par l'entreprise dans le milieu professionnel. Dans le cadre de ce cours, nous vous proposons d'utiliser une forme qui vous convienne le mieux, mais qui doit contenir :

— Pour une fonction :
  1. Le rôle de la fonction
  2. Les types de paramètres entrant (entier, logique [booléens], chaine de caractères, ...), leur dimension et une brève description de leur rôle *Exemple : v - entier - N×1 - vecteur vitesse*
  3. Les types de paramètres sortant (entier, logique [booléens], chaine de caractères, ...), leur dimension et une brève description de leur rôle
  4. Votre nom, prénom et groupe

— Pour un script :
  1. Le rôle du script
  2. Votre nom, prénom et groupe
  3. Une date (éventuellement)

— Tout au long du code : des explications brèves lorsque vous employez une astuce pas forcément évidente à comprendre

Les commentaires en Matlab se font à l'aide du symbole %, vous avez le choix entre en mettre plusieurs ou un seul. [4]

2. Compte rendu

Dans un dossier compressé (.zip), vous enverrez à votre responsable :

— Une capture d'écran de votre répertoire de travail (Exercice 1) ;

— Votre fichier « *mainScriptTP1* » contenant les réponses aux exercices et problèmes ;

— Les fichiers de fonctions « *xBin* » et « *pyramEtoile* » ;

— Le script « *pile ou face* » de l'exercice 5.3 ;

— Un document pdf contenant vos constatations et vos commentaires sur la séance.

4. Lorsque vous mettez un seul symbole, il s'agit d'un commentaire. Lorsqu'il y en a deux : vous délimitez une portion de code, ces commentaires sont visibles depuis la fenêtre « détail » de Matlab.

3. Problèmes

Problème 1. Soit N = 1 :144, trouvez la fonction qui vous permettra de transformer ce vecteur ligne en une matrice 12×12. De cette matrice, proposez une méthode pour extraire les sous-matrice suivantes (en une ligne) :

- Une sous-matrice formée par les coefficients a_{ij} pour $i = 1, ..., 6$ et $j = 7, ..., 12$;
- Celle formée par les coefficients a_{ij} pour $(i, j) \in \{3, 4, 7, 8, 11, 12\}$;
- Celle formée par les coefficients a_{ij} pour i et j pairs.

Problème 2. Trouvez l'équivalent Matlab de la fonction « *xBin* » (Exercice 5.).

Problème 3. Créez la fonction « *pyramEtoile2* » qui combine la pyramide montante et la pyramide descendante.

Problème 4. Créez la fonction « *pyramEtoile3* » contenant une seule ligne et effectuant la même fonction que « *pyramEtoile* »

Problème 5. Créez la fonction « *pyramEtoile4* » contenant 3 lignes et effectuant la même fonction que « *pyramEtoile2* »

Problème 6. Réalisez un script demandant à l'utilisateur de saisir deux entiers et retournant le résultat de la somme en binaire.

Problème 7. Trouvez le type de donnée qui nous permettra de stocker dans un même vecteur les éléments suivants : 'chien', 42, [1 2 ; 3 4] et π jusqu'à sa 9-ième décimale (utilisez « *vpa* ») ? Quelle commande vous permet d'accéder au troisième élément de ce vecteur ? d'accéder à son contenu ?