

# FSAB1502 - Exercices introductifs à MATLAB

Benoît Frénay

15 février 2011

## Introduction

**MATLAB** est un outil que vous serez amené à utiliser souvent durant vos études à l'EPL. C'est avant tout un **environnement de calcul interactif**, c'est-à-dire une espèce de calculatrice qui permet de manipuler et visualiser facilement des nombres, des matrices, des fonctions, des champs vectoriels, etc. Mais MATLAB est également un **langage de programmation de haut niveau** qui permet de développer des applications. Enfin, ce qui explique sûrement le succès de MATLAB est le nombre impressionnant de **toolboxes** et **fonctions** qu'il propose. Grâce à celles-ci, vous pourrez par exemple facilement simuler des processus physiques.

Les exercices qui vous sont proposés ici ont pour but de vous familiariser avec MATLAB. Lorsque vous les aurez réalisés, vous serez capables de vous débrouiller avec ses fonctionnalités de base et de trouver vous-même de l'aide.

Chaque exercice est scindé en étapes. À chaque fois, les fonctions nécessaires sont listées et des exercices complémentaires sont proposés. Il n'est pas indispensable de réaliser ces derniers, mais ils vous permettront d'aller plus loin.

Pour poser vos questions sur MATLAB ou prendre un rendez-vous pour recevoir de l'aide, n'hésitez pas à envoyer un email à [benoit.frenay@uclouvain.be](mailto:benoit.frenay@uclouvain.be).

Bonne découverte !

## 1 Au commencement

**Fonctions utilisées** : `who`, `clear`.

Allons-y ! Commencez par lancer MATLAB : la fenêtre principale est découpée en plusieurs parties, dont la **command window**, le **workspace** et la **command history**. C'est dans la command window que vous taperez des commandes MATLAB. Ces commandes apparaîtront ensuite dans la command history et le workspace contiendra les variables que vous créerez.

Vous allez maintenant vous familiariser avec l'environnement graphique de MATLAB et apprendre à créer une variable, puis à la supprimer.

### Étape 1

Créez une **variable** `a` contenant la valeur 42 grâce à la syntaxe `a = 42`. Affichez-la ensuite en tapant son nom dans la command window.

### Étape 2

Votre variable se trouve maintenant dans le workspace. Double-cliquez sur celle-ci pour voir son contenu. Vérifiez également que vos commandes apparaissent dans la command history. Vous pouvez copier celles-ci pour les réutiliser.

### Étape 3

Entre chaque exercice, nous vous conseillons de vider votre workspace. Commencez par faire apparaître une liste de vos variables avec la commande `who`. Ensuite, utilisez la commande `clear` pour toutes les supprimer. Enfin, vérifiez qu'elles ont bien disparu en utilisant à nouveau `who`.

## 2 Des variables et des nombres

**Fonctions utilisées :** `help`, `doc`, `+`, `*` (`mtime`) et `.*` (`time`).

Comme son nom l'indique, MATLAB est avant tout conçu pour manipuler des nombres et des matrices. Votre première tâche consiste à vous familiariser avec la création et la manipulation de matrices. Vous serez également amené à utiliser l'aide de MATLAB.

### Étape 1

Commencez par créer les deux **matrices** A et B telles que

$$A = \begin{bmatrix} 3 & -2 \\ -1 & 2 \end{bmatrix}$$
$$B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Pour cela, utilisez la syntaxe `A = [3 -2; 1 2]` où ; marque le début d'une nouvelle ligne. Vérifiez que A et B sont bien dans le workspace et affichez-les.

### Exercice complémentaire

Créez des matrices avec les fonctions `zeros`, `ones` et `eye` en utilisant `help` et `doc` pour comprendre comment les appeler.

### Étape 2

En utilisant les **opérateurs** `+` et `*`, calculez les matrices C , D et E telles que

$$C = 2A - 1$$
$$D = A + B$$
$$E = AB$$

### Étape 3

En MATLAB, les **matrices à une dimension** sont appelées **vecteurs**. Les vecteurs sont donc une série de nombres placés l'un à la suite de l'autre. Par exemple, la matrice à une seule colonne

$$b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

est un vecteur. Créez la variable `b` et calculez ensuite le produit `Ab`. Vérifiez que MATLAB refuse de calculer `bA` (pourquoi ?).

### Étape 3

Une difficulté importante pour débiter en MATLAB est de **penser matriciel**, c'est-à-dire écrire autant que possible les calculs à l'aide d'opérations matricielles. Par exemple, si nous souhaitons calculer le carré de la norme de  $\mathbf{b}$  telle que

$$\|\mathbf{b}\|^2 = \sum b_i^2,$$

il existe deux solutions : utiliser une boucle ou des opérations matricielles. En utilisant les opérateurs de multiplication `*` et de transposition `'`, calculez  $\|\mathbf{b}\|^2$  sans utiliser de boucle. Conseils : n'oubliez pas que  $\mathbf{b}$  est une matrice à une dimension pour MATLAB et travaillez sur papier avant de programmer.

### Étape 4

MATLAB distingue deux sortes d'opérateurs sur les matrices : les **opérateurs matriciels** et **élément-par-élément**. L'opérateur `*` (`mtimes`) que vous avez déjà utilisé est un exemple d'opérateur matriciel. Utilisez l'**aide** pour découvrir la différence avec l'opérateur élément-par-élément `.*` (`times`), en tapant `help times` et `help mtimes` ou `doc times` et `doc mtimes`. Enfin, calculez la matrice  $\mathbf{F}$  où  $F_{ij} = A_{ij}B_{ij}$ .

#### Remarque

Notez que les fonctions `help` et `doc` affichent la **documentation** de n'importe quelle fonction MATLAB, mais `doc` est généralement bien plus complète et propose en plus des exemples, des explications, des références, etc. `doc` propose également des introductions aux fonctionnalités de MATLAB : algèbre linéaire, polynômes, traitement du signal, statistiques, simulation numérique, etc.

#### Exercice complémentaire

Utilisez l'aide de MATLAB pour trouver l'effet des opérateurs `^` (`mpower`) et `.^` (`power`). Ensuite, essayer de prévoir ce que vaudront les deux matrices  $\mathbf{G} = \mathbf{A}^2$  et  $\mathbf{H} = \mathbf{A}.^2$  et comparez votre résultat avec celui de MATLAB.

## 3 Il était une fonction

**Fonctions utilisées :** `linspace`, `sin`, `figure`, `plot`, `hold`, `title`, `xlabel`, `ylabel`.

MATLAB est souvent utilisé pour étudier des fonctions. Votre seconde tâche consiste à visualiser l'évolution de deux tensions alternatives  $V_1(t)$  et  $V_2(t)$  (voir Fig. 1). Pour rappel, une tension alternative est une fonction de la forme

$$V_i(t) = V_0 \sin(\omega_i t + \phi_i)$$

où  $V_0$  est l'amplitude,  $\omega = 2\pi f$  est la vitesse angulaire liée à la fréquence  $f$  et  $\phi$  est le déphasage. Ici,  $V_0 = 1\text{V}$ ,  $f_1 = 1\text{Hz}$ ,  $f_2 = 4\text{Hz}$  et  $\phi_1 = \phi_2 = 0$ .

#### Exercice complémentaire

Choisissez une autre fonction et recommencez les étapes de cette tâche. Par exemple, visualisez la position verticale  $y(t)$  d'un corps en chute libre jusqu'à l'impact.

### Étape 1

Commençons par visualiser  $V_1(t)$  pendant 1s. Pour cela, il vous faut deux vecteurs contenant les valeurs de  $t$  et de  $V_1(t)$ . Créez un vecteur `t` contenant 1000 valeurs entre 0 et 1 en utilisant la **fonction** `linspace`. Ensuite, calculez un vecteur `V1` contenant les valeurs de la tension pour chaque valeur de `t`. Utilisez l'aide pour découvrir comment utiliser les fonctions `linspace` et `sin`.

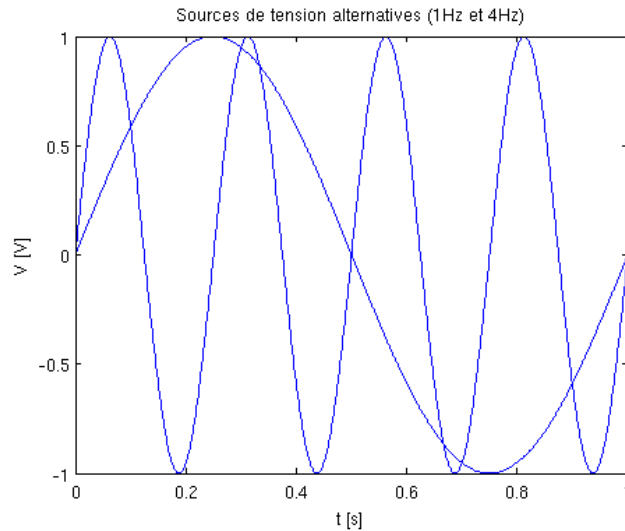


Figure 1: Exemple de résultats avec plot.

### Remarque

N'oubliez pas que  $V1$  est un vecteur. Autrement dit, chacun de ses éléments correspond à un indice allant de 1 à 1000. La même chose vaut pour  $t$ . C'est uniquement en faisant correspondre l'élément  $t(i)$  à l'élément  $V1(i)$  que l'on peut connaître la valeur de la tension à un instant donné.

### Étape 2

Pour pouvoir afficher  $V_1(t)$ , il faut ouvrir une nouvelle figure. Pour cela, utilisez la commande `figure`. Vérifiez ensuite que vous pouvez fermer cette fenêtre grâce à la commande `close`, puis créez en une autre pour les étapes suivantes.

### Étape 3

La figure est prête à afficher tout ce que vous souhaitez. Commencez par afficher l'évolution de  $V_1(t)$  en utilisant la fonction `plot`. Pour le moment, ne vous souciez que des deux premiers arguments de cette fonction.

### Exercice complémentaire

Essayer de changer l'apparence de la ligne qui apparaît sur la figure. Pour cela, donnez `'r.'` comme troisième argument à `plot`. Essayez d'autres valeurs, par exemple remplacez `r` par `b` ou `g`, `.` par `o` ou `x`, `-` par `--` ou `..`.

### Étape 4

Enfin, il reste à afficher sur la même figure  $V_2(t)$ . Pour cela, utilisez la fonction `hold` qui permet de superposer plusieurs graphes et répétez les étapes 1 à 3.

### Étape 5

Notre figure est presque terminée, il ne reste plus qu'à lui ajouter un titre et munir ses axes de labels grâce aux commandes `title`, `xlabel` et `ylabel`.

### Remarque

Si nécessaire, les limites et les échelles des axes peuvent également être modifiées.