Laboratoire de programmation MATLAB

GNG1503 – Génie de la Conception

Objectif

Apprendre les bases de l'environnement du logiciel Matlab pour écrire de simples codes mathématiques. Utiliser MATLAB pour la visualisation graphique des données et l'écoute auditive des signaux.

Contexte

MATLAB est un langage de haute performance pour l'informatique technique. Il intègre le calcul, la visualisation et la programmation dans un environnement facile à utiliser où les problèmes et les solutions sont exprimés dans une notation mathématique familière. Les applications typiques comprennent: Mathématiques, calcul, développement d'algorithmes, modélisation, simulation, prototypage, analyse de données, exploration, visualisation, graphisme scientifique et technique, développement d'applications et la construction d'interface graphique par l'utilisateur.

MATLAB est un système interactif dont l'élément de données de base est un tableau qui ne nécessite pas de dimensionnement. Cela vous permet de résoudre de nombreux problèmes de calcul technique, en particulier ceux qui ont des formulations matricielle et vectorielle, dans une fraction du temps qu'il faudrait pour écrire un programme dans un scalaire sans langage interactif tel que C ou FORTRAN.

Le nom MATLAB est synonyme de matrice de laboratoire. MATLAB a été initialement écrit pour fournir un accès facile au logiciel matriciel. MATLAB a évolué sur une période de plusieurs années avec l'apport de nombreux utilisateurs. Dans les environnements universitaires, il est l'outil d'enseignement standard pour les cours d'introduction et avancés en mathématiques, en génie et en sciences. Dans l'industrie, MATLAB est l'outil de choix pour la recherche, le développement et l'analyse à haute productivité.

Le langage MATLAB est un langage matriciel / tableur de haut niveau avec des instructions de flux de commande, des fonctions, des structures de données, des entrées / sorties et des fonctions de programmation orientée objet. Il permet à la fois de réaliser «des petits programmes» pour créer rapidement des programmes, et «des grands programme» pour créer des programmes d'application complets et complets.

Référence

Vous pouvez trouver toutes les informations sur MATLAB dans le site officiel de MATLAB:

Https://www.mathworks.com/help/matlab/learn_matlab/plots.html

Tutoriel

Pour accéder au didacticiel, allez à: www.lynda.com , connectez-vous et écrivez MATLAB dans la zone de recherche.

Il n'y a qu'un seul tutoriel MATLAB que vous verrez. Ouvrez-le et vous pouvez commencer à le regarder.

Le tutoriel étant en Anglais, des explications supplémentaires peuvent être retrouvé dans l'appendice de se manuel de laboratoire pour ceux qui ne se sentiront pas confortable avec les vidéos. Vous pouvez choisir soit d'écouter le tutoriel en anglais, soit de lire les explications supplémentaire.

Pré-laboratoire

Avant d'arriver au laboratoire, l'étudiant devrait revoir et se familiariser avec le laboratoire et les procédures. Il est **fortement** recommandé d'écouter la première partie de la vidéo à la maison pour comprendre la base de MATLAB.

Questions Pré-lab

Qu'est-ce que le terme MATLAB représente?

Matrice de laboratoire.

Quelles sont les capabilités de MATLAB?

Il intègre le calcul, la visualisation et la programmation dans un environnement facile à utiliser où les problèmes et les solutions sont exprimés dans une notation mathématique familière.

Quel est l'élément mathématique de base que MATLAB utilise?

MATLAB est un système interactif dont l'élément de données de base est un tableau qui ne nécessite pas de dimensionnement.

Comment est appelé la fenêtre où tu tapes des commandes?

Appeler la fonction pole.

Comment tu effaces la fenêtre de commande?

Réutiliser la fonction pole.

IMPORTANT: Pendant le laboratoire, vous allez prendre des captures d'écran pour chaque partie. Coller ces captures d'écran dans un document Word (ou autre) à mesure que vous avancez à travers le manuel de laboratoire pour que vous puissiez créer un document PDF de toutes vos captures d'écran de votre travail. Ce document PDF de vos captures d'écran sera soumis aussi comme livrable pour ce laboratoire.

PARTIE A. Concept Général

Écoutez les vidéos suivantes :

- Understanding MATLAB interface (Comprendre l'interface MATLAB)
- Working with MATLAB variables (Travailler avec les variables de MATLAB)
- Everything is a matrix (Tout est une matrice)
- Understanding data structures (Comprendre les structures de données)

Exercices :

1) Écrire trois matrice X, Y et Z. X est une matrice 3x3, Y est une matrice 3x4 et Z est une matrice 4x4. Imprimez-les sur la fenêtre de commande (Command Window).
2) Faites une capture d'écran de la fenêtre de commande. Vous allez l'utiliser pour votre rapport de laboratoire.
3) Effacer le contenu de la fenêtre de commande.
4) Écrire le code suivant:

```
x = 1:2:30
y = 1:3:30
z = 1:4:30
```

Expliquez ce qui se passe dans chaque cas

Ceci montre des series de nombres qui commence par 1 et terminent par 30 et qui ont un difference entre elle dependant du nombre au centre.

5) En utilisant l'exemple illustré sur la vidéo, créer une matrice qui contient un **nom**, une **adresse** et **un numéro de téléphone**. Imprimez-les sur l'écran Commande Windows en une fois.
6) Faites une capture d'écran de la fenêtre de commande. Vous allez l'utiliser pour votre rapport de laboratoire.
7) Effacer le contenu de la fenêtre de commande.

PARTIE B. Syntaxe de base MATLAB

Écoutez les vidéos suivantes :

- Basic commands (Commandes de base)

- Using built-in functions and variables (Utilisation de fonctions et de variables intégrées)
- Working with matrix and scalar operations (Travailler avec des opérations matricielles et scalaires)
- Control flow (Flux de contrôle)

Exercices:

1) Considérer trois matrices A = 'Université', B = 'd'' and C = 'Ottawa'. Trouvez un moyen d'imprimer sur l'écran Commande Windows le texte: "**Université d'Ottawa**". Pensez au laboratoire Excel où vous deviez combiner un prénom et un nom.
   **Notez bien : les espaces fonctionnent seulement au début d'une chaîne de texte et non à la fin. Aussi, pour ajouter une apostrophe, il faut mettre deux guillemets simples de suite plutôt que juste une.*
2) Faites une capture d'écran de la fenêtre de commande. Vous allez l'utiliser pour votre rapport de laboratoire.
3) Effacer le contenu de la fenêtre de commande.
4) Écrire deux matrices Y et Z où:

$$Y = \begin{matrix} 2 & 1 & 4 \\ 3 & 2 & 1 \\ 1 & 4 & 3 \end{matrix} \qquad Z = \begin{matrix} 2 & 3 & 6 \\ 1 & 5 & 7 \\ 3 & 4 & 1 \end{matrix}$$

5) En utilisant MATLAB, effectuez les opérations suivantes:
   T = Y+Z
   U = Y-Z
   V = Y*Z
   W= Y.*Z
6) Faites une capture d'écran de la fenêtre de commande. Vous allez l'utiliser pour votre rapport de laboratoire.
7) Quel est la différence entre V et W?

   V fait la multiplication matricielle tantdis que W fait la multiplication entre les nombres par ex (a du Z x a du Y)

8) Quelle est la valeur dans la matrice T correspondant à la ligne 3 colonne 2? Écrivez votre réponse, ainsi que la syntaxe MATLAB utilisée pour accéder à cet index.

   8 et on le trouve en faisant T(3,2)

9) Qu'est-ce que le fait de taper *W(:,2)* dans la fenêtre de commande et d'appuyer sur entrer vous donne? Expliquer ce que le point-virgule fait lors de l'accès aux indices.

Ceci nous donne toutes les valeurs de la deuxieme colone de W.

PARTIE C. Programmation en MATLAB

Écoutez les vidéos suivantes :

- How are program files stored (Comment les fichiers de programme sont-ils stockés)
- Viewing and editing programs (Affichage et modification des programmes)
- Creating scripts (Création de scripts)
- Debugging (Débogage)

Exercices:

1) Ouvrir un nouveau script dans MATLAB.
2) Écrire un code qui génère une matrice aléatoire 3x3.
3) Enregistrez et exécutez le code.
4) Faites une capture d'écran de la fenêtre de commande. Vous allez l'utiliser pour votre rapport de laboratoire.
5) Télécharger le fichier Example_1 dans Campus Virtuel et enregistrez le dans le **dossier MATLAB** dans "document".
6) En utilisant MATLAB, lisez le fichier "Example_1" et trouvez la moyenne de tous les nombres dans le fichier.

En inserant le fichier Exemple_1 sur MATLAB et ecrivant

mean(Example_1). Ceci nous donne une moyenne de 107.5000

PARTIE D. Traçage de graphique

Écoutez les vidéos suivantes :

- Creating basic plots (Création de graphiques de bases)
- Adding annotations (Ajout d'annotations)

1) Ouvrir un nouveau script.
2) Écrire le code suivant:

```
x = -20:2:20
y = (x.^2)
plot(x,y)
```

3) Exécutez-le et observez le graphique.

4) Ajoutez des informations à votre graphique: axe des x, axe des y et le titre.
5) Faites une capture d'écran de la fenêtre de commande. Vous allez l'utiliser pour votre rapport de laboratoire.
6) Écrire le code suivant:

```
x = 0:0.1:20
y1 = sin(x)
y2 = cos(x)
subplot(211)
plot(x,y1)
xlabel('x')
ylabel('y1')
title('Sine function')
subplot(212)
plot(x,y2)
xlabel('x')
ylabel('y2')
title('Cosine function')
```

7) Exécutez le code.
8) Remplacer subplot(211) par subplot(121) et subplot(212) par subplot(122)
   Expliquez ce qui se passe?

D'apres ce que j'ai remarque, ceci fait changer la variation d'axe des Y.

PARTIE E. Signal audio (dois être fait à partir d'un ordinateur de laboratoire parce que les serveurs utilisés pour RemoteApps n'ont pas de gestionnaire audio)

Dans cette partie nous allons synthétiser des signaux audios avec MATLAB. Pour le faire, nous allons utiliser la fonction *sound()*. Cette fonction prend une matrice (y) et une fréquence d'échantillonnage (fs) et convertit ces informations en un son qui est joué par vos haut-parleurs. La syntaxe est la suivante:

sound(y, fs)

1) Ouvrir un nouveau script.
2) Téléchargez le fichier *noteFrequency.mat* sur Campus Virtuel et enregistrez-le dans votre dossier MATLAB actuel et utilisez le code suivant pour charger les données dans votre espace de travail.

```
load noteFrequency.mat
```

3) Vous avez maintenant une structure 1x1 qui a les fréquences des notes c, D, E, F, G, A, B, C où c et C sont respectivement le C bas et le C haut. Nous pouvons accéder à la fréquence d'une note en tapant *frequency.note*. Par exemple entrer *frequency.A* retourne 440 Hz.
4) Écrire le code suivant :

```
notes = 'cDEFGABC';
timeArray = [1 1 1 1 1 1 1 1];
```

5) Nous avons maintenant une matrice de notes (rappelez-vous qu'une chaîne est juste une matrice de caractères!) et une matrice correspondante de valeurs de temps décrivant combien de temps nous allons jouer la note. (C-a-d. `notes(1) = c` et `timeArray(1) = 1` signifiant que la première note est un C bas pendant 1 seconde).

6) Pour jouer successivement chacune de ces paires note/temps, nous allons utiliser une boucle *for*. Écrivez le code suivant dans votre script. Notez que tout ce qui suit un % est un commentaire (en vert) et n'est pas compilé dans le programme, donc vous n'avez pas besoin d'écrire ceux-ci.

```
for i = 1:8 % Ici nous avons 8 notes à jouer alors "i" va de 1 à 8

    fs = 24000; % C'est notre fréquence d'échantillonnage en Hz
    t = 0:timeArray(i)*fs; % Nous avons besoin d'une matrice de temps comme dans les parties D et F
    note = notes(i); % Obtenez la note que nous allons jouer
    sineWave = sin(2*pi*frequency.(note)*t/fs); % Générer une onde sinusoïdale définie par la paire note / temps. Sin utilise des radians, d'où le "pi"
    sound(sineWave,fs) % Jouez le son
    pause(timeArray(i)) % Pause pour la durée de la note

end % Termine la boucle
```

7) Si vous exécutez ce code, vous devriez entendre une échelle! Notez comment nous utilisons la variable d'espace réservé *i* pour accéder aux indices dans notes et timeArray. Cela nous permet d'utiliser la boucle for pour jouer chaque note en séquence.

8) Maintenant, notes + timing = musique! Modifiez le code ci-dessus pour jouer les paires notes / temps suivantes. Astuce: Les notes et les horaires sont différents, mais il en est de même du nombre de notes.

Ordre	Note	Durée (s)
1	E	0.8
2	G	0.6
3	G	0.2
4	c	0.8
5	D	0.4
6	E	0.4
7	F	0.4
8	G	0.4
9	A	0.4
10	D	0.8

Quelle chanson entendez-vous? Inclure une capture d'écran de votre code dans votre rapport.

O Canada.

PARTIE F. La balle qui tombe

(1) $y = y_i + (v_i * t) - (\,^1/_2 * g * t^{2)}$
(2) $v = v_i - (g * t)$

Ces deux équations sont les plus fondamentales de la chute libre. Ils décrivent l'évolution de la position et de la vitesse en fonction du temps. Dans ces équations, l'accélération a_y = -g est constante, y_i représente la position initiale (à t = 0s) et v_i la vitesse initiale. Nous considérons un axe y dirigé vers le haut et g correspond à la valeur de l'accélération gravitationnelle (près de la terre g = 9,8 m / s2).

Exercice:

Une balle est lancée verticalement vers le haut à une vitesse initiale de 30 m / s à partir d'une hauteur de 20m.
En utilisant MATLAB, calculer la position et la vitesse de la balle à tous les instants t de 0s à 7s avec un intervalle de 0,5s.

Tracer dans la même figure deux graphes **vitesse vs temps** et **position vs temps**. Ajoutez toutes les informations de votre graphique.

SOUMMISSION

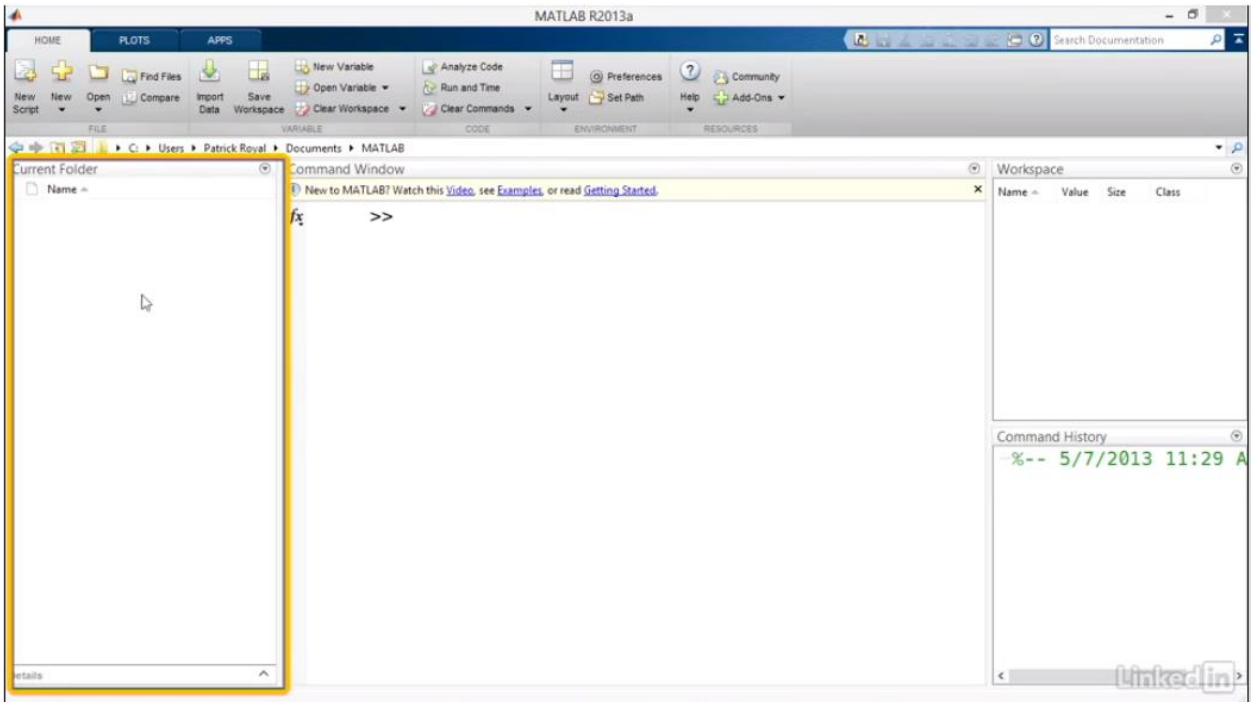
Soumettez ce manuel de laboratoire comme un PDF imprimé avec les réponses aux questions sur les lignes fournies en téléchargeant sur Campus Virtuel avant la date limite. Vous devez aussi inclure dans votre soumission le document PDF avec vos captures d’écran. Tous les fichiers téléchargés devraient être soumis comme une seule soumission avec plusieurs téléchargements, plutôt que des soumissions séparées.

Appendix 1 : Document supplémentaire

Ce document comporte toutes les informations nécessaires pour répondre aux questions du laboratoire. Si vous êtes confortable à écouter la vidéo en Anglais, vous pouvez vous passer du document.

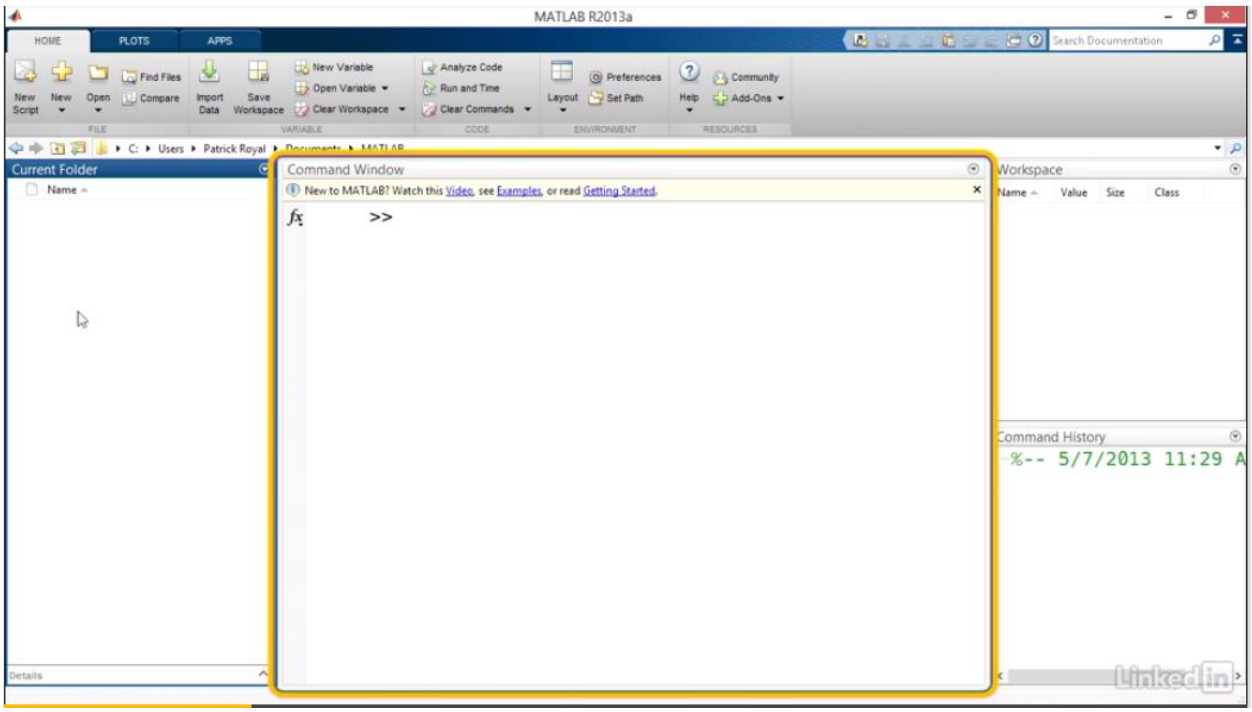
Comprendre l'interface MATLAB

Lorsque vous ouvrez MATLAB pour la première fois, vous verrez que l'interface est divisée en quatre fenêtres. Sur la gauche, la fenêtre de dossier actuelle (current folder).



C'est l'endroit où sont placés tous les fichiers pertinents à MATLAB dans le dossier où vous êtes. Mais c'est aussi là que toutes les fonctions et les scripts que vous créez seront stockés. Il est important de vous assurer que vous êtes dans le bon dossier, lorsque vous exécutez des scripts et des fonctions car MATLAB considérera le dossier actif dans le but d'exécuter des fonctions. Par défaut, MATLAB définira le dossier actif dans un dossier nommé MATLAB dans votre dossier Documents.

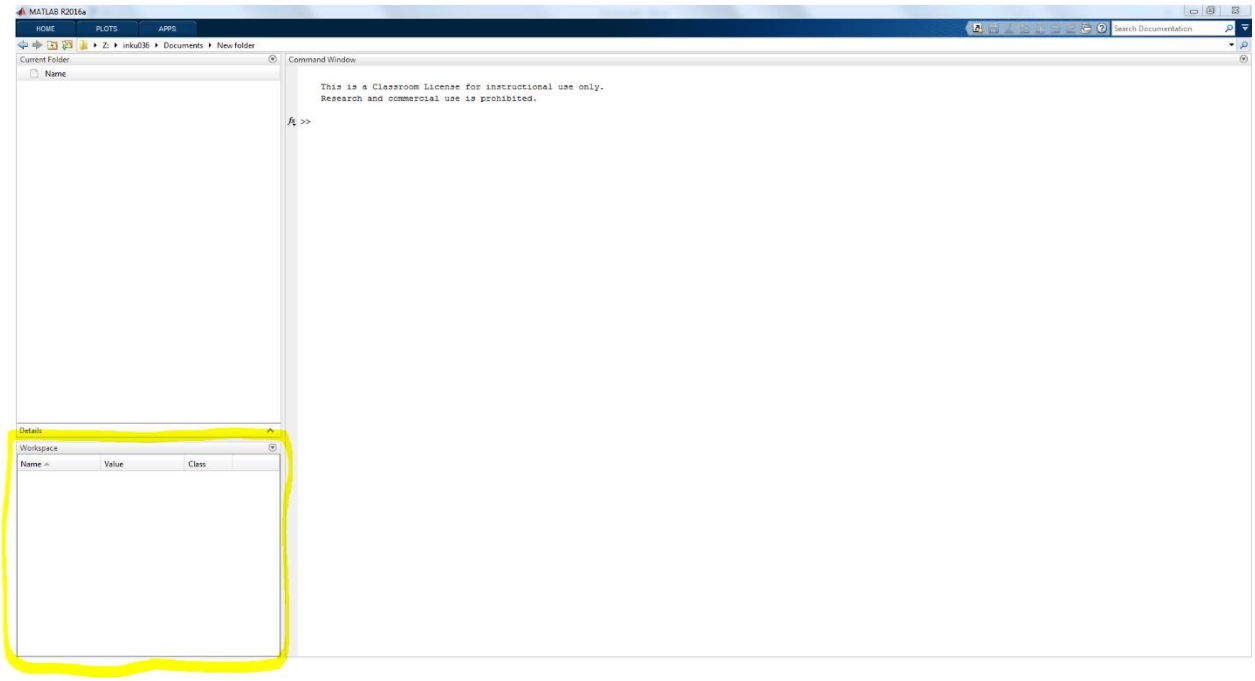
Au milieu de l'écran se trouve la fenêtre de commande.



Cette zone contient toutes vos entrées et sorties lorsque vous exécutez diverses fonctions. Chaque fois que vous exécutez un script ou une fonction, toute sortie sera affichée ici. Techniquement, la fenêtre de commande possède toutes les mêmes fonctions que l'éditeur de script. Ainsi, il est possible d'écrire des programmes directement sur cette ligne. Cependant, dans la pratique, vous voudrez probablement quelque chose que vous pouvez enregistrer, éditer et exécuter plusieurs fois.

À tout moment, si la fenêtre de commande devient trop encombrée, vous pouvez la supprimer en tapant **clc** et en appuyant sur Entrée, ou en cliquant sur la flèche du coin supérieur droit du carreau et en choisissant Effacer la fenêtre de commande.

En bas à gauche de l'écran se trouve l'espace de travail (Workspace).



Il contient la liste de toutes les variables actives dans votre simulation. Pour modifier les informations disponibles sur vos variables, cliquez avec le bouton droit de la souris sur les colonnes et sélectionnez l'information que vous souhaitez afficher.

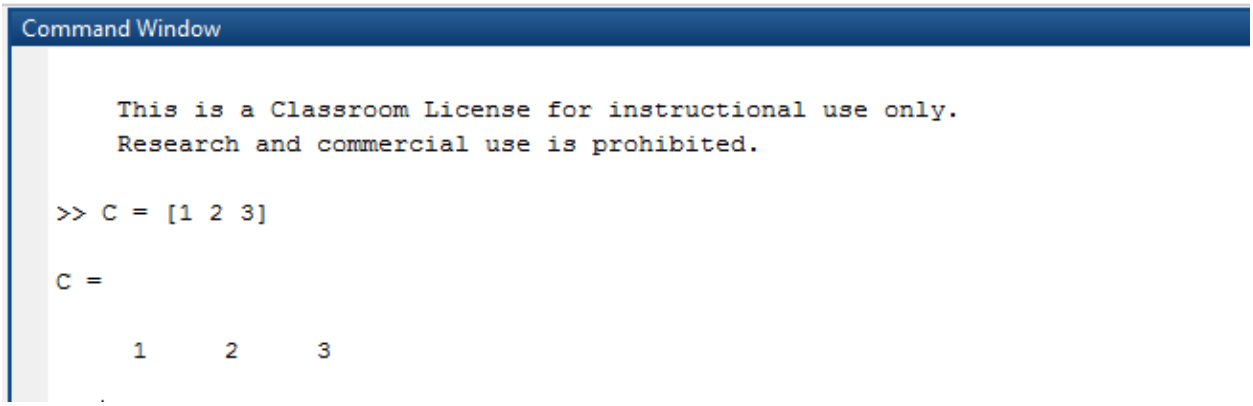
L'espace de travail affiche toutes les variables que vous avez utilisées dans une session MATLAB donnée, quel que soit le script ou la fonction dont ils proviennent. Donc, après un certain temps, il peut être encombré de données anciennes. Pour nettoyer l'espace de travail, cliquez sur la flèche déroulante en haut à droite de l'écran Workspace et choisissez Effacer l'espace de travail. Après avoir confirmé, MATLAB supprimera toutes les variables existantes.

Travailler avec les variables de MATLAB

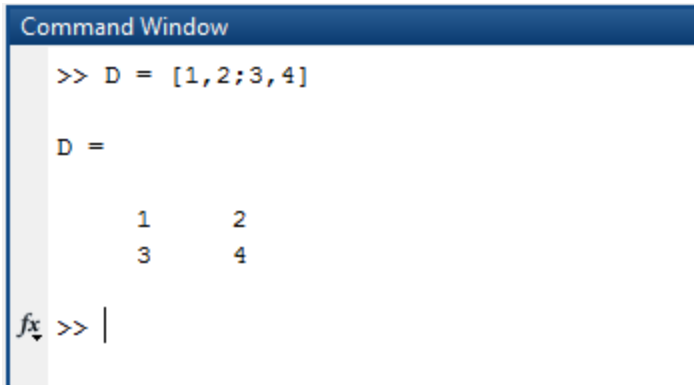
Jetons un coup d'oeil à la façon de créer et de manipuler les variables MATLAB de base. Pour commencer, nous allons avoir besoin de certaines variables. Pour l'instant, nous allons utiliser la méthode la plus simple de création de variables, en tapant dans la fenêtre de commande. Définir une nouvelle variable dans MATLAB est très facile. Contrairement à d'autres langages de programmation comme C ou Java, il n'est pas nécessaire d'ajouter une instruction de déclaration de variable. En d'autres termes, vous n'avez pas besoin de dire à MATLAB quelque chose comme, cette variable est un entier, ou cette variable est une chaîne. Au lieu de cela, tout ce que vous devez faire est de dire à MATLAB ce à quoi votre variable est égale, alors vous pouvez le faire maintenant.

Dans la fenêtre de commande, tapez A égal à 1 et appuyez sur Entrée. La fenêtre de commande (Command Window) vous rend le résultat A est égal à 1. L'espace de travail (Workspace) affiche maintenant une nouvelle variable appelée A avec une valeur de 1 et une taille de 1 par 1. Cela dit juste que A est un scalaire.

MATLAB prend également en charge les variables qui sont des vecteurs ou des matrices. Pour créer un vecteur dans votre déclaration de variable, placez des crochets autour de votre terme et séparez chaque valeur entre crochets avec une virgule ou un espace. Par exemple, C = [1 2 3] fait que C soit créé comme un vecteur un par trois avec les trois valeurs qui y sont stockées.



Pour créer une matrice, le même principe s'applique, sauf que vous utilisez un point-virgule pour séparer les lignes. Par exemple, D = [1,2;3,4] fait que D soit créé comme une matrice 2 par 2 avec ces valeurs.



Comprendre les structures de données

Les structures sont construites dans le mécanisme de stockage MATLAB qui vous permet d'associer plusieurs éléments de données ensemble. Par exemple, vous pouvez avoir une liste de noms et d'adresses, et vous pourriez vouloir faire correspondre chaque nom à son adresse. Pour créer une structure et y ajouter des données, il suffit de définir chaque champ de données à l'aide d'un point. Dans notre exemple, nous dirons quelque chose comme *s.nom = John Doe*. Et puis *s.adresse =123 Fake Street*.

Pour ajouter des données supplémentaires, vous pouvez ajouter des données pour s(2), s(3), etc. Par exemple, on pourrait dire *s(2).nom = Jane Doe* et *s(2).address = 124 Fake street*. Comme d'habitude, cela est considéré comme une matrice, de sorte que votre structure peut avoir un nombre quelconque de dimensions et peut même contenir des sous-structures. L'accès aux données fonctionne de la même manière. Saisie de *s(1.1).nom* renverra seulement le nom du premier point de données, tandis que s(1,1) sans qualificateur renverra l'entrée entière. Si vous saisissez un nom avec un numéro de ligne ou de colonne, vous renverrez tous les noms de tous les champs.

Commandes de base

Pour définir des variables en tant que texte plutôt qu'une valeur numérique, placez des guillemets simples autour du texte que vous définissez. Si vous voulez que votre chaîne soit variable plutôt que fixe, les fonctions *num2str* et *strcat* seront utiles. Nam2str convertit une valeur numérique en une chaîne, ce qui permet d'ajouter un nombre non prédéterminé à la chaîne. *Strcat* concatène deux chaînes strictement pour former une chaîne plus longue.

```
>> a = 'Hello'

a =

Hello

>> b = 'World'

b =

World

>> c = strcat(a,b)

c =

HelloWorld
```

Contrairement à Java, MATLAB ne vous oblige pas à placer un point-virgule à la fin de chaque ligne. Si vous le faites, MATLAB va l'interpréter comme une commande pour supprimer la sortie pour tout ce qui se passe sur la ligne précédente. C'est une fonction extrêmement utile, car par défaut, MATLAB affiche les résultats de chaque fonction, équation, définition de variable ou boucle pendant le programme. Pour les scripts plus volumineux, cela peut accabler rapidement la fenêtre de commande et rendre le programme inutilisable.

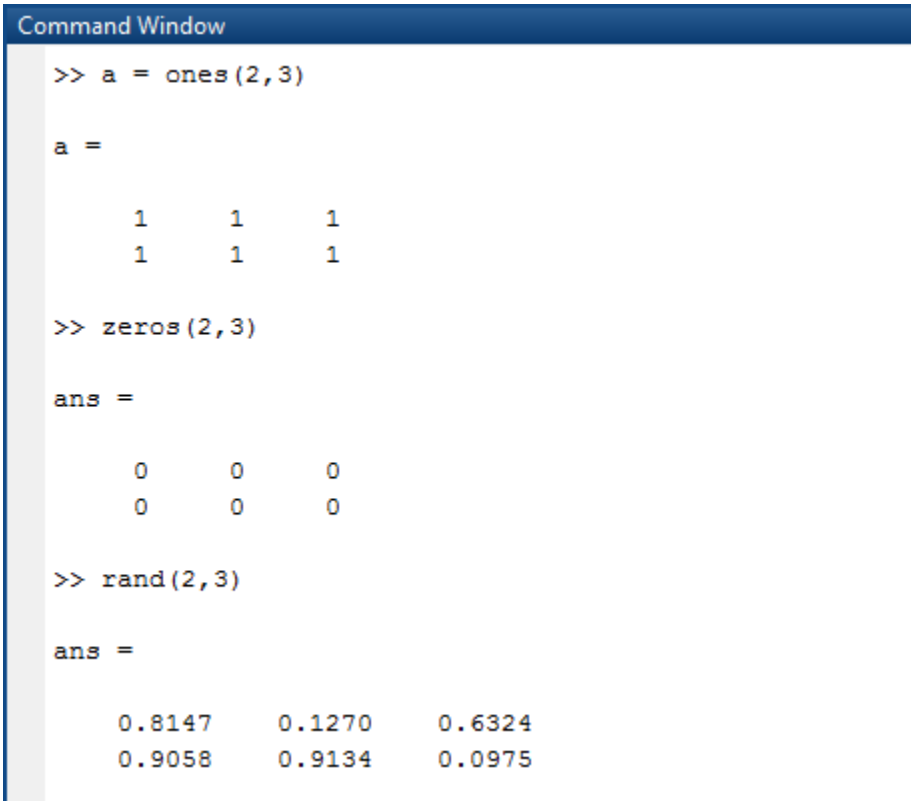
En général, sauf si vous voulez spécifiquement voir la sortie d'une ligne, c'est une bonne pratique de terminer chaque ligne avec un point-virgule.

```
>> a = 'Hello';
>> b = 'World';
>> c = strcat(a,b);
>> % If we want to see the value of a we just write :
>> c

c =

HelloWorld
```

Utilisation de fonctions et de variables intégrées

Fondamentalement, une fonction diffère d'un script parce ce qu'elle a des canaux d'entrée et des canaux de sortie fixe. Cela rend les fonctions extrêmement utiles lorsque vous écrivez des programmes compliqués qui peuvent répéter les mêmes calculs encore et encore. Plutôt que de réécrire votre code, vous pouvez simplement réécrire cette partie de celui-ci dans une fonction distincte. MATLAB est livré avec une grande variété de fonctions déjà mises en œuvre. Donc, nous allons commencer par examiner comment les utiliser. Les fonctions les plus couramment utilisées sont des fonctions de génération de matrice qui sont utilisées pour créer une matrice avec certaines données de démarrage.

A titre d'exemple, écrivons a = ones(2,3) dans la fenêtre de commande. Cela indique à MATLAB de générer une nouvelle matrice avec deux lignes et trois colonnes, puis de définir toutes les valeurs dans cette matrice égale à 1. La fonction **zéros** fait la même chose avec des zéros, la fonction **rand** fait la même chose avec des nombres aléatoires et ainsi de suite. La chose importante à noter ici est que toutes ces fonctions utilisent la même syntaxe. Tapez le nom de la fonction suivi des parenthèses contenant chacune des entrées de la fonction séparées par des virgules.



Travailler avec des opérations matricielles et scalaires

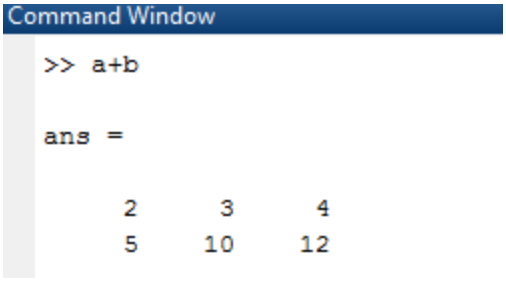
Examinons les opérations matricielles de base utilisées par MATLAB. Puisque MATLAB traite toutes les variables comme s'il s'agissait d'une matrice, il est important de distinguer entre des opérations comme la multiplication de la matrice par rapport à la multiplication du scalaire. Nous allons pratiquer différents types d'opérations matricielles.

Utilisons les matrices ci-dessous :

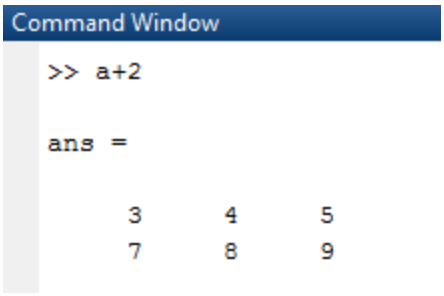
Command Window

```
>> a = [1,2,3;5,6,7]

a =

     1     2     3
     5     6     7

>> b = [1,1,1;0,4,5]

b =

     1     1     1
     0     4     5

>> c = [3,1;2,2;0,0]

c =

     3     1
     2     2
     0     0

>> d = [4,4,2;-1,2,4]

d =

     4     4     2
    -1     2     4

>> e = 1:4

e =

     1     2     3     4

>> f = [2,1,4,3]

f =

     2     1     4     3
```

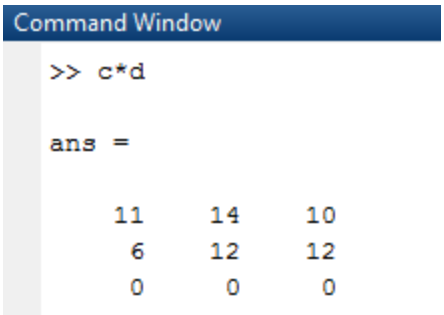
Pour commencer, tapons a + b dans la fenêtre de commande. Puisque la matrice a et la matrice b ont la même dimension, MATLAB l'interprétera comme addition par morceaux et ajoutera simplement les valeurs correspondantes dans chaque cellule des matrices.



D'autre part, si vous tapez a + 2, MATLAB va interpréter cela comme ajout scalaire, de sorte qu'il ajoutera deux à chaque cellule dans A.



Le même principe s'applique à la multiplication des matrices. Si nous entrons c * d, MATLAB voit que ces deux variables sont des matrices, et leurs dimensions internes correspondent(le nombre de colonne de c est égal au nombre de ligne de d). Donc, il effectuera automatiquement la multiplication matricielle.



D'autre part si vous tapez c * 2 MATLAB considère 2 comme une constante et multipliera chaque terme dans c par 2.

```
Command Window
>> c*2

ans =

     6     2
     4     4
     0     0
```

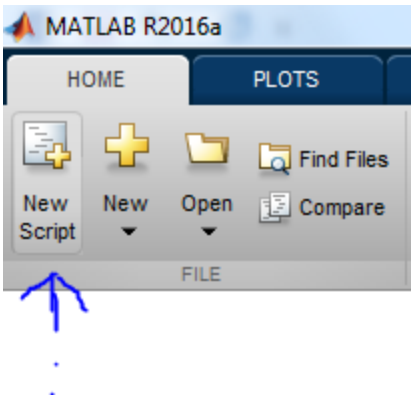
La multiplication matricielle est un processus assez direct. Mais que se passe-t-il si nous voulions que MATLAB effectue une multiplication par morceaux des entrées correspondantes de la matrice. Les matrices a et b ont les mêmes dimensions, mais si vous entrez juste une fois b, MATLAB ne sait pas que nous voulons multiplier les entrées correspondantes. Donc, il suppose que nous voulons multiplier les matrices et affiche une erreur parce que les dimensions intérieures de ces deux matrices ne correspondent pas. Au lieu de cela, nous pouvons ajouter un point devant l'astérisque.

```
Command Window
>> a*b
Error using  *
Inner matrix dimensions must agree.

>> a.*b

ans =

     1     2     3
     0    24    35
```

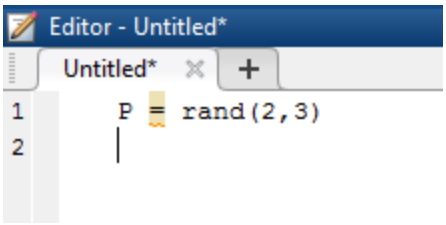
L'ajout d'un point devant toute opération, addition, soustraction, multiplication, division, exposants ou même égalité, indique à MATLAB d'exécuter l'opération par morceaux. Il s'agit d'un moyen vraiment efficace de manipuler de grandes quantités de données à la fois.

Affichage et modification des programmes

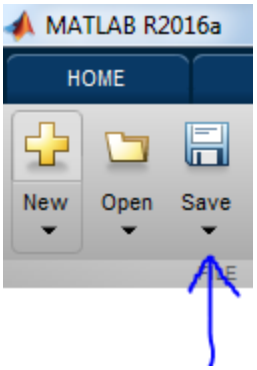
Créer un nouveau programme est facile. Il suffit de cliquer sur le bouton Nouveau script pour générer le script et ouvrir automatiquement la fenêtre Éditeur de scripts.



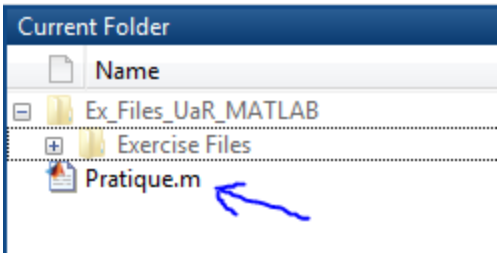
Ici, vous pouvez taper tout le code qui sera exécuté comme une partie du programme. Par exemple, nous allons simplement créer un script très simple qui génère une matrice aléatoire deux par trois. Donc P = rand(2,3).



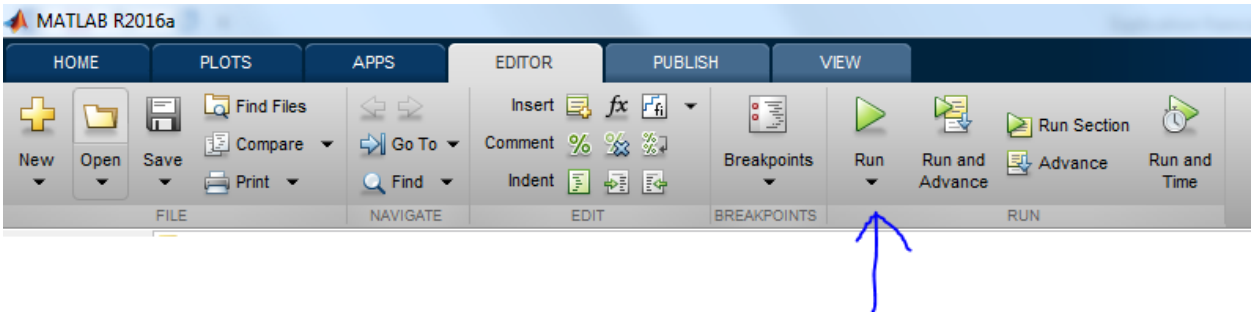
Lorsque vous êtes prêt à enregistrer, cliquez sur le bouton Enregistrer et donnez au programme un nom et il sera automatiquement enregistrer dans le dossier actif.



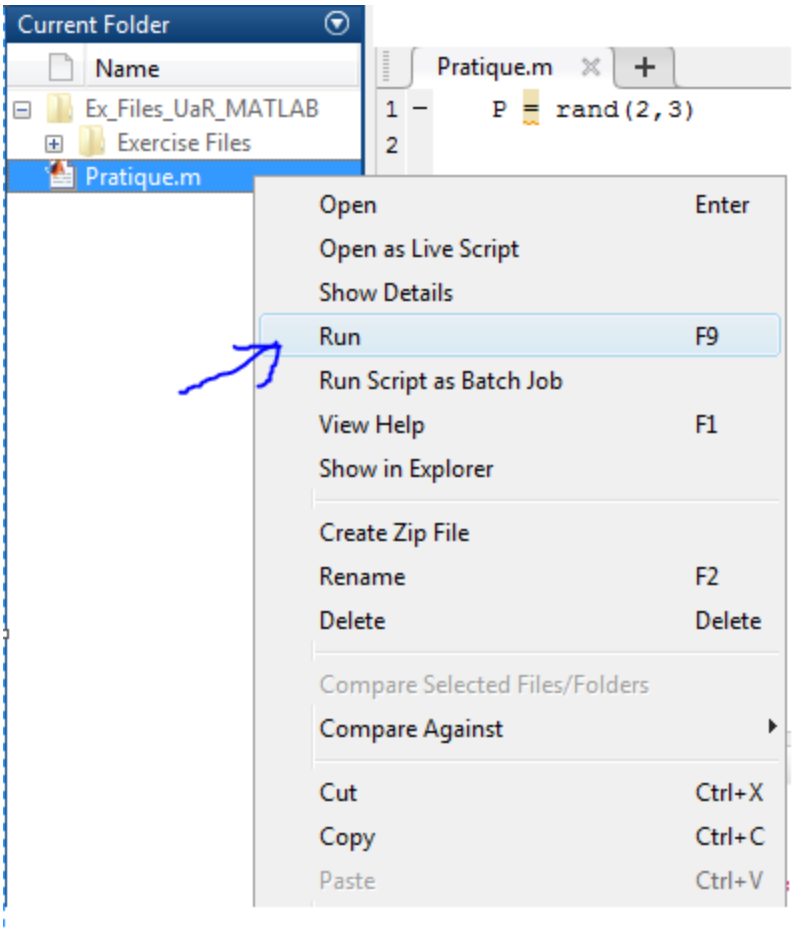
Maintenant, chaque fois que vous voulez revenir au script pour le modifier, il suffit de double-cliquer sur le nom dans le volet des dossiers en cours et cela va rouvrir la fenêtre et permettre l'édition du script exactement comme avant.



Pour exécuter le programme, il existe plusieurs options différentes. Vous pouvez d'abord cliquer sur le bouton Exécuter dans la fenêtre d'édition. Cela exécute le programme immédiatement sans fermer la fenêtre, vous permettant de voir rapidement et facilement comment les modifications apportées au programme affectent la sortie.



Deuxièmement, vous pouvez cliquer avec le bouton droit de la souris sur le script à partir de la fenêtre du dossier en cours. Et choisissez Exécuter ou appuyez sur F9.



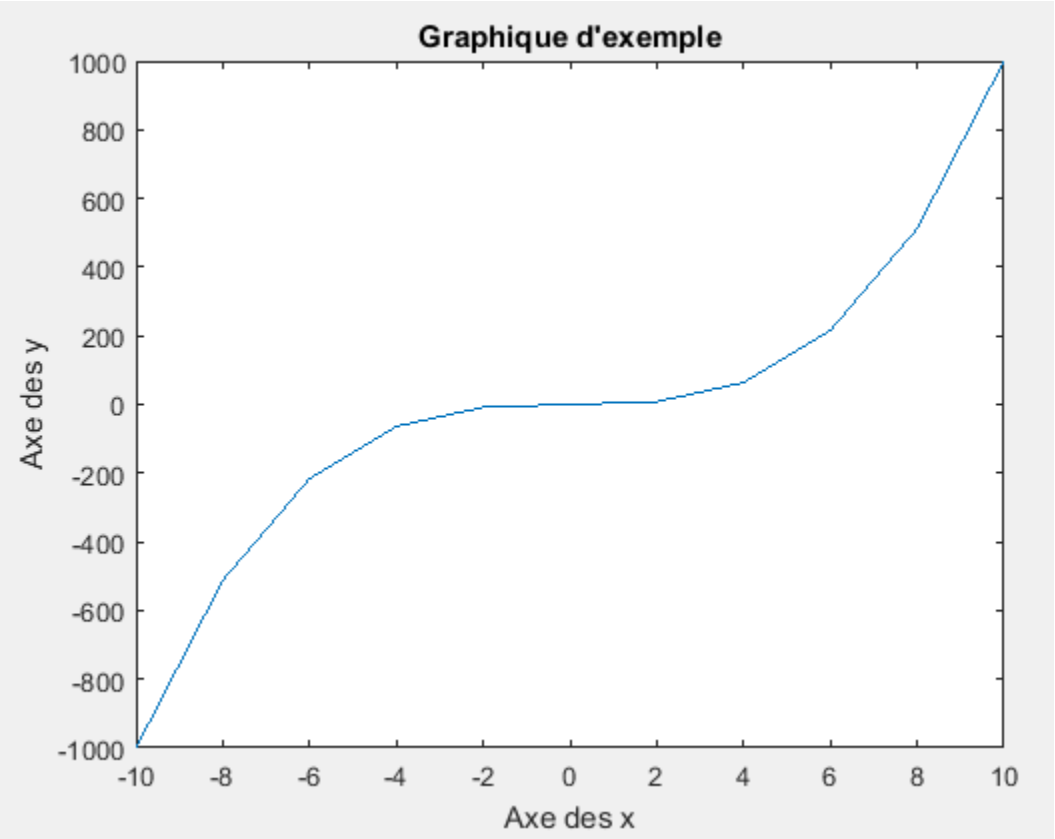
Création de graphiques de bases

Examinons les moyens d'afficher les données dans MATLAB. La fonction de base pour tracer un graphe à deux dimensions dans MATLAB est la fonction *plot*. Il existe une grande variété de syntaxes pour cette fonction, vous permettant de changer beaucoup des paramètres au fur et à mesure de sa création. Le tracé le plus simple prend un seul vecteur de données et trace les valeurs de données par rapport au nombre d'index des données dans le vecteur : *plot(x,y)*

Ajout d'annotations

Pour ajouter des informations sur un graphique comme le titre, les axes,… il suffit de taper les fonctions *xlabel*, *ylabel* ou *title* suivie de l'information à entrer.

```
x = -10:2:10;
y = (x.^3);
plot(x,y)
xlabel('Axe des x');
ylabel('Axe des y');
title('Graphique d''exemple');
```



Appendix II: Introduction aux matrices

Une matrice est une disposition rectangulaire de nombres en lignes et en colonnes. Par exemple, la matrice A a deux lignes et trois colonnes.

$$A = \begin{bmatrix} -2 & 5 & 6 \\ 5 & 2 & 7 \end{bmatrix}$$

Dimensions des matrices

Les dimensions d'une matrice indiquent sa taille: le nombre de lignes et de colonnes de la matrice, dans cet ordre.
Puisque la matrice A a deux lignes et trois colonnes, nous écrivons sa dimension en 2x3.
En revanche, la matrice B a 3 lignes et 2 colonnes donc c'est une matrice 3x2.

$$B = \begin{bmatrix} -8 & -4 \\ 23 & 12 \\ 18 & 10 \end{bmatrix}$$

Éléments d'une matrice

Un élément matriciel est simplement une entrée matricielle. Chaque élément d'une matrice est identifié en nommant la ligne et la colonne dans laquelle elle apparaît.
Par exemple, considérons la matrice G:

$$G = \begin{bmatrix} 4 & 14 & -7 \\ 18 & 5 & 13 \\ -20 & 4 & 22 \end{bmatrix}$$

L'élément G(3,1) est l'entrée dans la troisième ligne et la première colonne qui est -20

Opérations des matrices

- Addition

Pour additionner deux matrices de même dimension, additionnez simplement les entrées dans les positions correspondantes.
Rappelez-vous, les matrices doivent avoir **la même dimension**, ce qui signifie le même nombre de lignes et de colonnes.

Exemple:

$$\begin{bmatrix} 3 & 7 \\ 2 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 2 \\ 8 & 1 \end{bmatrix} = \begin{bmatrix} 3+5 & 7+2 \\ 2+8 & 4+1 \end{bmatrix}$$

$$= \begin{bmatrix} 8 & 9 \\ 10 & 5 \end{bmatrix}$$

- Multiplication

a) Scalaire par matrice

Le terme de multiplication scalaire se réfère au produit d'un nombre réel et d'une matrice. Dans la multiplication scalaire, chaque entrée dans la matrice est multipliée par le scalaire donné.

$$2 \cdot \begin{bmatrix} 5 & 2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 2 \cdot 5 & 2 \cdot 2 \\ 2 \cdot 3 & 2 \cdot 1 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & 4 \\ 6 & 2 \end{bmatrix}$$

b) Produit de deux matrices

La multiplication matricielle se réfère au produit de deux matrices. C'est une opération totalement différente. Vous pouvez multiplier deux matrices **si et seulement si**, le nombre de colonnes dans la première matrice est égal au nombre de lignes dans la deuxième matrice. Sinon, le produit de deux matrices est indéfini.
Les dimensions de la matrice du produit sont (lignes de première matrice) × (colonnes de la deuxième matrice)

Par exemple, si nous multiplions une matrice 2 × 3 par une matrice 3 × 1, la matrice produit est 2 × 1

$$\underset{2 \times 3}{\begin{vmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \end{vmatrix}} \times \underset{3 \times 1}{\begin{vmatrix} t11 \\ t21 \\ t31 \end{vmatrix}} = \underset{2 \times 1}{\begin{vmatrix} M11 \\ M21 \end{vmatrix}}$$

Here is how we get M_{11} and M_{12} in the product.

$M11 = r_{11} \times t_{11} + r_{12} \times t_{21} + r_{13} \times t_{31}$

$M21 = r_{21} \times t_{11} + r_{22} \times t_{21} + r_{23} \times t_{31}$