TP N ° 06 : Les fonctions dans Matlab

Dans ce TP, nous verrons l'utilisation des fonctions dans Matlab. Nous verrons comment écrire une fonction dans Matlab.

1. Ecrire des fonctions simples

Pour écrire une fonction dans Matlab, la première règle à respecter est de donner au fichier **.m** le même nom que la fonction que l'on est en train d'écrire. Par exemple, une fonction qui s'appellerait **mafact** devra être écrite dans le fichier **mafact.m**.

Pour écrire, une fonction dans Matlab, on doit d'abord donner les noms des valeurs en sortie générées par la fonction, puis le nom de la fonction, et enfin les noms des paramètres en entrée de la fonction :

```
function [sortie1,sortie2, ...] = nom_fonction(entree1,entree2, ...)
...
End
```

Par exemple, si on souhaite faire une fonction **produit**, qui calcule et renvoie en sortie le produit de deux scalaires passés en paramètre, on écrira dans le fichier **produit.m** :

```
function [res] = produit(a,b)
res = a*b;
end
```

Pour tester cette fonction, placez-vous (à l'aide de l'explorateur de fichiers Matlab) dans le répertoire qui contient le fichier produit.m et on écrira dans la fenêtre de commande :

```
e = produit(2,4);
```

Une fois cette commande exécutée, on récupère bien dans **e** la valeur de sortie de la fonction, c'est à dire le produit des deux paramètres en entrée.

On remarquera que l'on ne voit pas, dans la zone des variables de Matlab, une variable nommée **res**. En effet, la variable **res** est interne à la fonction **produit**, et ne vit que pendant l'appel de cette fonction. Une fois la fonction terminée, la variable est effacée. De plus, la variable **res** n'existe que pour la fonction produit : elle ne peut pas être lue par une autre fonction ou par une commande directement dans Matlab.

Considérez cette autre fonction :

```
function [res] = somme
res = a+b;
end
```

Cette fois-ci, on appellera cette fonction ainsi :

```
a=2; b=4;
e = somme();
```

Cette fonction ne marche pas : même s'il existe des variables **a** et **b** dans les variables de Matlab, ce ne sont pas des variables internes à la fonction **somme**. Or, une fonction ne peut pas lire les variables de Matlab, elle ne connait que ses variables internes et les variables passées en paramètre.

2 Ecrire des fonctions avec un nombre variable de paramètres en entrée

On peut écrire des fonctions dont le nombre de paramètres n'est pas fixe, mais peut grandir. Pour ce faire, on d_eclare la fonction ainsi :

```
function [sortie1,sortie2, ...] = nom_fonction(entree1,entree2,...,varargin)
...
End
```

Ici, on déclare une fonction avec un certain nombre de paramètres de sortie obligatoires, ainsi qu'un certain nombre de paramètres d'entrée obligatoires. Le mot clef **varargin** permet de spécifier qu'en plus des entrées obligatoires, il pourra y avoir des entrées supplémentaires à la fonction.
Considérez ce code qui permet de faire le produit de tous les entiers passés en paramètre de la fonction :

```
function [res] = produit(a,b, varargin)
res = a*b;
    for i=1:length(varargin)
res = res*varargin{i};
end
end
```

Si l'on appelle ensuite, dans l'éditeur de code Matlab, la fonction ainsi :

```
r = produit(2,4)
```

La fonction place dans r la valeur 8, comme on pouvait s'y attendre. Si l'on appelle la fonction ainsi :
r = produit(2,4, 3, 7)
La fonction place dans r la valeur 168. En réalité, la fonction a placé dans son paramètre d'entrée **a** la valeur 2, dans son paramètre d'entrée **b** la valeur 4, et dans le paramètre d'entrée **varargin** une liste contenant les nombres 3 et 7. Dans le code de la fonction, on parcourt cette liste et on multiplie la variable **res** avec les valeurs de la liste.

3 Ecrire des fonctions avec un nombre variable de paramètres en sortie

Afin d'écrire des fonctions avec un nombre variable de paramètres de sortie, on utilise le même principe que précédemment, mais avec le mot clef **varargout**. Par exemple, voici le code d'une fonction permettant de demander des entiers à l'utilisateur pour remplir les variables de sortie :

```
function [varargout] = demander()
for i=1:nargout
varargout{i} = input('Entrez une valeur : ');
end
end
```

La grande différence avec la partie précédente vient de l'utilisation du mot clef **nargout** qui permet de connaitre le nombre de paramètres de sortie choisi par l'utilisateur. En effet, on peut pas utiliser la fonction length sur la variable **varargout** car elle n'est pas encore construite et initialisée lorsque la fonction début. C'est au fur et à mesure que l'on remplit les valeurs de **varargout{k}** dans la boucle for que cette variable est construite.
Si l'on appelle cette fonction ainsi :

```
[a b c] = demander();
```

Alors la fonction demandera trois éléments qu'elle rangera dans les variables **a**, **b** et **c**.