TP n° 2 - Un peu plus de boucles

Tous les sujets et les corrigés sont disponibles aux adresses suivantes :
`http://www.lif.univ-mrs.fr/~vpoupet/enseignement/matlab09.php`
`http://www.lif.univ-mrs.fr/~pvanier/?q=cours`

Exercice 1. *Pour se rafraîchir la mémoire*

La dernière fois, vous avez vu comment affecter des variables, créer et modifier des tableaux (vecteurs ou matrices, selon la dimension), écrire des scripts et faire des boucles permettant de répéter des instructions un certain nombre de fois.
Commençons par vérifier que vous savez encore faire ces choses-là...

1. Créez un tableau `tab` contenant les entiers pairs de 0 à 42, puis écrivez un script qui parcourt le tableau et remplace chacune des valeurs par son carré.

Réponse :

```
>> tab = 0:2:42;
```

puis dans un script :

```
for i = 1:size(tab,2)
  tab(i) = tab(i).^2;
end
```

La fonction `size` renvoie les dimensions d'une matrice. Ici `tab` est un tableau d'une ligne et 22 colonnes, mais seule la seconde dimension (nombre de colonnes) nous intéresse, d'où `size(tab, 2)` qui demande la taille de `tab` dans la deuxième dimension.

Exercice 2. *Fonctions*

Il existe de nombreuses fonctions pré-définies en *Matlab*, mais il arrivera forcément un moment où vous voudrez utiliser une fonction qui n'est pas définie. Fort heureusement, il est possible de définir ses propres fonctions et de s'en servir exactement comme les fonctions pré-existantes.
Supposons que l'on veuille définir la fonction `cos2` définie par

$$\cos 2 : \left\{ \begin{array}{ccc} \mathbb{R} & \rightarrow & \mathbb{R} \\ x & \mapsto & cos^2(x) \end{array} \right.$$

Il faut alors créer un nouveau fichier de script (il faut un fichier séparé pour chacune des fonctions que l'on définit), puis écrire

```
function r = cos2(x)
r = cos(x)^2
```

Il faut ensuite sauvegarder le script sous le nom `cos2.m`. Une fois que le script est sauvegardé, il est possible d'appeler la fonction directement dans *Matlab* par son nom.
Quelques points importants sur la définition des fonctions :
- le mot-clé `function` de la première ligne indique que le script est une définition de fonction ;
- la suite de la première ligne spécifie le nom de la fonction (ici `cos2`), le nombre d'arguments qu'elle prend et le nom qu'on leur donne dans la définition (ici un seul argument que l'on a appelé `x`) ainsi que le nom que l'on donne dans la définition qui suit au résultat (ici on a décidé de l'appeler `r`).

– le corps de la fonction (toutes les lignes après la première) peut être arbitrairement long et l'on peut y effectuer toutes les opérations *Matlab* exactement comme dans un script. Lorsque l'on appellera la fonction avec des arguments, toutes les lignes du corps seront exécutées et une fois l'exécution terminée, la valeur renvoyée par la fonction sera la valeur de la variable que l'on a annoncée comme étant le résultat (dans notre exemple `r`) ;
– le nom du fichier sous lequel vous enregistrez la fonction doit impérativement être le même que le nom de la fonction (suivi de l'extension `.m`).

1. Définissez la fonction `cos2` comme il a été indiqué, puis testez-la dans l'interpréteur sur quelques exemples.

2. Définissez la fonction `fibo` qui prend un argument n et renvoie la valeur du n-ième terme de la suite de Fibonacci vue au TP précédent (vous pouvez réutiliser les lignes du script de la dernière fois).

Réponse :

```
function y = fibo(n)
  fib = [1, 2];
  for i = 3:n
    fib(i) = fib(i-1) + fib(i-2);
  end
  y = fib(n);
```

Exercice 3. *Matrices*

Matlab est particulièrement adapté lorsque l'on veut faire des applications numériques sur des matrices. Voyons quelques méthodes permettant de les manipuler.

1. Définissez une matrice `M = [1 2; 3 4]` puis essayez les opérations suivantes dans l'interpréteur :

```
>> 2 * M + 3
>> M + M
>> sqrt(M)
>> M * M
>> M .* M
>> ones(4)
>> ones(3, 5)
```

Quelle est la différence entre les opérateurs `*` et `.*` ? Que fait la fonction `ones` ?

Réponse : L'opérateur `*` désigne le produit de deux matrices (comme vous l'avez vu en maths), tandis que `.*` désigne le résultat du produit terme à terme de deux matrices de mêmes dimensions : la case (i, j) du résultat est le produit des cases (i, j) de chacune des deux matrices de départ.
La fonction `ones` crée une matrice ayant les dimensions indiquées ne contenant que des 1.

2. Comment peut-on créer facilement une matrice 54×42 ne contenant que des 7 ?

Réponse :

```
>> M = 7 * ones(54, 42);
```

Si l'on veut créer une matrice $(a_{i,j})$ de dimensions 8×9 définie par $a_{i,j} = i + j$, on peut utiliser un script et des boucles `for`. Cependant dans ce cas une seule boucle ne suffit pas, et il va falloir en utiliser deux « imbriquées » : une première boucle qui parcourt les lignes de la matrice (les différentes valeurs de i) et une seconde boucle qui parcourt les colonnes (les valeurs de j).

3. En supposant que l'on ne s'intéresse qu'à une ligne `i` particulière de la matrice (donc ici `i` est une constante dont on suppose que la valeur est définie), écrivez une boucle `for` qui parcourt les différentes valeurs de j possibles (de 1 à 9 dans notre exemple) et pour chacune définit `a(i,j) = i+j`.

Réponse :

```
for j = 1:9
  a(i,j) = i+j;
end
```

4. Écrivez une boucle `for` qui pour chaque valeur possible de `i` (de 1 à 8) exécute la boucle précédemment écrite.

Réponse :

```
for i = 1:8
  for j = 1:9
    a(i,j)=i+j;
  end
end
```

5. Vérifiez que vous avez bien défini la matrice $(a_{i,j})$ de dimensions 8×9 telle que $a_{i,j} = i + j$.

Réponse :

```
>> disp(a);
```

6. En utilisant deux boucles, écrivez un script qui calcule la somme de deux matrices `A` et `B`, de mêmes dimensions $n \times m$.

Réponse :

```
for i = 1:n
  for j = 1:m
    C(i,j) = A(i,j) + B(i,j);
  end
end
```

7. Si vous êtes motivé, et que vous avez bien compris le fonctionnement des deux boucles imbriquées, essayez d'écrire un script qui calcule le produit matriciel de deux matrices carrées `A` et `B` de dimension $n \times n$.

Réponse :

```
for i = 1:n
  for j = 1:n
    x = 0
    for k = 1:n
      x = x + A(i,k)*B(k,j);
    end
    C(i,j) = x
  end
end
```