Chapitre 5 : La Programmation sous Matlab

Université Alger1, Dept MI

2° année Maths, Semestre 3, 2016

Matière : Outils de Programmation 2

Contact : fodil.laib@hotmail.com

Introduction

- Matlab peut être utilisé comme un langage de programmation évolué.

- On peut écrire :
  - des scripts (fichiers de commandes)
  - ou définir de nouvelles fonctions

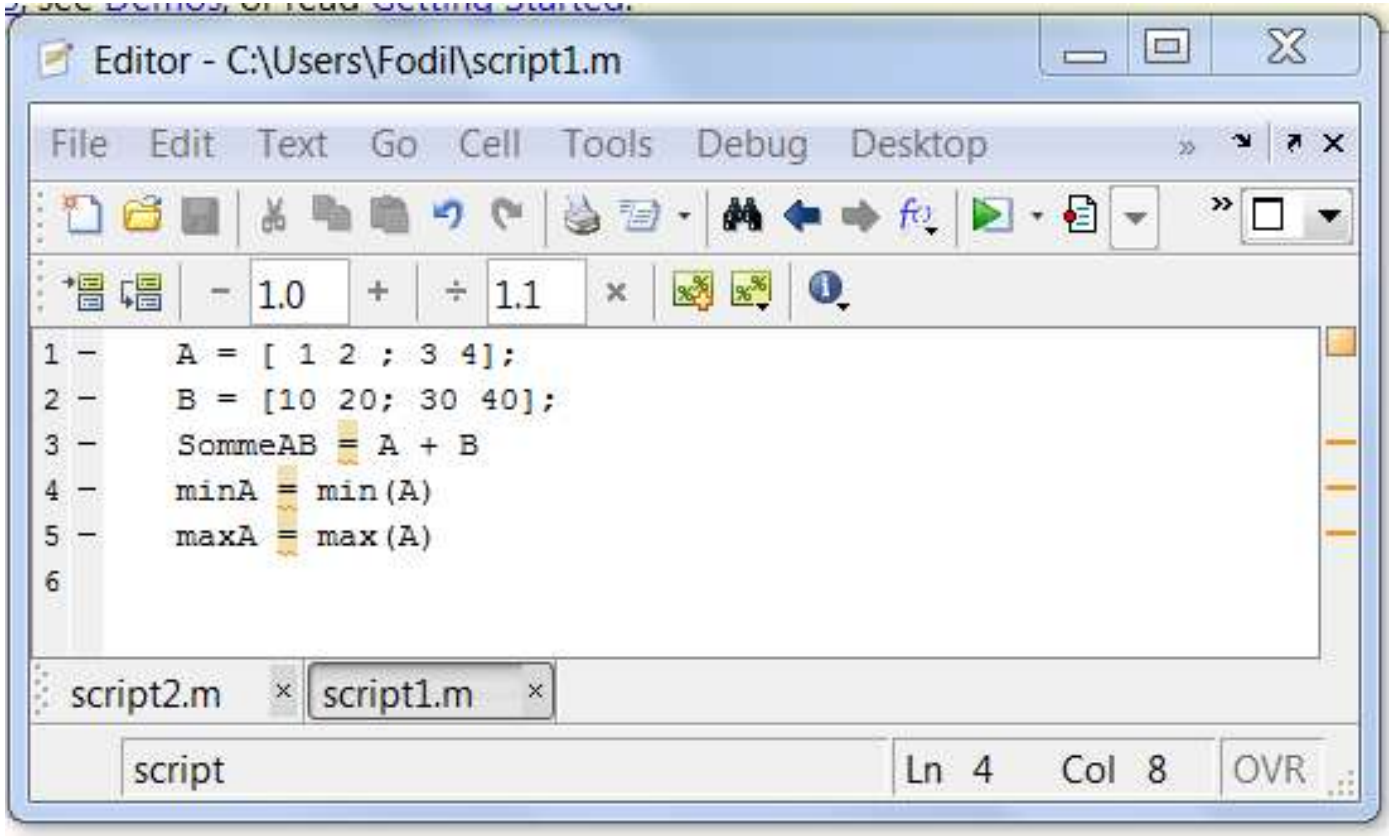
- Ces scripts et fonctions peuvent utiliser les fonctions built-in de Matlab.

Les Scripts

- Un script, ou un programme, est une **suite d'instructions Matlab**.
- Ces instructions sont écrites dans **un fichier texte, enregistré avec l'extension .m**
- Pour exécuter un script, il faut évoquer son nom de fichier dans la ligne de commande de Matlab.
- Les instructions du scripts s'exécutent l'une après l'autre comme si elles étaient saisies sur la ligne de commande de Matlab.
- **Les variables définies dans le scripts restent dans la mémoire de Matlab** après l'exécution du script.
- Le texte venant après **le signe % est un commentaire**, donc il n'est pas traité par Matlab.

Exemple 1 de script

- On appelle l' éditeur de Matlab avec la commande
- >> edit
- La fenêtre de l'éditeur s'affiche.
- On saisit les instructions suivantes, et on sauvegarde le fichier sous le nom **script1.m** :



Exécution de script1.m

Pour exécuter le script1, on revient à la fenêtre de Matlab, et on saisit :

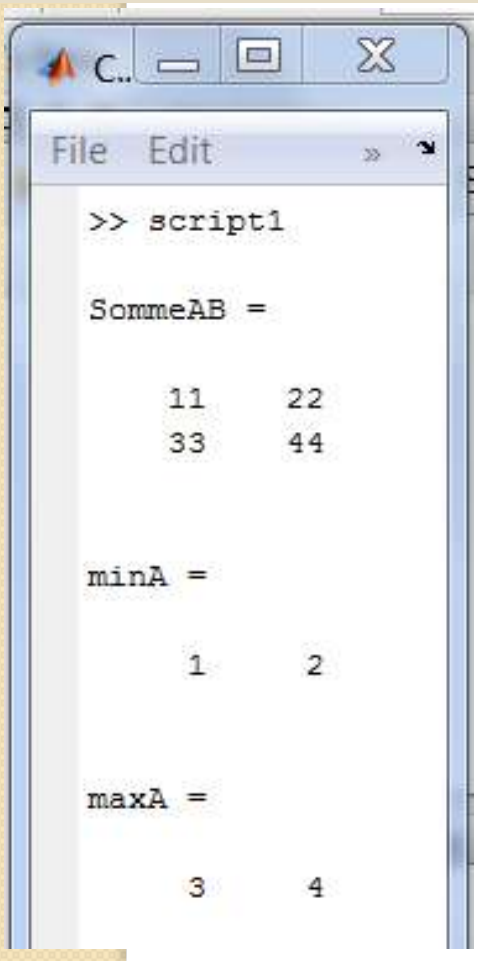
>> script1

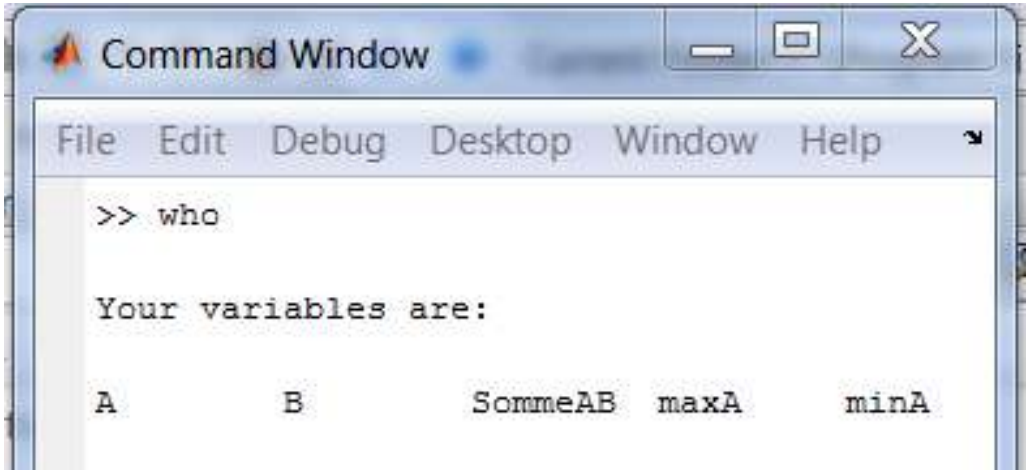
On obtient ce qui suit :

Apres l'exécution de script1, les variables A et B sont toujours disponibles dans la mémoire de Matlab.

Pour s'en convaincre, tapons

>> who





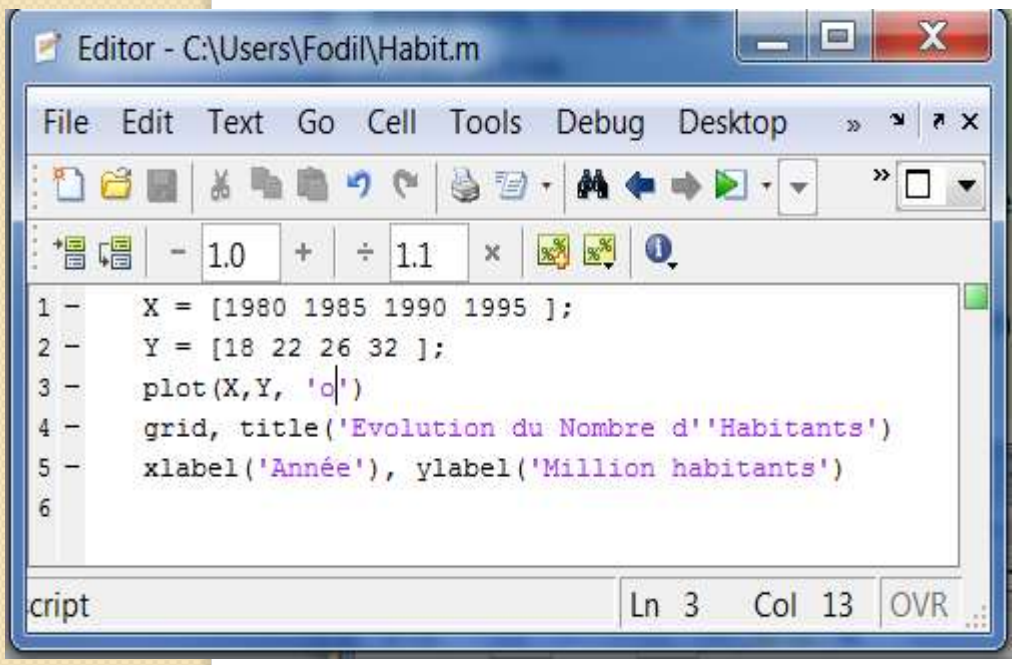
Exemple 2 de script

Evoquons l'editeur de Matlab avec l'instruction :

>> edit Habit

Ceci ouvrira l'editeur et créera le fichier Habit.m

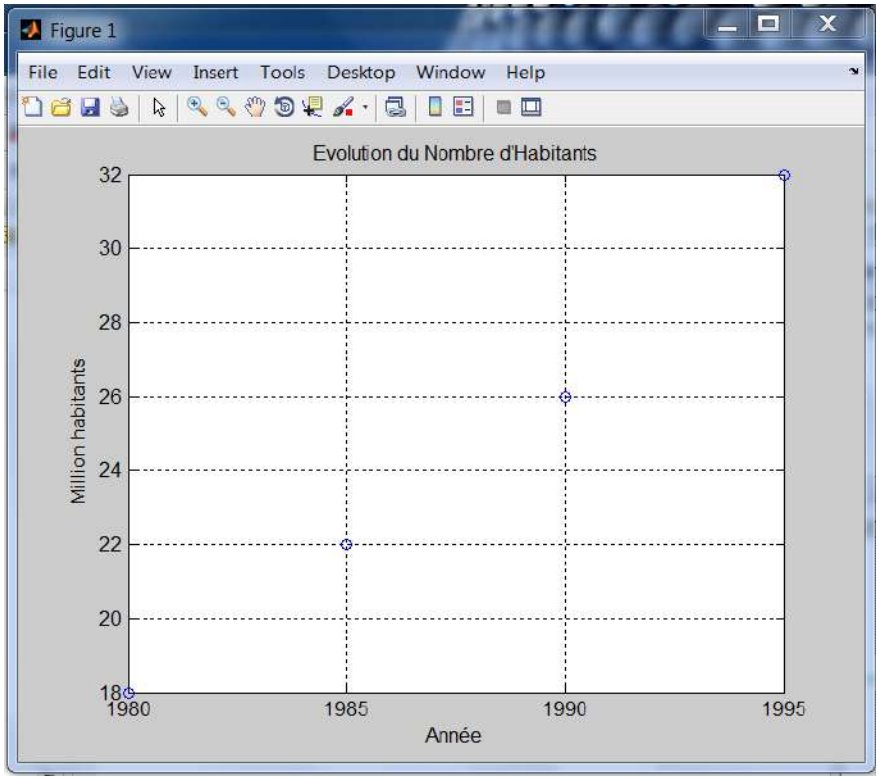
Saisissons les instructions suivantes dans le fichier Habit.m :



Une fois le fichier enregistré, revenons à la ligne de commande de Matlab, et tapons

>> Habit

Le graphe suivant s'affiche :



Conditions et Boucles

Dans un script, on peut utiliser les instructions conditionnelles :

- If condition

```
        instructions
else
        instructions
end
```

Ou bien

- switch expression

```
case value1
        instructions
Case value2
        instructions
…
otherwise
        instructions
end
```

- La boucle for

```
for variable = expr
        instructions
end
```

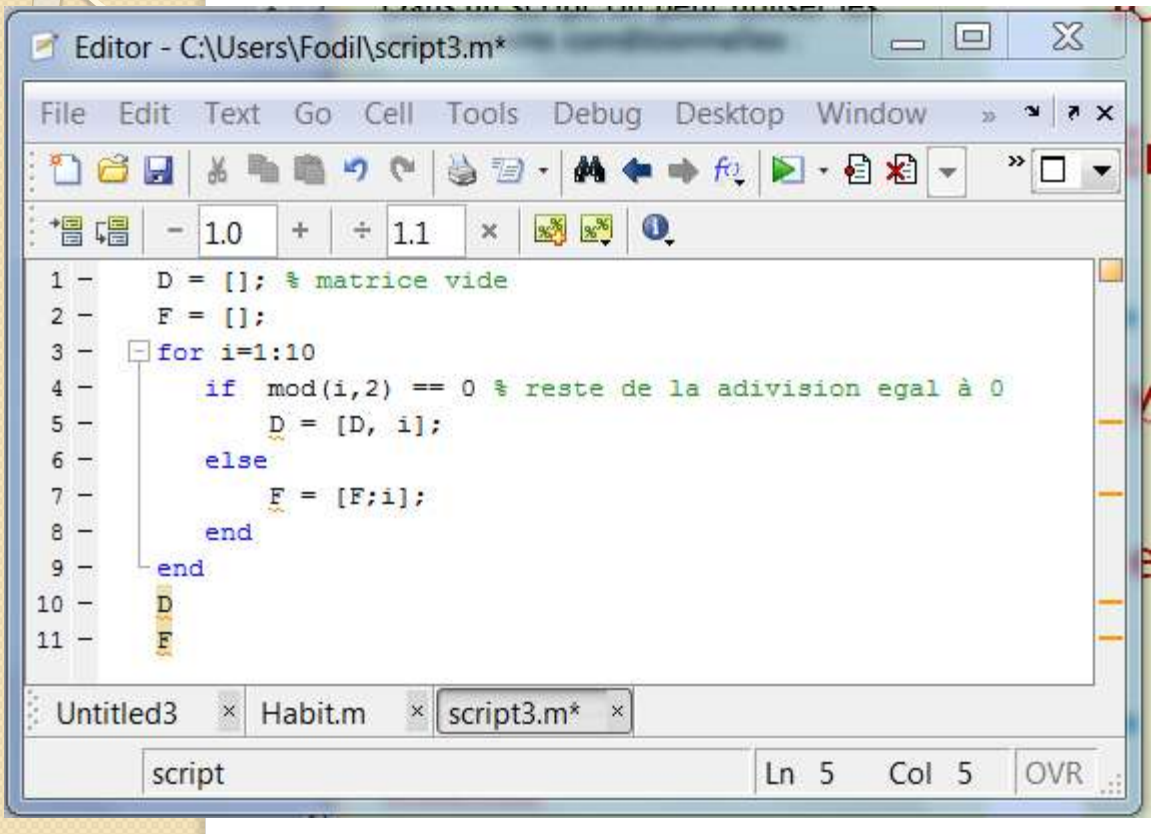
- La boucle while

```
while condition
        instructions
end
```

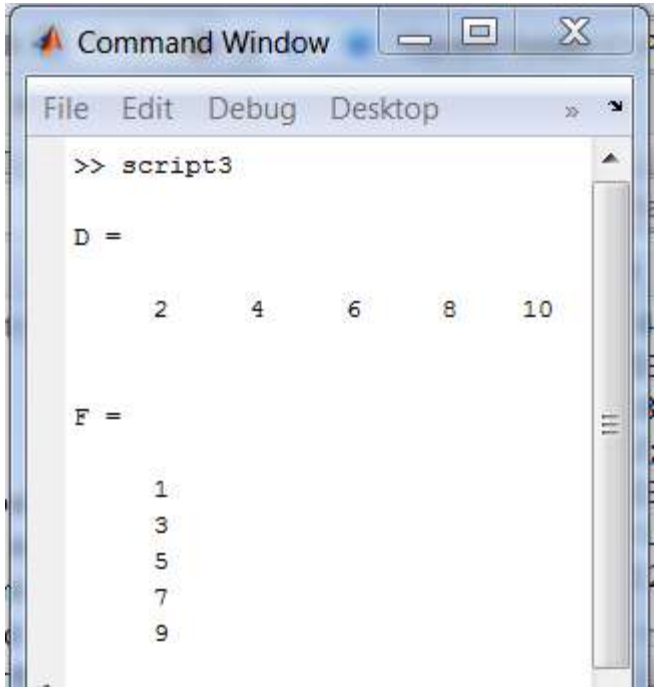
- On peut sortir d'une boucle à l'aide de l'instruction break.

Exemple de Script avec if et for

Créons le script3 ci-dessous :



L'exécution de script3 donne ce qui suit :



Les Fonctions

Une fonction est un script qui

- reçoit des arguments en entrée
- renvoie des résultats
- les variables définies localement seront effacées après l'exécution de la fonction

- La première ligne du fichier de la fonction commence par
  - function [A,B,…] = MaFonc(X,Y,…)
  - où
    - X,Y, … : sont les arguments de la fonction
    - A, B, … : sont les résultats

- La fonction doit être enregistrée dans un fichier portant le nom de la fonction et l'extension « .m » (MaFonc.m)

- Les commentaires rédigés juste après la première ligne seront affichés lorsqu'on évoque

  >> help MaFonc

Exemple 1 de Fonction

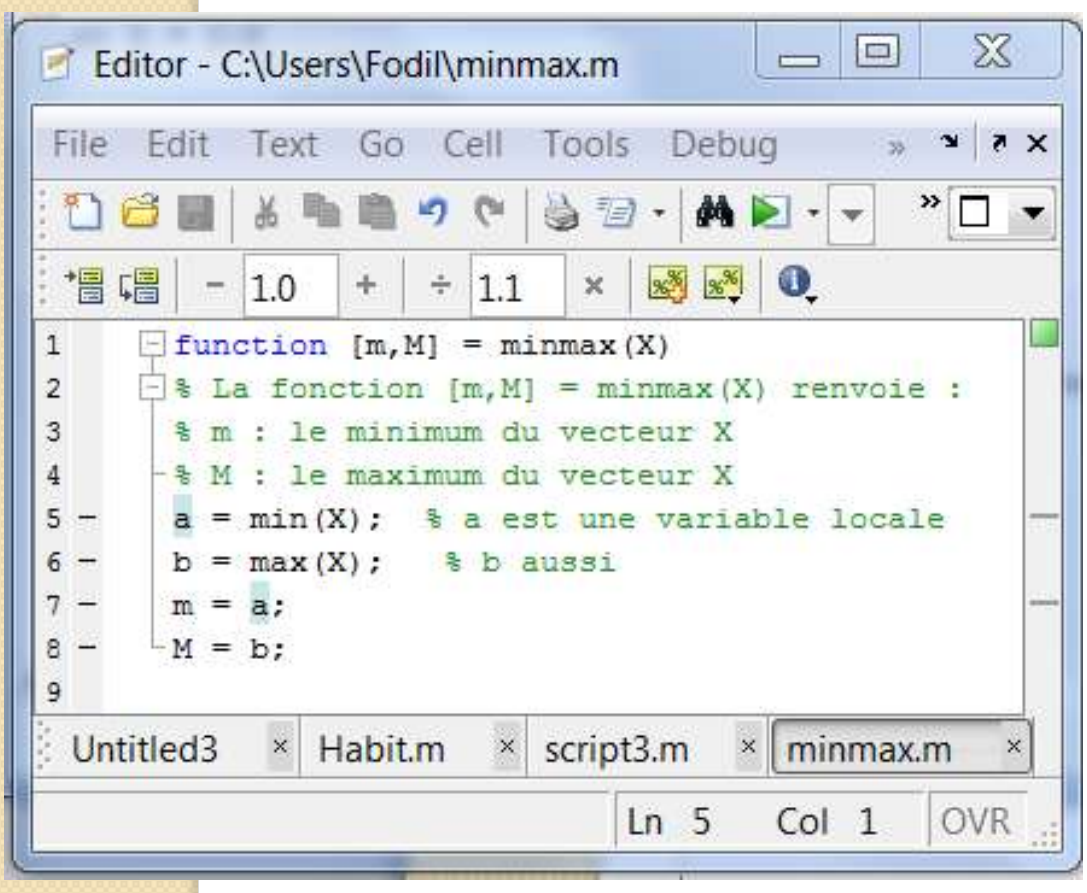
Dans l'editeur de Matlab, définissons la fonction minmax suivante

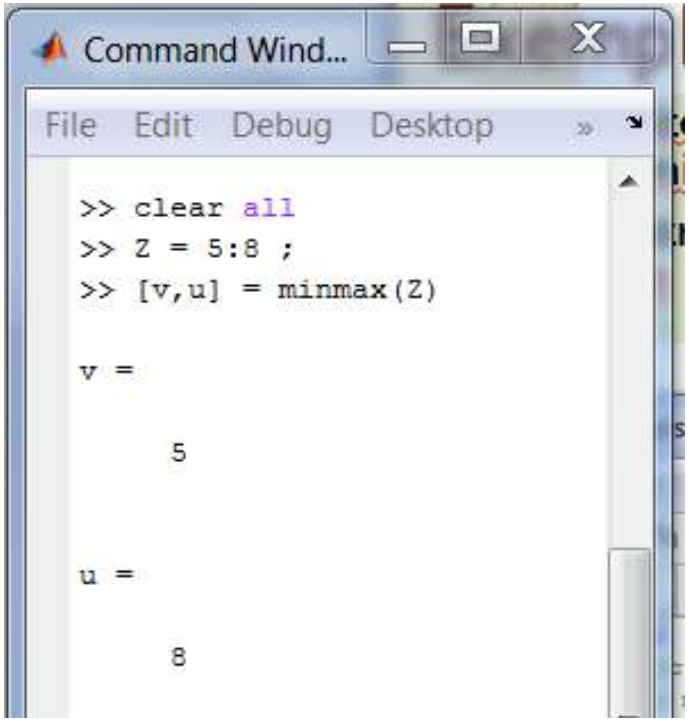
et enregistrons la dans le fichier minmax.m

On revient à Matlab.

On efface toutes les variables.

On crée le vecteur Z, puis on appelle la fonction minmax :





...

Après l'exécution de minmax, on constate grace à la commande

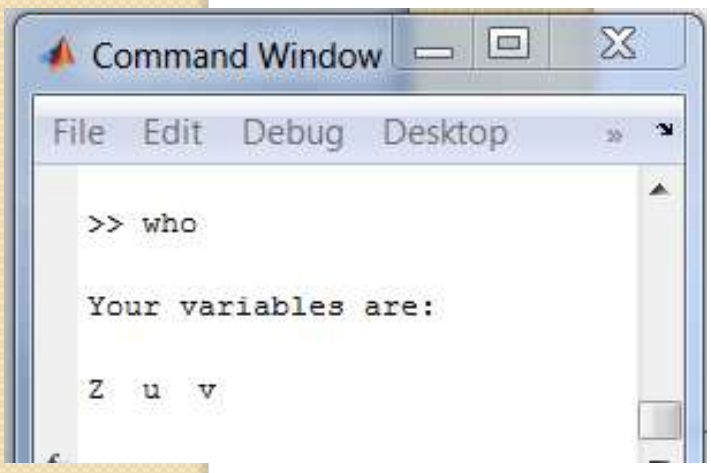
>> who

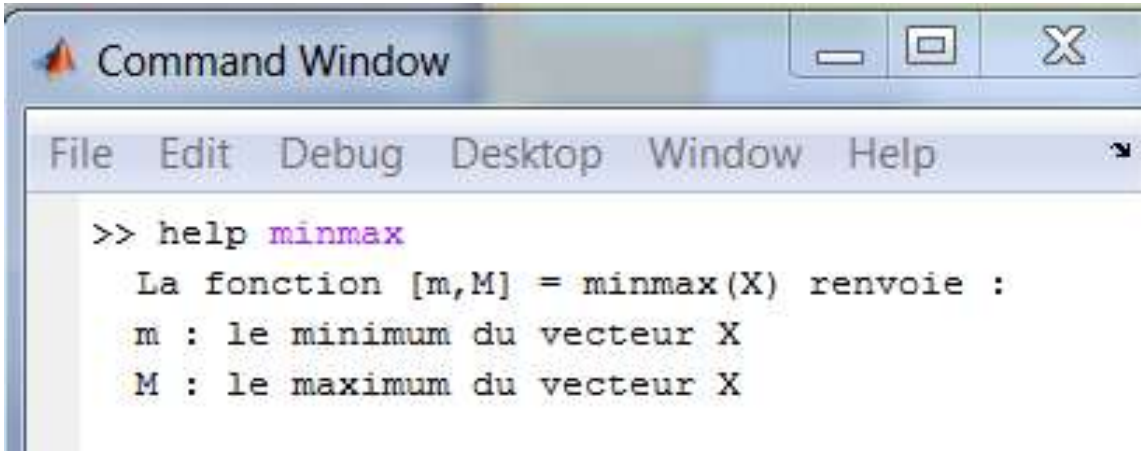
que les seules variables dans la mémoire sont Z, u et v.

Les variables locales a et b ont été effacées :

Exécutons la commande

>> help minmax





Exemple 2 de Fonction

- Créons la fonction som2 suivante.
- Puis, appelons som2 avec A comme argument :

Editor - C:\Users\Fodil\som2.m

File Edit Text Go Cell Tools Debug Desktop

```
1   function [l,s]=som2(R,s0)
2   % La fonction  [l,s]=som2(R,s0) renvoie
3   % le numero de la ligne, l, de la matrice R
4   % tel que la somme, s, de cette ligne est > s0
5   % si cette ligne n'existe pas, som2 renvoie 0.
6   l = 0;
7   [m,n] = size(R);
8   i=1;
9   while i <= m
10      s = sum(R(i,:));
11      if s > s0
12          l = i;
13          break
14      end
15      i = i + 1;
16  end
17
```

d3 | Habit.m | script3.m | minmax.m | som2.m

om2 Ln 7 Col 1 OVR

Command Window

File Edit Debug Desktop Window Help

```
>> help som2
  La fonction  [l,s]=som2(R,s0) renvoie
  le numero de la ligne, l, de la matrice R
  tel que la somme, s, de cette ligne est > s0
  si cette ligne n'existe pas, som2 renvoie 0.

>> A

A =

     1     2     3
     5    10     6
     2     6     5

>> [ligne,somme] = som2(A,20)

ligne =

     2


somme =

    21
```

Un Script Peut Appeler un Autre Script

Parfois, il est utile de sauvegarder les données dans un fichier script A, et appeler ce script à partir d'un script B.

Par exemple, créons un fichier de données Data1.m

```
% Fichier de données,
% crée le 10/12/2016
A = [ 3     0      3
      1     -5      0
      0     2      9 ];

b = [  13    29    49 ]';
```

Puis, créons un fichier de traitement (**Trait1.m**) qui appelle le fichier de données :

```
% Fichier de traitement

Data1 % appel du fichier Data1.m

% Resolvons le systeme Ax=b
if det(A) ~= 0
    disp('La solution est :')
    x = inv(A)*b
else
    disp('Pas de solution')
end
```

Puis, à partir de la fenêtre de commande Matlab, tapons :

```
>> Trait1
La solution est :

x =

   -2.5116
   -6.3023
    6.8450
```

Les Fonctions Locales

Dans le fichier d'une fonction A, on peut ajouter d'autres **fonctions locales** B, C, …

Exemple, définissons une fonction **minabs** dans le fichier **minabs.m**; dans le même fichier, ajoutons une fonction loclae **minvec** :

```
function y = minabs(X)
% La fonction y = minabs(X) renvoie
% le minimum en valeur absolue, y,
% de la matrice X

[n,m] = size(X);
y = inf;
for i = 1:n
    z =  minvec( X(i,:) ) ;
    if y > z
        y = z;
    end
end

function a = minvec(V)
% minvec renvoie le minimum
% en valeur absolue du vecteur V
V1 = abs(V);
a = min(V1);
```

A partir de Matlab, appelons la fonction minabs, puis minvec.

Uniquement la première est reconnue :

```
>> R = [10 -5 2 ; -18  -50  1 ]

R =

    10    -5     2
   -18   -50     1

>> minabs(R)

ans =

     1

>> minvec([5 -3  4])
Undefined function 'minvec'
```