Initiation MATLAB – Séance 3

–

Autres structures de données, entrées/sorties et animation

Guillaume Revy / Chu-Duc Nguyen

17/12/08

1 En MATLAB, tout n'est pas matrice...

On a vu jusqu'à maintenant que MATLAB permet de manipuler facilement des matrices. Mais, il permet également la manipulation de deux autres types de structure de données : les *cell-arrays* et les *structures*, qui permettent notamment de stocker des éléments de différents types et de différentes tailles dans un seul tableau. Ces structures de données sont très utilisées par MATLAB, comme par exemple pour spécifier les options (texte) lors de l'affichage graphique.

1.1 Les structures

Une structure est assimilable à un tableau pour lequel on peut définir différents champs pour chaque élément. La définition d'une structure consiste alors à définir, pour chaque élément de la structure (du tableau), ses différents champs. Cette définition peut se faire de deux manières.

```
>> matlab_td3(1).date = '09/05/07';
>> matlab_td3(2).date = '23/05/07 - 30/05/07';
>> matlab_td3(3).date = '06/06/07 - 13/06/07';
>> matlab_td3(1).titre = 'Seance 1';
>> matlab_td3(2).titre = 'Seance 2';
>> matlab_td3(3).titre = 'Seance 3';
>> matlab_td3(1).duree = 2;
>> matlab_td3(2).duree = 3;
>> matlab_td3(3).duree = 3;

>> matlab_td3
matlab_td3 =
1x3 struct array with fields:
    date
    titre
    duree

>> matlab_td3 = struct('date',{'09/05/07','23/05/07 - 30/05/07',
                               '06/06/07 - 13/06/07'},
                    'titre',{'Seance 1','Seance 2','Seance 3'},
                    'duree',{2,3,3});
```

On peut observer que, lorsque l'utilisateur demande l'affichage de la structure, seul les noms des différents champs sont affichés (et non leurs contenus). L'accès à un champ d'un élément de la structure et sa modification se font de la manière suivante :

```
>> matlab_td3(1).titre
ans =
Seance 1
>> matlab_td3(1).titre = 'Initiation MATLAB - Seance 1';
>> matlab_td3(1).titre
ans =
Initiation MATLAB - Seance 1
```

On peut remarquer que si le contenu du champ est un matrice, elle peut être manipulée comme n'importe quelle matrice.

1.2 Les cell-arrays

Les cell-arrays diffèrent des structures dans le sens où on accède aux différents éléments en utilisant les indices dans le tableau et non plus les noms des champs. La structure précédente peut être définie par un cell-array de la manière suivante :

```
>> matlab_td3{1,1} = '09/05/07';
>> matlab_td3{2,1} = '23/05/07 - 30/05/07';
>> matlab_td3{3,1} = '06/06/07 - 13/06/07';
>> matlab_td3{1,2} = 'Seance 1';
>> matlab_td3{2,2} = 'Seance 2';
>> matlab_td3{3,2} = 'Seance 3';
>> matlab_td3{1,3} = 2;
>> matlab_td3{2,3} = 3;
>> matlab_td3{3,3} = 3;

>> matlab_td3
matlab_td3 =
    '09/05/07'              'Seance 1'    [2]
    '23/05/07 - 30/05/07'   'Seance 2'    [3]
    '06/06/07 - 13/06/07'   'Seance 3'    [3]
```

Contrairement aux structures, lorsque l'utilisateur demande l'affichage d'un cell-array, le contenu des différents éléments est affiché (sauf quand le contenu est trop important, dans ce cas, seul le type de l'élément est affiché). L'affichage d'un élément particulier et sa modification se font comme pour une structure.

```
>> matlab_td3{1,2}
ans =
Seance 1
>> matlab_td3{1,2} = 'Initiation MATLAB - Seance 1';
>> matlab_td3{1,2}
ans =
Initiation MATLAB - Seance 1
```

Pour afficher le contenu complet d'un cell-array, on pourra utiliser la commande `celldisp`.

2 Les entrées/sorties

2.1 Entrée au clavier et clics souris

L'utilisateur peut saisir des informations au clavier grâce à la commande `x = input(...)`. La commande `[x y] = ginput(n)` retourne les vecteurs x et y des coordonnées de n clics souris sur la fenêtre courante.

```
>> x = input('Saisir une valeur de x : ');
Saisir une valeur de x : 5
>> x
x =
     5
>> x = input('Saisir un titre : ','s');
Saisir un titre : Seance 1
>> x
x =
     Seance 1
```

2.2 Sortie à l'écran

Pour afficher quelque chose à l'écran, l'utilisateur peut soit utiliser les commandes `disp`, qui affiche le contenu d'une variable (chaîne de caractères, vecteur, matrice...), ou `fprintf`, qui fonctionne comme la fonction `printf` en C.

```
>> A = [1 2 3];
>> disp(A);
     1     2     3
>> fprintf('-> %1.5e * 2 = %1.5e \n',[A ; 2*A]);
-> 1.00000e+00 * 2 = 2.00000e+00
-> 2.00000e+00 * 2 = 4.00000e+00
-> 3.00000e+00 * 2 = 6.00000e+00
```

La commande `sprintf` fonctionne de la même manière que `fprintf`, à la seule exception qu'elle retourne le résultat de l'affichage sous forme d'une chaîne de caractères.

2.3 Lecture et écriture dans un fichier

On a vu que l'on pouvait sauvegarder le contenu des variables dans un fichier (commande `save`). Mais on peut également écrire toute sorte de texte. Pour ce faire, il faut ouvrir le fichier (`fopen`), écrire le texte (`fprintf`) et refermer le fichier (`fclose`).

```
>> A = [1 2 3];
>> fid = fopen('essai.txt','w');        % fid : descripteur du fichier
>> fprintf(fid,'A(x) = %g\n',A);        % 'w' : ouverture en écriture
>> fclose(fid);
```

La lecture du contenu du fichier se fait de la manière, avec les commandes `fscanf` ou `textscan`.

```
>> fid = fopen('essai.txt','r');        % 'r' : ouverture en lecture
>> A = fscanf(fid,'A(x) = %g ')
A =
     1
     2
     3
>> fclose(fid); fid = fopen('essai.txt','r');
>> B = textscan(fid,'A(x) = %f ')       % B est un cell-array
B =
    [3x1 double]
>> B{1}
ans =
     1
     2
     3
>> fclose(fid);
```

3 Création d'une animation avec MATLAB

On a vu jusqu'à maintenant comment afficher (plus ou moins facilement) des données dans un espace 2D et 3D. MATLAB permet également de donner du mouvement à ces affichages et de créer des animations. La création d'une animation suit 3 étapes :

1. `moviein` : initialisation (allocation) de la mémoire nécessaire au stockage des images de l'animation
2. `getframe` : génération des images de l'animation
3. `movie` : exécution de l'animation un certain nombre de fois

L'étape d'initialisation n'est plus nécessaire depuis la version *MATLAB Release 11 (5.3)*. L'exemple suivant affiche la courbe de la fonction $\sin(x)$ sur l'intervalle $[0, 2\pi]$ de manière progressive en 15 images, et rejoue ensuite deux fois l'animation.

```
axis([0 2*pi -1 1]);
for i = 1:15
    b = 2*pi/15;
    x = 0:0.1:b*i;
    plot(x,sin(x));
    axis([0 2*pi -1 1]);
    m(:,i) = getframe;
end
movie(m,2);
```

4 Exercices d'application

4.1 Cell-arrays, structures et saisie au clavier

Ce premier exercice consiste à saisir les notes pour chaque élève d'une classe dans différentes matières, et à afficher un histogramme, qui pour chaque élève montre sa note dans chaque matière et sa moyenne générale.

1. Créer une fonction qui permet de saisir au clavier un ensemble de note, correspondant aux notes d'un élève dans chaque matière.
2. Écrire un script qui permet de saisir les notes (de chaque matière) pour chaque élève, et de les stocker dans une structure.
3. Construire et afficher l'histogramme (commande `hist`) défini précédemment (ajouter des légendes et commentaires).
4. Écrire l'ensemble de ces données dans un fichier : `Eleve : <nom> | Matiere 1 : <note> | ...`
5. Effacer le contenu dans l'espace de travail, puis relire le contenu du fichier dans une structure, et réafficher l'histogramme à partir de la structure obtenue. Refaire cette même question en lisant le contenu du fichier dans un cell-array.

4.2 Fenêtre graphique et saisie de données

Dans le premier exercice, on a saisit les données au clavier. On peut imaginer le refaire en saisissant les notes depuis des zones de saisie de texte que l'on aurait ajoutées sur une fenêtre.

1. Créer une fonction qui prend une liste de matières en paramètre, qui ouvre une fenêtre (`figure`), avec pour chaque matière une zone de saisie de texte (`uicontrol('Style', 'edit'...)`). Ces zones de saisie permettront de saisir les notes du premier élève.
2. Ajouter un bouton qui permet de sauver dans une structure de données adequate les notes saisies, puis de passer à l'élève suivant.
3. Ajouter finalement un bouton qui permet d'afficher l'histogramme défini dans l'exercie 1.

4.3 Animation d'un carré

Le deuxième exercice de la première séance avait pour objectif l'écriture de fonctions permettant les translations et rotations d'un carré dans l'espace 2D. L'objectif est aujourd'hui de créer une animation à l'aide de ces fonctions.

1. Construire et afficher graphiquement un carré dans l'espace 2D.
2. Créer une animation de 20 images, au cours de laquelle on fera subir au carré précédemment défini différentes translations et rotations de manière aléatoire.
3. Ajouter sur la fenêtre un bouton permettant de rejouer cette animation.

On pourra étendre cet exercice à l'affichage 3D, et créer l'animation d'un cube.

Remarque : pour créer les interfaces graphiques, vous pourrez utiliser l'exemple disponible à l'adresse suivante :
`http://perso.ens-lyon.fr/guillaume.revy/teaching/200809/interface.m`