Applications numériques
TP1 : Découverte de MatLab

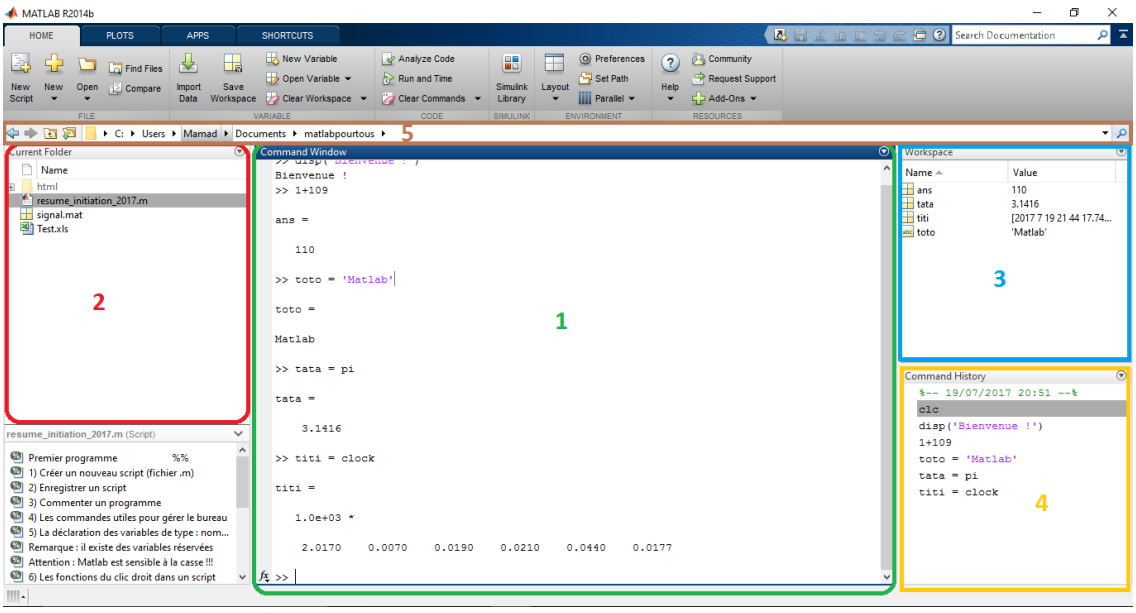
Septembre 2020

3iL INGENIEURS
</les sciences informatiques>

MatLab est un logiciel de calcul numérique, qui, comme son nom l'indique (**Mat**rix **Lab**oratory), est largement centré sur le calcul matriciel. Composé d'un langage de programmation (qui lui est propre) couplé à un environnement de travail complet, il nécessite une prise en main spécifique. Dans ce premier TP, qui fera aussi office de support de cours, l'objectif va donc être de vous familiariser avec cet outil, et d'en découvrir plusieurs fonctionnalités.

1 Premier contact

La fenêtre de MatLab est un IDE à part entière, c'est à dire qu'elle est aussi riche en contenu que difficile à appréhender lors de sa découverte. Voici donc à peu de choses près ce que vous devriez avoir sous les yeux lorsque vous lancez l'application :



L'interface principale se décompose en 5 grandes zones :

- Command window : C'est ici que vous entrerez vos premières instructions. C'est également ici que les résultats sont affichés (un peu à l'image d'une console)
- Current Directory : Vous montre le contenu du répertoire de travail courant. Cet espace deviendra très utile lorsque vous créerez vos premiers programmes
- Workspace : Liste de l'ensemble des variables connues par MatLab, ainsi que leur valeur. Pour obtenir des détails, il est possible de double cliquer sur une variable en particulier.
- Command History : Historique des dernières commandes que vous avez exécutées. Un simple glisser-déposer vous permet ainsi de réexécuter une commande passée.
- Répertoire courant : C'est le dossier dans lequel MatLab est actuellement en train de travailler. Lorsque vous créerez des programmes, il faudra bien faire en sorte de les enregistrer dans le répertoire courant, faute de quoi MatLab sera incapable de les exécuter.

2 Premières instructions

Lorsque vous travaillez dans MatLab, vous émettez des commandes qui affectent des variables et appellent des fonctions (préexistantes ou que vous avez vous-même écrites). Le langage de programmation étant interprété, il est possible d'exécuter les instructions à la volée, c'est à dire d'entrer une instruction, d'obtenir son résultat, puis d'entrer la suivante, et ainsi de suite. Dans la fenêtre de commande :

1. Ecrivez l'instruction $5 + 8$
   (a) Que se passe-il dans la fenêtre de commande ?
   (b) Que se passe-t-il dans le Workspace ?
2. Ecrivez l'instruction $x = pi/4$
   (a) Que se passe-il dans la fenêtre de commande ?
   (b) Que se passe-t-il dans le Workspace ?
3. Ecrivez l'instruction $x = pi/6$;
   (a) Que se passe-il dans la fenêtre de commande ?
   (b) Que se passe-t-il dans le Workspace ?
4. Quelles instructions vous permettent d'affecter aux variables $y1$ et $y2$ les valeurs de $sin(x)$ et $cos(x)$? Vous ferez en sorte que le résultat ne s'affiche pas dans la fenêtre de commande.
5. En utilisant les variables créées, quelle instruction vous permet de vérifier que $cos^2(x) + sin^2(x) = 1$? Cette fois, le résultat devra apparaitre dans la fenêtre de commande.

3 Vecteurs et matrices

Comme son nom l'indique, MatLab est prévu pour manipuler un seul type d'objet : les matrices ! Les scalaires sont ainsi des matrices de dimension 1×1, tandis que les vecteurs sont des matrices de dimension $1 \times n$ ou $n \times 1$, en fonction qu'ils soient lignes ou colonnes.

3.1 Construction explicite

Pour définir le contenu d'une variable, la première des solutions consiste donc à former une matrice en entrant un à un ses coefficients. Pour un scalaire, vous savez déjà faire, mais pour vecteurs et matrices, il faut respecter une syntaxe spécifique :

- **vecteur ligne :** [1 2 3] ou $[1, 2, 3]$. Les crochets marquent le début et la fin du vecteur, et les espaces ou les virgules délimitent ses différents coefficients
- **vecteur colonne :** $[1; 2; 3]$. Le point virgule permet de séparer les différents coefficients, et correspond donc en pratique à un retour à la ligne.
- **matrice :** $[1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$. Les coefficients sont donc entrés ligne par ligne, chacune de ces dernières étant finie par un point virgule.

1. Quelle instruction permet de créer une variable x sous la forme d'un vecteur ligne contenant $\pi/6$, $\pi/4$ et $\pi/3$?
2. Quelle instruction permet de créer une matrice M de dimension 3×3 de sorte que la première ligne contienne x, tandis que les deuxième et troisième ligne contiennent respectivement $cos(x)$ et $sin(x)$?
   (a) une première fois en remplissant explicitement chaque case
   (b) une seconde fois en réfléchissant un peu plus !

3.2 Construction rapide

En pratique, cette solution, un peu rébarbative, est souvent contournée en utilisant des raccourcis. Il en existe plusieurs, qui permettent de créer des vecteurs ou matrices précis :

VECTEURS

`n:m`	nombres de `n` à `m` par pas de 1
`n:p:m`	nombres de `n` à `m` par pas de `p`
`linspace(n,m,p)`	`p` nombres de `n` à `m`
`[x,y]`	concaténer les vecteurs `x` et `y`
`reshape(x,u,v)`	crée une matrice de taille `[u,v]`, à partir de x

MATRICES PARTICULIÈRES

`zeros(m,n)`	matrice nulle de taille `m,n`
`ones(m,n)`	matrice de taille `m,n` dont tous les coefficients valent 1
`eye(n)`	matrice identité de taille `n`
`diag(x)`	matrice diagonale dont la diagonale est le vecteur `x`
`magic(n)`	carré magique de taille `n`
`rand(m,n)`	matrice de taille `m,n` à coefficients i.i.d.de loi uniforme sur $[0, 1]$
`randn(m,n)`	matrice de taille `m,n` à coefficients i.i.d. de loi normale $\mathcal{N}(0, 1)$

1. Quelle instruction permet de créer un vecteur contenant les valeurs de -8 à -5 avec un pas de 0.25 ?
2. Quelle instruction permet de créer un vecteur de longueur 100 de $-\pi$ à π ?
3. Quelle instruction permet de créer une matrice de dimension 10×10 contenant tous les nombres entiers de 1 à 100 dans l'ordre croissant (la première colonne contiendra les nombres de 1 à 10). ?
4. Quelle instruction permet de créer une matrice diagonale dont les valeurs non nulles sont toutes des nombres aléatoires (loi uniforme) de l'intervalle $[0, 5]$.

3.3 Extraction de composantes

Créer une matrice ou un vecteur, c'est bien, mais l'utiliser c'est encore mieux ! Et parfois, ce n'est pas la matrice (ou le vecteur) en tant que tel qu'on souhaite utiliser, mais seulement certains de ses coefficients. Il existe donc un ensemble d'opérateurs permettant l'extraction d'un sous-ensemble d'une variable. En voici quelques exemples :

`size(A)`	nombre de lignes et de colonnes de `A`
`A(i,j)`	coefficient d'ordre `i,j` de `A`
`A(i1:i2,:)`	lignes `i1` à `i2` de `A`
`A(i1:i2,:) =[]`	supprimer les lignes `i1` à `i2` de `A`
`A(:,j1:j2)`	colonnes `j1` à `j2` de `A`
`A(:,j1:j2)=[]`	supprimer les colonnes `j1` à `j2` de `A`
`A(:)`	indexation linéaire de A, (concaténation des vecteurs colonnes de `A`)
`A(i)`	coefficient d'ordre i dans l'indexation linéaire
`diag(A)`	coefficients diagonaux de `A`

Attention, à l'inverse des langages de programmation que vous avez l'habitude d'utiliser, les indices en MatLab commencent à 1, pas à 0 !

1. Créez une matrice A de 3×3 contenant dans l'ordre croissant les nombres entiers de 1 à 9.
   (a) Quelle instruction permet de récupérer la deuxième colonne ?
   (b) Quelle instruction permet de récupérer les deux dernières lignes ?
   (c) Quelle instruction permet de générer une matrice 2×2 contenant seulement les 4 coins ?
   (d) Quelle instruction permet de générer un vecteur contenant les nombres entiers de 1 à 9 dans l'ordre croissant ?
2. Créez deux vecteurs $x = [1\ 2\ 3\ 4\ 5\ 6]$ et $y = [7\ 8\ 9\ 10\ 11\ 12]$ (avec élégance bien sur).
   (a) Quelle instruction permet de générer la matrice $M1 = \begin{pmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{pmatrix}$?
   (b) Quelle instruction permet de générer la matrice $M2 = \begin{pmatrix} 1 & 2 & 3 & 7 & 8 & 9 \\ 4 & 5 & 6 & 10 & 11 & 12 \end{pmatrix}$?
   (c) Quelle instruction permet de générer la matrice $M3 = \begin{pmatrix} 1 & 3 & 5 \\ 8 & 10 & 12 \end{pmatrix}$?
3. En utilisant les matrices qui viennent d'être créées, quelle instruction permet de générer la matrice suivante :

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 7 & 8 & 9 \\ 1 & 2 & 3 & 7 & 8 & 9 \\ 4 & 5 & 6 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{pmatrix}$$

3.4 Opérations matricielles

`A'`	transposée de `A`
`rank(A)`	rang de `A`
`inv(A)`	inverse de `A`
`expm(A)`	exponentielle de `A`
`det(A)`	déterminant de `A`
`trace(A)`	trace de `A`
`poly(A)`	polynôme caractéristique de `A`
`eig(A)`	valeurs propres de `A`
`[U,D]=eig(A)`	vecteurs propres et valeurs propres de `A`
`+ -`	addition, soustraction
`* ^`	multiplication, puissance (matricielles)
`.* .^`	multiplication, puissance terme à terme
`A\b`	solution de $Ax = b$
`b/A`	solution de $xA = b$
`./`	division terme à terme

1. En reprenant la matrice M de l'exercice précédent :
   (a) Créez la matrice Mt égale à la transposée de M ?
   (b) Quel est le résultat de la multiplication terme à terme de M par Mt ?
   (c) Quelles sont les valeurs propres de M ?
2. Considérons le système linéaire suivant : $\begin{cases} 6x + y - 5z &= 10 \\ 2x + 2y + 3z &= 11 \\ 4x - 9y + 7z &= 12 \end{cases}$
   (a) Exprimez ce système sous forme matricielle $Ax = b$
   (b) Montrez que ce système admet une solution, en calculant le déterminant de A.
   (c) En utilisant la puissance de MatLab, déterminez le vecteur x solution de l'équation
   (d) vérifiez la validité de cette solution (en multipliant A par x).

3.5 Opérateurs logiques

En plus des opérations matricielles, il est également possible de relier **logiquement** plusieurs matrices. On utilisera pour cela les opérateurs ci-dessous

`Opérateurs relationnels`	`<`, `<=`, `>=`, `==` (égalité), `~=` (différent)
`Opérateurs logiques`	& (et), | (ou), ~ ou `not` (non)

Comme vous vous en doutez, le résultat de ce type d'opération est booléen. Appliqués à une matrice, il produit donc une matrice de même dimension, formée de 0 et de 1.

1. Créez une matrice M aléatoire à 3 lignes, 7 colonnes (loi uniforme)
   (a) Construisez la matrice P, où chaque coefficient vaut 1 si le coefficient correspondant de M est inférieur à 0.4, et 0 sinon.
   (b) Construisez la matrice Q, égale à la matrice M pour les valeurs comprises dans l'intervalle $[0.2, 0.6]$, et à -1 sinon.
2. Créez un vecteur contenant 1000 valeurs binaires, de sorte que 10% de ces valeurs soient des 1.

3.6 Fonctions usuelles

Pour finir, MatLab inclut un ensemble de fonctions usuelles, qui permettent une analyse rapide de vecteurs ou de matrices. Attention toutefois, la plupart d'entre elles sont prévues pour fonctionner sur des vecteurs, ce qui veut dire que si vous les utilisez sur une matrice, elles seront appliquées successivement sur chaque colonne de cette dernière. Vous aurez l'occasion d'en découvrir au fil des séances, mais en voici quand même quelques unes sur lesquelles vous allez vous exercer :

`max(x)`	maximum
`min(x)`	minimum
`sort(x)`	tri par ordre croissant
`[y, I] = sort(x)`	retourne en plus les indices des élements de `x`
`find(x)`	retourne les indices non nuls de `x`
`[y, I] = find(x)`	retourne des lignes (dans le vecteur `I`) et des colonnes (dans le vecteur `J`) des élements non nuls du `x`
`sum(x)`	somme des éléments de `x`
`prod(x)`	produit des éléments de `x`
`mean(x)`	moyenne des éléments de `x`
`std(x)`	écart type

1. Calculez 23!
2. Tirez 100 nombres aléatoire de l'intervalle $[5, 7]$ (loi uniforme, centrée sur 6), et stockez les dans un vecteur x.
   (a) Calculez la moyenne et l'écart type de ces valeurs.
   (b) Quelle est la valeur minimale du vecteur, et la position du coefficient qui la réalise ?
   (c) Construisez le vecteur y correspondant aux valeurs entières de x. Vous utiliserez pour cela la fonction *floor*
   (d) Quelle est la proportion de coefficients égaux à 5 dans y ?
3. Construisez une matrice M aléatoire à 3 lignes, 7 colonnes (loi uniforme).
   (a) Calculez la moyenne et l'écart type de ces valeurs.
   (b) Quelle est la valeur minimale de la matrice, et la position du coefficient qui la réalise ?
   (c) Combien des coefficients de M sont-ils inférieurs à 0.3 ou supérieurs à 0.7 ?

4 Représentation graphique des résultats

4.1 Représentations de points dans le plan

Il existe plusieurs possibilités pour représenter un ensemble de points $(x(i), y(i))$. Les deux plus utilisées sont la réalisation de courbes ou de nuages de points, ainsi que l'utilisation d'histogrammes :

`plot(x,y,'s')`	tracé d'une courbe ou d'un nuage de points
`bar(x,y,'s')`	tracé sous forme d'un histogramme

's' est un paramètre facultatif constitué d'une chaîne de caractères. Il permet de spécifier les propriétés du tracé (couleur, pointillés, symboles utilisés pour le tracé de points). Pour avoir la liste des valeurs possibles, utilisez l'aide : **help plot**.

1. On cherche à représenter la fonction $sin(x)$ sur l'intervalle $[-\pi, \pi]$:

(a) Créez un vecteur x contenant 10 valeurs régulièrement échantillonnées sur cet intervalle

(b) Utilisez x pour tracer la fonction $sin(x)$. Que constatez-vous ?

(c) Proposez une amélioration pour résoudre le problème visible à la question précédente

(d) Faites en sorte que la fonction apparaisse sous la forme d'une courbe rouge, et en pointillée.

(e) Débrouillez-vous pour ajouter un titre, une légende sur l'axe des x, et une légende sur l'axe des y !

(f) Débrouillez-vous pour que l'axe des x aille de $-\pi$ à π et que l'axe des y aille -1 à 1 !

4.2 Gestion des graphiques

Dessiner des courbes, c'est bien, mais dessiner exactement ce qu'on veut et où on veut, c'est encore mieux. Pour cette raison, il existe plusieurs fonctions à utiliser en fonction des besoins. Voici les principales :

`hold on`	les prochains tracés se superposeront aux tracés déjà effectués
`hold off`	le contenu de la fenêtre graphique active sera effacé lors du prochain tracé
`clf`	efface le contenu de la fenêtre graphique active
`figure(n)`	affiche ou rend active la fenêtre graphique numéro `n`
`close`	ferme la fenêtre graphique active
`close all`	ferme toutes les fenêtres graphiques
`subplot(n,m,p)`	partage la fenêtre graphique active en `m` $\times$ `n` espaces graphiques et sélectionne le `p`-ième.

1. Représentez sur la même figure, $cos(x)$ et $sin(x)$ sur l'intervalle $[-\pi, \pi]$. Vous choisirez des couleurs différentes, et ajouterez une légende.

   (a) En utilisant le "Data Cursor" dans la barre des tâches de la figure, déterminez graphiquement les valeurs de x pour lesquelles $sin(x) = 0.2$.

   (b) En utilisant le "Zoom In", faites en sorte que la figure représente maintenant uniquement l'intervalle $[0, \pi]$.

2. Représentez sur la même figure, mais dans 3 cadrans différents, les fonctions exponentielle, logarithme néperien, et x^2. Débrouillez-vous pour que chaque cadran ait son titre.

3. Tirez 100 couples de points (x, y) aléatoirement dans un carré $[0, 1] \times [0, 1]$.

   (a) Représentez le nuage de points obtenus dans une figure.

   (b) Calculez le centre de gravité G du nuage de points, et ajoutez le en rouge dans la figure.

   (c) Débrouillez-vous pour sauvegarder la figure sous le nom "nuage.fig".

   (d) Chargez le fichier dans une autre fenêtre graphique.