Conseils pour améliorer la lisibilité de vos codes MATLAB

Voici quelques conseils pour rendre vos codes MATLAB plus lisibles, ceci aura pour effet d'en simplifier la lecture et la compréhension pour d'autres développeurs, mais aussi pour vous-même si vous êtes amenés à travailler de nouveau sur un de vos codes au bout de plusieurs mois.

1. Commenter votre code :

Utiliser les commentaires en utilisant le caractère « % » pour mieux comprendre le rôle de chaque ligne de code, une séquence de lignes de code, une boucle, une fonction ou tout un algorithme (programme).

Par exemple :

```
Choix=-1; % initialisation par valeur inacceptable pour pouvoir  % entrer dans la boucle.
% =   =  = cette boucle permet de lire un choix 0,1 ou 2 = = =
while (Choix<0)||(Choix>2)
   Choix=input(sprintf('\nEntrer votre choix (0, ou 1 [1]): '));
    if isempty(Choix)
        Choix=1;       % le 1 est le choix par défaut
    else
        if (Choix<0)||(Choix>2)
            sound(sin(1:3000));
        end
    end
end
```

2. Utiliser les messages pendant l'exécution :

Faciliter la compréhension de des tâches effectuées pendant l'exécution par l'utilisateur en affichant des messages expliquant l'étape ou la nature des résultats affichés.

Par exemple :

```
fprintf('Voici les valeurs du vecteur propre : ');
disp(X) ;
```

On peut améliorer davantage l'affichage des messages en leur ajoutant des sauts de lignes ou des espaces de tabulation devant le texte :

```
fprintf('\t\tVoici les valeurs du vecteur propre :\n');
disp(X) ;
```

3. Utiliser l'indentation du code :

L'indentation du code aligne verticalement le début de chaque ligne de code selon des règles propres à chaque langage.

En plus d'améliorer la lisibilité du code, l'indentation du code permet souvent de distinguer des erreurs à l'œil nu. Par contre, elle ne sert généralement à rien pour le compilateur ou l'interpréteur (sauf pour certains langages comme Python)

Pour effectuer une indentation correcte du code MATLAB, il faut sélectionner l'ensemble du code dans l'éditeur, puis utiliser le menu « Text > Smart Indent ». On peut aussi utiliser les raccourcis clavier suivants :

- `Windows : Ctrl+A (sélection) puis Ctrl+I (indentation)`
- `Mac : ⌘+A (sélection) puis ⌘+I (indentation)`
- `Linux (Emacs) : Ctrl+X,H (sélection) puis Ctrl+Alt+/ (indentation)`

Exemple de code sans indentation :

```
M = rand(2,3);
for i = 1:2
for j = 1:3
if M(i,j)<=0.5
M(i,j) = 0;
Else
M(i,j) = 1;
end
end
end
M
```

Le même code avec indentation :

```
M = rand(2,3);
for i = 1:2
    for j = 1:3
        if M(i,j)<=0.5
            M(i,j) = 0;
        else
            M(i,j) = 1;
        end
    end
end
```

4. Mettre des espaces dans les lignes de code :

a. Avec l'opérateur « = »

Ajouter un espace avant et après l'opérateur d'affection « = » améliore la lisibilité du code.

Par exemple :

```
M=rand(2,3);
```

La commande ci-dessus s'écrit plus lisiblement comme ceci :

```
M = rand(2,3);
```

Les termes de droite et de gauche sont directement identifiables.

Des espaces peuvent également être ajoutés autour du signe « = » de la première ligne des boucles for-end :

```
for i = 1:2
```

Certains préconisent également d'ajouter des espaces avec les autres opérateurs mathématiques.

Par exemple :

```
x = a/b+c;
```

L'équation ci-dessus pourrait aussi s'écrire :

```
x = a / b + c;
```

b. Avec les arguments d'une fonction :

On peut également utiliser des caractères d'espacement après les virgules au niveau des arguments d'une fonction.

Par exemple :

```
[i,j,v] = find(M>0.5,1,'first');
```

L'appel à la fonction find devient :

```
[i, j, v] = find(M>0.5, 1, 'first');
```

L'identification de chaque argument est ainsi améliorée.

5. Éviter d'utiliser des virgules pour séparer les éléments d'un tableau

En mathématiques, nous utilisons la virgule comme séparateur de décimales (les anglo-saxons utilisent le point).

Dans MATLAB, la virgule peut servir à séparer des valeurs numériques sur une même ligne d'un tableau.

Par exemple :

```
M = [1, 2, 3 ; 4, 5, 6 ; 7, 8, 9];
```

Les virgules sont ici inutiles pour MATLAB et peuvent, à tort, faire penser à une suite de nombres réels.

Leur suppression enlève donc cette possible confusion sans nuire à la lisibilité du code.

L'exemple devient :

```
M = [1 2 3 ; 4 5 6 ; 7 8 9];
```

6. Ne pas supprimer le « 0 » pour les nombres réels entre -1 et1 :

La syntaxe MATLAB permet de ne pas utiliser le caractère « 0 » dans l'écriture des nombres réels contenus dans l'intervalle]-1,1[.

Par exemple :

```
if M(i,j)<=0.5
```

Dans la condition précédente, la valeur 0.5 pourrait s'écrire .5 et la condition deviendrait :

```
if M(i,j)<=.5
```

On voit ici que le rapprochement des termes « <=. » diminue la lisibilité du code.

Ceci est encore plus vrai avec les valeurs négatives :

```
if M(i,j)<=-.5
```

7. Supprimer les parenthèses superflues :

a. Opérations mathématiques

Les parenthèses ne doivent être utilisées que pour assurer l'ordre de priorité des opérations.

Par exemple, dans le calcul suivant :

```
result = ((X(1)*35.1)+(X(5)*20.25)+(X(8)*2.5)-(X(9)*105.78));
```

Les parenthèses ne servent à rien et peuvent être avantageusement remplacées par des espaces :

```
result = X(1)*35.1 + X(5)*20.25 + X(8)*2.5 - X(9)*105.78;
```

b. Opérations logiques

Comme précédemment, selon l'ordre de priorité des opérateurs, certaines parenthèses peuvent être supprimées dans les conditions logiques.

Par exemple :

```
if (x(1)>0.5)&&((x(2)<1)||(x(3)>0.1))
```

La condition précédente peut se simplifier comme ceci :

```
if x(1)>0.5 && (x(2)<1 || x(3)>0.1)
```

8. Opter pour l'exécution étape par étape :

S'il y a beaucoup de résultats à afficher, donner à l'utilisateur la possibilité de faire l'exécution étape par étape. Par exemple afficher les résultats de la 1ère itération, puis en appuyant une touche o affiche les résultats de la 2ème et ainsi de suite.

Par exemple ajouter la ligne suivante à chaque fois qu'on préfère stopper momentanément l'exécution d'une boucle :

```
Touche = input('Une touche pour continuer ...');
```

Sources :

Le blog du site : « developpez.com : https://blog.developpez.com/ » avec quelques ajouts personnels.