-Feuille de TP N°03 – Scripts et Fonctions

Le but de ce TP est d'apprendre aux étudiants comment écrire et utiliser des scripts et des fonctions en Matlab et comprendre les cas d'utilisation de chaque concept dans le but de pouvoir écrire des programmes entiers et résoudre des tâches de plus en plus complexes.

Rappel de cours :

- Qu'est-ce qu'un script ?
  Un script est un fichier de code source contenant une série d'instructions généralement utilisé pour effectuer une tâche spécifique d'une manière automatisée. Il peut inclure des variables, des boucles, des conditions et des fonctions.
- Quelle sont les différences principales entre un script et une fonction ?
  Un script est simplement un fichier contenant une séquence d'instructions Matlab, tandis qu'une fonction prend des arguments en entrée, effectue un traitement spécifique, et renvoie des valeurs en sortie. Un script peut être exécuté directement, contrairement à une fonction qui doit être appelée en utilisant son nom suivi des arguments nécessaires pour son exécution.
- Combien de sortie peut avoir une fonction dans Matlab ?
  Il n'y a pas de limite aux nombre de variables de sortie qu'une fonction Matlab peut retourner (contrairement à la majorité des langages de programmation), es fonctions Matlab peuvent donc retourner plusieurs valeurs.
- Les paramètres d'entée d'une fonction sont-ils passés par valeur, ou par variable ? Qu'est-ce que cela implique ?
  Tous les paramètres d'entrée d'une fonction Matlab (y compris les tableaux) sont passés par valeur, ce qui implique que le changement des paramètres dans le corps de la fonction n'aura aucun effet sur les paramètres effectifs passés lors de l'appel.
- Que font les fonctions « num2str » et « strcat » ?

  Nous utilisons ces deux fonctions pour manipuler les chaînes de caractères.

  - **La fonction "strcat" permet de concaténer plusieurs chaînes de caractères en une seule. Elle prend en entrée un ou plusieurs arguments de type chaîne de caractères et renvoie une chaîne de caractères résultante qui est la concaténation de tous les arguments.**
  - **La fonction "num2str" permet de convertir un nombre en chaîne de caractères. Elle prend en entrée un nombre et renvoie une chaîne de caractères correspondante.**

  Remarque importante : ce n'est qu'à partir de la version 2016b de Matlab qu'il est devenu possible de concaténer les chaînes de caractère avec le « + » (ça marche même pour les chaîne de caractères avec les valeurs numériques), à l'université on utilise la version 2007, donc il faut concaténer les chaînes de caractères avec « strcat » après avoir converti toutes les variables numériques en chaînes de caractères également. Si un étudiant utilise l'opérateur « + » plutôt que « strcat » dans une micro-interrogation ou un examen considérez que sa réponse est juste.

Exercice 01 : premiers scripts

- Ecrire un script qui demande à l'utilisateur de saisir deux valeurs « a » et « b », puis calcule et affiche la somme des carrés des deux nombres ($a^2 + b^2$).

```
a = input('donner la valeur de a : ');
b = input('donner la valeur de b : ');
c = a ^ 2 + b ^ 2;
display(c);
```

- Ecrire un script Matlab qui demande à l'utilisateur de saisir une distance en kilomètres, puis il la convertit et l'affiche en miles (mettons 1 mile = 1.61 kilomètres).

```
a = input('donner une distance en kilometres: ');
b = a / 1.61;
display(b);
```

- Ecrire un script Matlab qui demande à l'utilisateur de saisir une température en degrés Celsius, puis la convertit en degrés Fahrenheit et affiche le résultat, en sachant que :

$$F = C \times \frac{9}{5} + 32$$

```
c = input('donner la température en Celsius : ');
f = c * 9 / 5 + 32;
display(f);
```

- Ecrire un programme Matlab qui permet à l'utilisateur de saisir le rayon d'un cercle, puis calcule et affiche son périmètre et sa surface.
  - Le programme doit afficher les résultats dans le format suivant : « Pour un rayon qui vaut **x** le périmètre est **y** et la surface est **z** ».

```
r = input('donner le rayon du cercle : ');
p = r * 2 * pi;
s = r ^ 2 * pi;
display(strcat('Pour un rayon qui vaut ', num2str(r), ' le
perimetre est ', num2str(p), ' et la surface est ',
Num2str(s)));
```

Exercice 02 : structures conditionnelles

- Ecrire un script Matlab qui demande à l'utilisateur de saisir un nombre entier, puis affiche un message indiquant si le nombre saisi est « pair » ou « impair ».
  - **Indication :** utiliser la fonction ***mod*** ou ***rem*** pour calculer le reste de division, la commande ***help*** pourrait être utile pour avoir plus d'informations.

```
n = input('Donner un nombre entier : ');
if mod(n, 2) == 0
      display('Nombre pair');
else
      display('Nombre impair');
end
```

- Ecrire un script Matlab qui permet à l'utilisateur de saisir trois nombre, puis calcule et affiche leur « min » et « max ».

```
a = input('Donner le premier nombre : ');
b = input('Donner le deuxieme nombre : ');
c = input('Donner le troisieme nombre : ');
```

```
if a > b
      min = b;
      max = a;
else
      min = a;
      max = b;
end
if c > max
      max = c;
end
if c < min
      min = c;
end
```

- Écrire un script qui prend en entrée une année et qui affiche si cette année est bissextile ou non. Une année est bissextile si elle est divisible par 4 mais non divisible par 100, ou si elle est divisible par 400.

```
a = input('Donner une annee : ');
if (mod(annee, 4) == 0 && mod(annee, 100) ~= 0)
      || mod(annee, 400) == 0
    disp('Annee bissextile');
else
    disp('Annee non bissextile');
end
```

Exercice 03 : les boucles

- Écrire un script qui prend en entrée un nombre entier « n » entre « 1 » et « 9 », puis qui affiche la table de multiplication de « n » jusqu'à 10.
    - Si la valeur de « n » et inférieure à « 1 » ou supérieure à « 9 », il faut demander à l'utilisateur de donner une autre valeur jusqu'à ce qu'il tombe dans le bon intervalle.
    - **Indication :** pensez à utiliser « strcat » et « num2str » pour afficher la table de multiplication dans le bon format.

```
n = 0;
while n < 1 || n > 9
    n = input('Saisissez un nombre entre 1 et 9 : ');
end
for i = 1:10
    disp(strcat(num2str(n), ' x ', num2str(i), ' = ',
num2str(n*i)));
end
```

- Ecrire un script Matlab qui demande de saisir un entier positif, puis dire si le nombre saisi est un nombre parfait ou pas. Un nombre est parfait s'il est égal à la somme de ses diviseurs propres (tous les diviseurs sauf le nombre lui-même).

```
n = input('Saisissez un entier positif : ');
sum = 0;
```

```
for i = 1:n-1
    if mod(n, i) == 0
        sum = sum + i;
    end
end
if sum == n
    disp('Nombre parfait');
else
    disp('Nombre non parfait');
end
```

- Ecrire un script qui permet de saisir un nombre entier positif, puis calcule et affiche la somme des chiffres qui le composent.
    - **Indication :** il n'y a pas de division entière dans Matlab, il faut donc arrondir le résultat de la division réelle.

```
n = input('Saisissez un entier positif : ');
S = 0;
while n > 0
    S = S + rem(n, 10);
    n = floor(n/10);
end
disp(S);
```

- Ecrire un script qui permet à l'utilisateur de remplir une matrice, puis calculer et afficher sa transposée, et sa rotation de 90° à droite. Le script doit suivre les étapes suivantes :
    - Demander le nombre de lignes et colonnes de la matrice à l'utilisateur.
    - Faire une boucle pour lire les valeurs de la matrice.
    - Calculer la transposée et la rotation dans deux matrices différentes.
    - Afficher les résultats.
    - **Attention :** vous n'avez pas le droit d'utiliser l'opérateur de transposition « ' » ni la fonction « rot90 », il faut utiliser les boucles.

```
l = input('Nombre de lignes : ');
c = input('Nombre de colonnes : ');
for i = 1:l
    for j = 1:c
        M(i, j) = input(strcat('Case (', num2str(i), ' ,',
num2str(j), ') : '));
    end
end
for i = 1:l
    for j = 1:c
        transpose(j, i) = M(i, j);
    end
end
for i = 1:l
    for j = 1:c
        rotation(c - j + 1, i) = M(i, j);
    end
end
```

```
disp('La matrice originale : ');
disp(M);
disp('La transposee : ');
disp(transpose);
disp('La rotation de 90 degres a droite : ');
disp(rotation);
```

Exercice 04 : déclarer et utiliser des fonctions

- Ecrire une fonction « premier » qui prend comme argument un nombre entier positif, et qui retourne « 1 » si son paramètre est premier, et « 0 » sinon.

```
function p = premier(n)
    p = 1;
    if n == 1
        p = 0;
    else
        % Il est possible d'arrêter la boucle à n / 2
        % ou même à sqrt(n)
        for i = 2:n-1
            if mod(n, i) == 0
                p = 0;
            end
        end
    end
end
```

  - Ecrire un script Matlab qui demande à l'utilisateur de saisir une valeur « n », puis calcule et affiche tous les nombres premiers qui sont strictement inférieurs à « n ».

```
n = input('Entrez un entier positif : ');
for i = 1:n-1
    if premier(i) == 1
        disp(i);
    end
end
```

- Ecrire une fonction Matlab qui prend comme paramètre un vecteur « a », et qui retourne deux vecteurs « x » et « y » tel que « x » contienne les éléments strictement positifs de « a », et « y » les éléments strictement négatifs (les valeurs nulles n'apparaissent sur aucun des résultats).

```
function [x, y] = separateur(a)
    x = [];
    y = [];
    for i = 1:length(a)
        if a(i) > 0
            x = [x a(i)]; % ou x(end + 1) = a(i);
        end
        if a(i) < 0
            y = [y a(i)]; % ou même y(length(y) + 1) = a(i);
        end
    end
end
```

- Initialiser un vecteur et tester votre fonction à partir de la fenêtre de commandes.

```
>> v = [1 -2 3 0 4 -5]; % Ou n'importe quelles valeurs.
>> [x y] = separateur(v);
```

- Ecrire une fonction Matlab qui prend en entrée un vecteur « T », et qui retourne un tableau « S » de la même taille, tel que « S[i] » soit le nombre d'éléments de « T » qui sont strictement inférieurs à « T[i] ».
  - **Indication :** décomposez le problème en deux fonctions pour vous faciliter le travail.

```
function n = compter_inferieurs(T, v)
    % Une fonction qui compte les valeurs inférieurs v dans T.
    n = 0;
    for i = 1:length(T)
        if T(i) < v
            n = n + 1;
        end
    end
end

function S = compter_vecteur(T)
    for i = 1:length(T)
        S(i) = compter_inferieurs(T, T(i));
    end
end
```

Exercice 05 : faites votre mieux

- Ecrire une fonction Matlab « appartient » qui prend comme paramètre un vecteur « T » et une valeur « x », et qui retourne « 1 » si « x » appartient à « T » et « 0 » sinon.

```
function a = appartient(T, x)
    a = 0;
    for i = 1:length(T)
        if T(i) == x
            a = 1;
        end
    end
end
```

- Modifier la fonction précédente pour qu'elle retourne le nombre d'occurrences de « x » ainsi que l'indice de chaque occurrence.
  - Si « x » n'appartient pas à « T », l'indice d'occurrences retourné sera un vecteur vide.

```
function [n, indices] = nombre_occurrences(T, x)
    indices = [];
    n = 0;
    for i = 1:length(T)
        if T(i) == x
            n = n + 1;
            indices(end + 1) = i; % indices = [indices, i];
        end
    end
end
```

- Ecrire une fonction Matlab « distinct » qui prend en entrée un vecteur « T », et qui retourne un autre vecteur contenant les valeurs distinctes de « T » (les valeurs répétées plusieurs fois dans « T » n'apparaissent qu'une seule fois dans le résultat).

```
function res = distinct(T)
    res = [];
    for i = 1:length(T)
        if ~appartient(res, T(i))
            res = [res T(i)]; % Ou res(end + 1) = T(i);
        end
    end
end
```

- Écrire une fonction qui prend en entrée une matrice « M », et qui retourne deux vecteurs « R » et « S », tels que « R » soit le tableau des éléments distincts de « T », et que « S[i] » le nombre d'occurrence de « R[i] » dans « T ».

```
function [R, S] = occurenceMatrice(M)
    R = [];
    S = [];
    [l c] = size(M);
    for i = 1:l
        for j = 1:c
            [n, ind] = nombre_occurrences(R, M(i, j))
            if n == 0
                R(end+1) = M(i);
                S(end+1) = 1;
            else
                S(ind(1)) = S(ind(1)) + 1;
            end
        end
    end
end
```