Chapitre 2: Types de données et variables

2.1 Particularités de MATLAB

Comme tout langage de programmation MATLAB permet de définir des données variables. Une variable est désignée par un identificateur qui est formé d'une combinaison de lettres et de chiffres. Le premier caractère de l'identificateur doit nécessairement être une lettre. Attention, MATLAB différencie majuscules et minuscules! Ainsi `X33` et `x33` désignent deux variables distinctes. Les variables sont définies au fur et à mesure que l'on donne leurs noms (identificateur) et leurs valeurs numériques ou leurs expressions mathématiques. L'utilisation de variables avec MATLAB ne nécessite pas de déclaration de type ou de dimension. Le type et la dimension d'une variable sont déterminés de manière automatique à partir de l'expression mathématique ou de la valeur affectée à la variable.

2.2 Les 4 types de données

Les 3 principaux types de variables utilisés par Matlab sont : les réels, les complexes et les chaînes de caractères. Le type logique est associé au résultat de certaines fonctions

Pour MATLAB toute variable est considérée comme étant un tableau d'éléments d'un type donné. MATLAB différencie trois formes particulières de tableaux. Les *scalaires* qui sont des tableaux à une ligne et une colonne. Les *vecteurs* qui sont des tableaux à une ligne ou à une colonne. Les *matrices* qui sont des tableaux ayant plusieurs lignes et colonnes. Une variable MATLAB est donc toujours un tableau que l'on appelle variable scalaire, vecteur ou matrice suivant la forme du tableau.

2.2.1 Le type complexe

L'unité imaginaire est désignée par *i* ou *j* (se transforme en *i*). les nombres complexes peuvent être écrits sous forme cartésienne a+ib ou sous forme polaire re^{it} . Les différentes écritures possibles sont :

a+i*b (ou a+b*i) et r*exp(i*t) (ou r*exp(t*i))

Avec a,b,r et t des variables de type réel.

Mais on peut écrire :

```
>> z = 2+i*5                >> z = 2+5*i
>> z = 2+5i       (mais z = 2+i5 error!   i5 = ???)
>> z =7*exp(i*3)            >> z =7*exp(3*i)            >> z =7*exp(3i)
```

Voici quelque commande concernant les nombres complexes :

si Z est de type complexe

imag(Z) retourne la partie imaginaire de Z

real(Z) retourne la partie réelle de Z

abs(Z) retourne le module de Z

angle (Z) retourne la partie imaginaire de Z

conj(Z) retourne le conjugué de Z (Z^*)

Ces dernières commandes permettent de passer aisément de la forme polaire à la forme cartésienne.

Il est possible que des variables de noms i ou j aient été redéfinies au cours d'un calcul antérieur alors on peut soit détruire ces deux variables (clear i,j)

i et j redeviennent alors l'unité imaginaire, soit réaffecter à i ou à j la valeur unité imaginaire, soit l'instruction : i = sqrt(-1).

2.2.2 Le type chaîne de caractères

Une chaîne de caractères est un tableau de caractères. Une donnée de type chaîne de caractère (char) est représentée sous la forme d'une suite de caractères encadrée d'apostrophes simples (').

Exemples:

```
>> ch1='bon'
ch1 =
       bon

>> ch2='jour'
ch2 =
       jour

>> whos
  Name      Size         Bytes    Class
  ch1       1x3              6    char array
  ch2       1x4              8    char array

Grand total is 7 elements using 14 bytes
```

```
>> ch=[ch1,ch2]
 ch =
      bonjour
>> ch(1)
 ans =
      b
>> ch(7)
 ans =
      r
>> ch(1:3)
 ans =
      bon
>> ch3='soi' ;
>> ch=[ch(1:3),ch3,ch(7)]
 ch =
      bonsoir
```

- Si une chaîne de caractères doit contenir le caractère apostrophe (') celui-ci doit être double dans la chaîne.

Exemple

```
rep='aujourd'hui'
??? rep='aujourd'hui'
              |
Error: Missing MATLAB operator.

>> rep='aujourd''hui'
rep =
      aujourd'hui
>> apos=''''
apos =
       '
```

- La chaîne de caractères vide s'obtient par 2 apostrophes''.

2.2.3 Le type logique

Le type logique possède 2 formes : 0 pour faux et 1 pour vrai

Un résultat de type logique est retourné par certaines fonctions ou dans le cas de certains tests.

Exemple :

```
>> a=1;   b=2;
>> test_E=(a= =b)
test_E =
          0
>> test_S=(a>b)
 test_S =
           0
>> test_I=(a<b)
 test_I =
           1
>> V=true
V =
    1
>> F=false
F =
    0
```

```
>> whos
  Name        Size        Bytes       Class
  F           1x1             1       logical array
  V           1x1             1       logical array
  a           1x1             8       double array
  b           1x1             8       double array
  test_E      1x1             1       logical array
  test_I      1x1             1       logical array
  test_S      1x1             1       logical array

Grand total is 7 elements using 21 bytes
```

2.2.4 Le type vecteur

On définit un vecteur ligne en donnant la liste de ses éléments entre crochets ([ ]). Les éléments sont séparés au choix par des espaces ou par des virgules.

On définit un vecteur colonne en donnant la liste de ses éléments séparés au choix par des points virgules (;) ou par des retours chariots.

On peut transformer un vecteur ligne X en un vecteur colonne et réciproquement en tapant X' (' est le symbole de transposition)

On peut obtenir la longueur d'un vecteur donné grâce à la commande length(X).

Exemple :

```
>> x1=[1 2 3], x2=[4,5,6,7], x3=[8;9;10]
x1 =
    1    2    3
x2 =
    4    5    6    7
x3 =
    8
    9
   10

>> length(x2)
ans =
    4
>> x3'
ans =
    8    9   10
```

```
>> whos
Name      Size          Bytes      Class
  x1      1x3             24     double array
  x2      1x4             32     double array
  x3      3x1             24     double array

Grand total is 10 elements using 80 bytes
```

2.2.5 Le type matrice

On définit une matrice en donnant la liste de ses éléments entre crochets. Les éléments d'une même ligne sont séparés au choix par des espaces ou par des virgules. Les lignes entre elles sont séparées par des retours chariot ou par des points virgules.

On peut obtenir les dimensions d'une matrice par la commande size. Soit A une matrice quelconque :

- size(A,1) donne le nombre de lignes.
- Size (A,2) donne le nombre de colonne.
- Size(A) donne le nombre de lignes et de colonnes → [m,n]=Size(A)

Exemple:

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

```
>> size(A)
ans =
    4    3
>> size(A,1)
ans =
    4
>> size(A,2)
ans =
    3
```

```
>> B=[1,2
    3,4 ]
 B= 1  2
    3  4

>> C=[2  0 9
       4 1 3]
 C= 2  1   9
    4  1  3
```

Remarques :

- Comme on ne définit pas de manière explicite le type d'une variable, il est parfois utile de pouvoir le déterminer. Cela est possible grâce aux commandes **`ischar`**, **`islogical`** et **`isreal`**.

`ischar(x)` retourne 1 si x est de type chaîne de caractères et 0 sinon. `islogical(x)` retourne 1 si x est de type logique et 0 sinon. La commande `isreal(x)` est à utiliser avec discernement: elle retourne 1 si x est réel ou de type chaîne de caractères (`(-:)` et 0 sinon (x est complexe à partie imaginaire non nulle ou n'est pas un tableau de valeurs réelles ou de caractères).

Exemple :

```
>> clear
>> x = 2; z = 2+i; rep = 'oui';

>> ischar(rep)
ans =
     1
>> ischar(x)
ans =
     0
>> isreal(z)
ans =
     0
>> isreal(x)
ans =
     1
>> isreal(rep)
ans =
     1
>>
```

➢ *Variables spéciales*

Ces noms de variables sont utilises par Matlab :

pi ans inf i or j realmin realmax eps ans etc.

```
>> pi
ans =
      3.1416

>> ans
ans =
      3.1416

>> eps
ans =
      2.2204e-016

>> realmax
ans =
      1.7977e+308
```

```
>> realmin
 ans =
      2.2251e-308

>> inf
ans =
      Inf

>> nan
ans =
      NaN

>>
```

2.3 Lecture des données

- pour lire une variable simple on utilise la fonction ***input*** comme suite :

```
>> x=input('introduire la valeur de x')
introduire la valeur de x4
x =
    4
>> x=input('introduire la valeur de x   ')
introduire la valeur de x   4
x =
    4
```

La différence entre les deux exemples est l'espace créé dans le commentaire après x ;

- pour lire un vecteur ou une matrice on utilise la même fonction :

```
>> y=input('introduire les valeurs du vecteur Y  ')
introduire les valeurs du vecteur Y  [1 2 3]
y =
    1    2    3

>> Z=input('introduire les valeurs du vecteur Z:  ');
introduire les valeurs du vecteur Z:  [2 5 8  7]
>>

>> T=input('introduire les valeurs de la matrice T  ')
introduire les valeurs de la matrice T  [1 2 3; 4 5 6]
T =
    1    2    3
    4    5    6
```

- S la chaîne de caractère contient une apostrophe il est indispensable de doubler l'apostrophe.

2.4 *affichage des données*

- pour afficher les données il suffi d'écrire le nom de la variable
- la commande ***disp*** permet d'afficher un tableau de valeurs numérique ou de caractères

Exemple

```
>> x=6;
>> disp(x)
    6
>> disp('bonjour')
bonjour
>> a='bon'
a =
     bon
>> disp(['le mot est  :  ',a])
le mot est  :  bon
>> disp(['la valeur de x est:  ',num2str(x)])
la valeur de x est:  6
```

num2str → Convertie un nombre en caractère.

Les formats d'affichage des réels

MATLAB dispose de plusieurs formats d'affichage des réels. Par défaut le format est le format court à 5 chiffres. Les autres principaux formats sont:

`format long` : format long à 15 chiffres.

`format short e` : format court à 5 chiffres avec notation en virgule flottante.

`format long e` : format long à 15 chiffres avec notation en virgule flottante.

MATLAB dispose également des formats **format short g** et **format long g** qui utilise la << meilleure >> des deux écritures à virgule fixe ou à virgule flottante. On obtiendra tous les formats d'affichage possibles en tapant help format. On impose un format d'affichage en tapant l'instruction de format correspondante dans la fenêtre de contrôle, par exemple format long. Pour revenir au format par défaut on utilise la commande format ou format short.

```
>> pi
ans =
   3.1416

>> format long
>> pi
ans =
  3.14159265358979

>>  format short e
>> pi^3
ans =
  3.1006e+01

>> format short g
>> pi^3
ans =
     31.006
```

Exemple :

```
x = [4/3 1.2345e-6]
format short
  1.3333    0.0000
format short e
  1.3333e+000  1.2345e-006
format short g
  1.3333  1.2345e-006
format long
  1.33333333333333   0.00000123450000
format long e
  1.333333333333333e+000    1.234500000000000e-006
format long g
  1.33333333333333             1.2345e-006
format bank
  1.33         0.00
format rat
  4/3         1/810045
format hex
   3ff5555555555555   3eb4b6231abfd271
```

– La commande `format` permet de choisir entre plusieurs modes d'affichage (sans interférer avec le `type` des valeurs numériques affichées qui est toujours le type `double`) :

Commande	Affichage	Exemple
`format short`	décimal à 5 chiffres	31.416
`format short e`	scientifique à 5 chiffres	31.416e+01
`format long`	décimal à 16 chiffres	31.4159265358979
`format long e`	scientifique à 16 chiffres	3141592653589 79e+01
`format hex`	hexadécimal	
`format bank`	virgule fixe à deux décimales	31.41
`format rat`	fractionnaire	3550/113
`format +`	utilise les symboles +, - et espace pour afficher les nombres positifs négatifs et nuls	+

2.5 Sauvegarde des données

Il est possible de sauvegarder une session MATLAB dans un fichier pour une utilisation ultérieure. L'instruction ***save nom-fic*** enregistre toutes les variables de l'espace de travail dans le fichier nom-fic.mat. Si aucun nom de fichier n'est précisé, le fichier par défaut est matlab.mat. Il est possible de ne sauver qu'une partie des variables (par exemple seulement la variable contenant le résultat d'un calcul) en utilisant l'instruction ***save nom-fic nom-var*** où nom-var est le nom de la (ou des) variable(s) à sauvegarder. Attention, seul le contenu des variables est sauvegardé et non pas l'ensemble des instructions effectuées durant la session. Pour ramener dans l'espace de travail les variables sauvegardées dans le fichier nom-fic.mat, taper ***load nom-fic.***

Exemple :

```
>> x=2*pi/3, y=sin(x), z=cos(x)
x =
   2.0944
y =
   0.8660
z =
  -0.5000

>> save data
>> save toto y z
>> who

Your variables are:

x  y  z

>> clear all
>> who
>>          % vide
```

```
>> load toto
>> who
   Your variables are:
   y  z

>> y
 y =
    0.8660

>> z
  z =
     -0.5000
>> x
??? Undefined function or variable 'x'.

>> load data
>> who
  Your variables are:
   x  y  z
>>
```

(Voir aussi la commande **diary**)