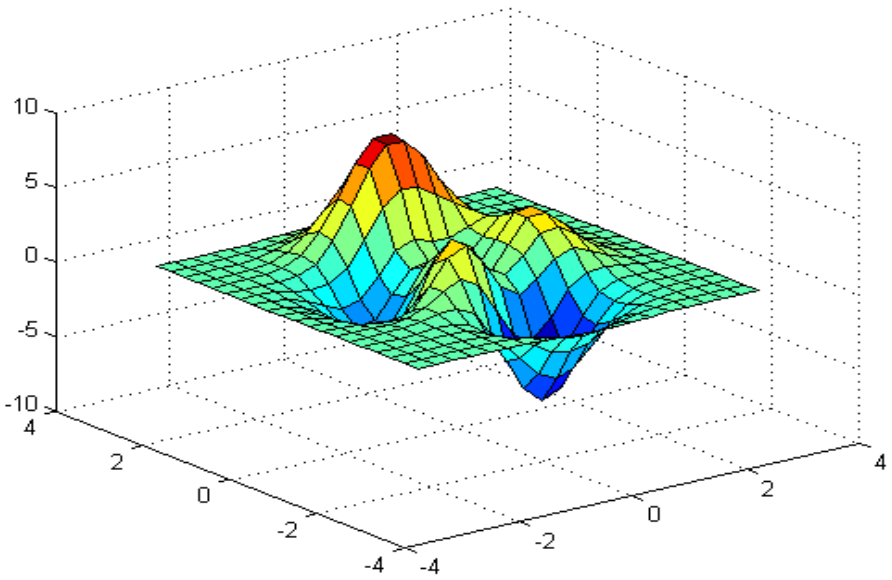
Intro à MATLAB -Les TDs



Thelma Coyle
I.S.M.

04 91 17 22 69

thelma.coyle@univ-amu.fr

9-11 mars 2016

Cours 1 Les Matrice et les fonctions de Base

Exo 1 : manipulation des indices

Dans un script qui s'appelle **exo1.m** effectuez les opérations suivantes :

1 Créer un scalaire, **N** qui est égale à 100 ;
2 Créez un vecteur (en ligne), **vecA**, qui contient les chiffres de 1 à N avec un pas de 1.
3 Créez un vecteur, **vecB** (en ligne), qui a N valeurs aléatoire (utiliser la fonction **rand**)
4 Créez un autre vecteur, **vecC**, comprenant les valeurs de **vecA** dont les indices sont des valeurs impaires
5 Créez un autre vecteur, **vecD**, comprenant les valeurs de **vecB** dont les indices sont des valeurs paires
6 Ajoutez 10 à tous les chiffres de **vecD** et stockez les résultats dans un vecteur **vecE**.
7 Effectuer la addition (terme à terme) de vecC à vecD, stockez les résultats dans un vecteur **vecF**.
8 Combinez les quatre vecteurs **vecC, vecD, vecE et vecF** dans une matrice 2D qui s'appelle **M** en mettant **vecC** sur la première colonne, **vecD** sur la deuxième etc.
9 Effectuez l'addition des éléments suivants (de **M**) : 1ère ligne, 2ème col : 2ème ligne, 1ère col : 3ème ligne, 4ème col ; et captez le résultat dans un variable **sum3**.
10 Supprimer toute la dernière ligne de **M**.
11 La commande **size** permet d'évaluer la taille d'une matrice. Il y a autant de valeurs dans la sortie de cette commande qu'il y a dimensions dans la matrice d'entrée. Attribuez le variable **numcols** au nombre de colonnes dans **M**.
12 En faisant appel à la commande **sum** de Matlab, effectuez la somme de toutes les colonnes en **M** et captez les valeurs dans un vecteur **sumM**.
13 En utilisant la commande **max** de Matlab, trouver la valeur maximale de la première colonne de M. - stockez la dans le variable **maxCol1**.
14 Déterminez la valeur maximale dans toute la matrice **M** et stockez-la dans **maxAll**.

Exo 2 : création d'une fonction

Dans l'éditeur créez un nouveau fichier, **exo2.m** et là-dedans copiez toutes les lignes de code du fichier **exo1.m**. Ce nouveau .m va doit fonctionner comme une fonction (qui s'appelle exo2) qui prend un variable d'entrée (le variable **N**) et qui renvoie les quatre résultats (**numcols, sumM, maxCol1** et **maxAll**) comme variables de sortie

Exo 3 : manipulation des matrices

1. Si A = rand(6,3), comment peut-on trouver la valeur moyenne de toute la matrice A ?

2. Comment déterminer les max des lignes ?

3. La commande **std** donne les écart-types colonne par colonne. Vérifiez que c'est bien le cas avec une matrice comme A. Comment peut-on trouver l'écart-type de toutes les valeurs dans A. ?

4. Si x est le vecteur rand(10,1) calculez la somme des x carrés

5. Si A = rand(4) (i.e., matrice carrée), déterminez le nombre d'éléments de A supérieur à 0.5.

Exo 4 : analyse simple de données

Le fichier Data.mat comprend des données (chaque colonne représente un essai) acquises à 100 Hz à partir d'un potentiomètre placé sur le poignet. Le fichier a été sauvegardé en format binaire, (propriétaire Matlab), on ne peut pas l'ouvrir avec Excel etc. Pour tous les essais, le sujet a dû effectuer un mouvement rythmique. Il y avait quatre conditions différentes.

Dans un script qui s'appelle AnalyseSimple.m :

- D'abord vous allez charger les données dans Matlab en tapant >> **load Data.mat** ; Dans le workspace vous devriez voir apparaître un variable **AllData** qui est une matrice de 25 colonnes et 1700 lignes.

- la première ligne définit les conditions des essais (Condition # 1 , 4, 6 et 8). La condition #1 représente le repos. Faites afficher dans la fenêtre de commandes la première ligne de la matrice **AllData** pour voir la séquence des conditions. Créez un variable (qui s'appelle **conditions**) qui ne contient que la première ligne de **AllData**.

- Les données expérimentales occupent ligne 2 jusqu'à1700 de **AllData**. Créez une autre matrice qui s'appelle **DataOnly** et qui contient toutes les données expérimentales de **AllData**.

- Déterminez l'amplitude moyenne de chaque essai. – stockez les valeurs dans un vecteur qui s'appelle **moy**.

- Déterminez l'amplitude max de chaque essai. – stockez les valeurs dans un vecteur qui s'appelle **maxval**.

- A partir du vecteur **conditions**, groupez les essais de chaque condition pour déterminer la valeur moyenne par condition.

- Dans le même esprit, déterminez la moyenne des valeurs maximales par condition.

Modifiez le script pour créer une fonction qui reçois la matrice de donnée comme paramètre d'entrée et qui renvoie les valeurs moyennes et maximales par conditions en sortie

Exo 5 : manipulation des chaines

Dans un script (**chaines.m**) effectuez les manipulations suivantes :

1. Avec la commande **input** faites entrer une chaîne de caractères dans le workspace (mémoire) de Matlab sous le nom **caract**
2. Evaluez la longueur de la chaîne, associez le résultat à un variable **long.**
3. Ajoutez le caractère 's' à la fin de la chaîne dans **caract**.
4. Identifiez le nombre d'occurrence de la lettre 'a' dans la chaîne.
5. Modifiez votre script pour créer une fonction (chainesF.m) qui accepte une chaîne de caractères comme entrée et qui renvoie le nombre de la lettre 'a' comme sortie

Cours 2 La Programmation

Exo 6 : Les Boucles FOR et WHILE, manipulation des chaines de caractères

Souvent on peut effectuer la même opération avec une boucle FOR et une boucle WHILE. Vous allez utiliser les deux possibilités pour compter le nombre d'occurrence d'une certaine lettre dans une chaîne de caractères. Cette double-opération doit être effectuée dans une fonction qui porte le nom **countletter** et qui accepte deux variables d'entrée-la chaîne à analyser et la lettre à compter, et qui renvoie le nombre d'occurrence de la lettre comme sortie. Choisissez les noms des variables vous-même. Effectuez le comptage d'abord par la boucle FOR et après par la boucle WHILE. Si les deux réponses sont égales, renvoyez cette valeur comme sortie (Si tout marche bien, les deux réponses sont forcément égales !). S'il n'y a pas de lettre clef dans la chaîne, la valeur renvoyée doit être zéro.

Ce comptage peut être fait également sans utiliser des boucles. Réfléchissez sur l'algorithme. (Il vous faut peut-être une des fonctions de base qui traitent les chaînes)

Exo 7 : Boucles, fonction

Ecrivez une fonction qui s'appelle **howmanyNum** qui détermine le nombre d'élément d'un vecteur qu'il faut sommer pour juste dépasser un seuil. Le vecteur et le seuil sont tous les deux passés comme variables d'entrée et le nombre d'éléments est renvoyé comme sortie. Est-ce que vous avez la possibilité d'utiliser les deux types de boucles pour créer cette fonction ?

Exo 8 : manipulation de chaine, input, Boucle IF

Ecrivez un script, **lookforChar**, qui trouve d'abord la position de toutes les occurrences d'un certain caractère dans une chaîne de caractères. Faites entrer la chaîne à analyser par la commande **input** en spécifiant que l'information à récupérer est bien une chaîne. Faites pareil pour le caractère à rechercher. Utiliser la fonction **strfind** pour ensuite identifier les indices des occurrences de la lettre clef. Ensuite (s'il y a bien des lettres clef dans la chaîne – testez en utilisant la fonction logique **isempty)** remplacez la dernière occurrence le la lettre par une étoile/astérix *.

Exo 9 : suppression de caractère, boucles

Vous allez écrire une fonction, qui s'appelle **remove2,** qui enlève les doublements de caractères dans une chaîne. La fonction accepte une entrée, la chaîne à analyser elle renvoie la chaîne modifiée comme sortie. L'algorithme est telle que' s'il y a deux lettres à la suite qui sont les même, il faut enlever la deuxième. (N'essayez pas pour l'instant de prévoir les cas de triples lettres etc....)

Exo 10 : boucle FOR, script ->fonction

- Créer un scripte qui normalise un vecteur par rapport au max.
- Modifiez votre algorithme afin de normaliser des matrices 2D colonne par colonne.
- A partir de votre scripte, créez une fonction.

Exo 11 : Trier des Noms, les boucles, les caractères, les fonctions sort, input

Vous allez écrire un script qui trie un ensemble de noms (qui comprennent tous 4 lettres) en ordre alphabétique (pour l'instant, en ne tenant compte que de la première lettre).

Algorithme :

- D'abord le script demande à l'utilisateur combien de noms il va falloir traiter – utilisez la fonction **input** pour faire entrer un chiffre de la fenêtre de commandes.
- Configurer une boucle qui demande à l'utilisateur de faire entrer le bon nombre de noms. Si possible (!) veuillez que les noms comprennent tous 4 lettres, pour qu'on puisse remplir une matrice, chaque ligne correspond à un nom.
- Si la première lettre du nom est en minuscule, remplacer la par la majuscule (utilisez les codes ASCII)
- Utiliser la fonction **sort** de matlab pour trier les premières lettres par ordre alphabétique. **Sort** renvoie deux sorties, un vecteur trié et un vecteur d'indices qui indiquent l'emplacement de chaque élément dans le vecteur d'origine. Ce deuxième paramètre de sortie peut vous faciliter la vie pour ordonner votre matrice de noms.

Exo 12 : modification d'une chaine

Dans un fichier **inverseC.m**, écrivez une fonction qui inverse les caractères dans une chaîne. La fonction accepte une chaîne comme entrée (de n'importe quelle longueur) et renvoie la chaîne inversée comme sortie.

Cours 3 Les Graphiques

Exo 13 : Manipulation des figures graphiques et des axes, appel à une fonction

Vous allez écrire un scripte qui affiche les données qui se trouvent dans le fichier

graphdata.dat. Pour charger ces données, utilisez la commande **load** avec la syntaxe suivante :

```
>> D = load(‘ graphdata.dat’)
```

Les données vont être chargées dans le mémoire de Matlab sous le nom **D**. Ce variable est une matrice de trois colonnes ; chaque colonne représente un essai.

1. Affichez les trois colonnes sur les mêmes axes dans une seule fenêtre
2. Affichez les trois colonnes dans trois différentes fenêtres
3. Affichez les trois colonnes sur trois différents systèmes d’axes dans une seule fenêtre.

Comme vous pouvez constater, les données représentent un signal oscillant. La fonction FiltMaxmin est fournie pour traiter ces données. Cette fonction applique un filtre (butterworth, ordre 2) à un vecteur d’entrée, et ensuite détermine les indices des points de rebroussement (les max et les mins). Il y a trois variables de sortie : les données filtrées, un vecteur des indices des max et un vecteur des indices des mins. (Ouvrez la fonction dans l’éditeur si vous voulez plus de précisions)

4. Faites appel à FiltMaxmin pour essai N° 1
5. Affichez le signal filtré sur une des figure déjà crée (voir étapes 1, 2, ou 3 ci-dessus)
6. Ajoutez les points de rebroussement avec un symbole (o,*,+ etc et aucune ligne), et si possible différentiez entre les max et les mins (symboles ou couleurs différents).
7. Modifiez votre scripte pour traiter les trois essais (si possible sans copier deux fois plus les lignes de code d’origine) .

(L’ordre de ces 7 étapes dans votre scripte ne correspond pas forcement à l’ordre de la question)

Cours 4 Les Fichiers

Exo 14 : Lire un fichier ASCII.

Dans le directory des fichiers fournis, il se trouve un fichier qui s'appelle ex_data.txt. La première ligne comprend un commentaire (3 mots) qui est suivi par deux colonnes des données. Les données sont des valeurs réelles (flottantes) et il y en a 10 dans chaque colonne. Qu'est-ce que c'est le commentaire? Evaluez la moyenne de chaque colonne.

Exo 15 : Lecture d’un fichier ASCII/ manipulation de matrices/affichage graphique

Vous allez écrire une fonction qui lit le fichier manipData.txt et qui organise les données pour effectuer des calculs simples. Le nom du fichier de donnée sera passé comme le seul paramètre d’entrée. Il y aura deux paramètres de sortie (voyez en dessous)

Le fichier contient huit colonnes. Chaque colonne contient un en-tête (titre, sur la première ligne) d’un mot de 5 caractères. Effectuez la lecture de la première ligne en associant chaque titre avec une ligne d’une matrice de caractères, que vous pouvez appeler comme vous voulez. Essayez d’éviter de répéter les mêmes commandes huit fois

Ensuite, lisez les données (dont il y a 8 colonnes mais on ne sait pas le nombre de lignes) et associez les valeurs avec un variable qui s’appelle data, et qui respecte l’organisation des données dans le fichier (en colonne). Chaque colonne représente les données d’un essai. Affichez ces huit essais dans une figure graphique (dont l’organisation reste à vous de gérer)

Dans deux vecteurs maxALL et minALL calculez les valeurs maximales et minimales de chaque essai. Dans une troisième vecteur, maxAMP, déterminez la différence entre la valeur max et min de chaque essai. Ces trois vecteurs sont les trois variables de sortie de la fonction.

Exo 16 : Exo Finale

Le but finale est d'effectuer une analyse simple sur trois fichiers de données (Sujet1.txt, Sujet2.txt et Sujet3.txt) – les données représentent trois essais et montrent les déplacements verticales (mm) de la tête pendant l'initiation d'un pas. La fréquence d'acquisition était 100Hz. Chaque fichier de données comprend une seule colonne de données

Une première version de l'analyse est fournie – ouvrez le fichier analyseDat1.m pour voir le code. Le nom du fichier à analyser est écrit explicitement dans le scripte, donc pour traiter un des autres fichiers il faut éditer le code. Après la lecture, il y a un affichage et un filtrage des données. Ensuite, on évalue la valeur du déplacement moyen et l'écart-type pendant la première demi-seconde ; pendant cette période initiale, le sujet est censé d'est dans un état stable. Ces valeurs sont utilisées pour identifier le début du mouvement – le bras est en mouvement dès que son déplacement diffère de son état stable par deux fois l'écart-type. Pour finir, on identifie la latence entre le début du mouvement et le point minimum (excursion maximale a droit)

Vérifiez que le scripte marche comme il faut et prenez le temps de comprendre l'algorithme. Modifiez ce code pour pouvoir faire entrer le nombre de fichiers à traiter et ensuite le code se débrouille pour effectuer le même analyse sur tous les fichiers. A la fin le variable d'intérêt sera un vecteur qui contient les valeurs de latence de tous les essais.

Des Questions 'Supplémentaires'

Exo 17 : Armstrong Numbers

An '**Armstrong Number**' is an integer such that the sum of the cubes of its digits equals the number itself.
For example, 407 is an Armstrong number since
4^3 + 0^3 + 7^3 = 64 + 343 = 407

Write a script (or a function) which finds all Armstrong numbers between 100 and 999… Simply display in the command window the numbers which you find. Your algorithm will probably be based on one of the following
Approach N° 1: you could decide to rely on converting each number in the given range into a string of three characters and then converting each character in turn into a single-digit number. You will probably need the functions **num2str** and **str2num**
Approach N°2: generate the full range of numbers within nested loops, - one each for the hundreds, tens and units.
Both approaches will need a short calculation and a final test on the result to see if the sum equals the original value

Exo 18 : **<u>Une question de déchiffrage</u>**

Voici une fonction MATLAB (elle est fournie parmi les fichiers prévus pour le cours)

```
function answer = mystery (test_value)
%
%
%
%
%
%
%
%

 sum = 0;

 for d = 1 : test_value - 1
        div_result = test_value / d;

        if (div_result = = floor(div_result))

                sum = sum + d;
        end
 end

 if (sum == test_value)
   answer = 1;
 else
   answer = 0;
 end
```

Que fait cette fonction ??? Voici quelques questions que peuvent vous orienter sur la bonne piste

1. Quelles formes prennent les paramètres d'entrée et de sortie ? (scalaires, vecteurs, matrice ?). Pour le paramètre de sortie, quelles sont les valeurs possibles ?

2. Dans la première condition IF (ligne #18, **if (div_result = = floor(div_result))** expliquez qu'est-ce qu'on teste. Si la condition est VRAI, qu'est-ce qu'on fait ? Si cette condition est FAUSSE qu'est-ce qu'on fait ?

3. Si le paramètre d'entrée est égale à 6, combien de fois s'est effectuée la boucle FOR, et est-ce que la condition IF (ligne #18) est bien VRAI pour la première itération ?

4. Pour deux différentes valeurs du paramètre d'entrée (**test_value** = 6 et **test_value** = 8) quels sont les valeurs finals de **sum** et **answer** ?

5. Il manque des commentaires dans cette fonction, surtout au début pour décrire le but global de la fonction. Ayant lu tous le code, écrivez quelques lignes de commentaire à mettre tout au début, qui expliquent <u>ce que fait</u> cette fonction. (et non pas comment elle le fait)