Informatique 3

Univ dbkm
Enseignent : AZIZOU Fethi

18/03/2017
1.0

Table des matières

Objectifs

Apprendre à l'étudiant la programmation en utilisant des logiciels faciles d'accès (essentiellement : *Matlab*, *Scilab*, *Mapple* ...). Cette matière sera un outil pour la réalisation des TP de méthodes numériques en S4

Chapitre 2 : Types de données et variables

1. Objectifs spécifiques

A la fin de ce chapitre l'étudiant doit être capable de:

- connaître les types de données MATLAB;
- Définir un vecteur et matrice sur MATLAB.

2. 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. 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. Une variable peut être de type réel, complexe, chaîne de caractères ou logique.

⚠ *Attention*

Matlab différencie majuscules et minuscules ! Ex: X33 et x33 désignent deux variables distinctes

3. Les types de données

3.1. Les types de données Matlab

Les trois principaux types de variables utilisés par Matlab sont les types réel, complexe et chaîne de

caractères. Il n'y a pas de type entier à proprement parler. Le type logique est associé au résultat de certaines fonctions. Signalons qu'il est inutile (impossible) de déclarer le type d'une variable. Ce type est établi automatiquement à partir des valeurs affectées à la variable.

Par exemple les instructions x = 2 ; z = 2+i ; rep = 'oui' ; définissent une variable x de type réel, une variable z de type complexe et une variable rep de type chaîne de caractères.

3.2. Le type complexe

L'unité imaginaire est désignée par *i* ou *j*. Les nombres complexes peuvent être écrits sous forme cartésienne *a + i*b* ou sous forme polaire r*exp(i*t). Les différentes écritures possibles sont a+ib, a+i*b, a+b*i, a+bi et r*exp(it) avec a, b, r et t des variables de type réel. Les commandes *imag*, *real*, *abs*, angle permettent de passer aisément de la forme polaire à la forme cartésienne et réciproquement. Si z est de type complexe, les instructions *imag(z)* et *real(z)* retournent la partie imaginaire et la partie réelle de z. Les instructions *abs(z)* et *angle(z)* retournent le module et l'argument de z.

```
>> z = [1+i, 2, 3i]
z =
   1.0000 + 1.0000i 2.0000 0 + 3.0000i
>> y = [1+i, 2, 3 i]
y =
   1.0000 + 1.0000i 2.0000 3.0000 0 + 1.0000i
>>
```

3.3. Le type chaîne de caractères

Une chaîne de caractères est un tableau de caractères. Une donnée de type chaîne de caractères (*char*) est représentée sous la forme d'une suite de caractères encadrée d'apostrophes simples ('). Une variable de type chaîne de caractères étant interprétée comme un tableau de caractères, il est possible de manipuler chaque lettre de la chaîne en faisant référence à sa position dans la chaîne.

L'exemple suivant présente différentes manipulations d'une chaîne de caractères.

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

3.4. Le type logique

Le type logique (logical) 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.

Dans l'exemple qui suit on considère une variable x contenant la valeur 123 et une variable y définie par l'expression mathématique y = exp(log(x)). On teste si les variables x et y contiennent les mêmes valeurs. La variable tst est une variable de type logique qui vaut 1 (vrai) les valeurs sont égales et 0 (faux) sinon. Suivant la valeur de tst, on affiche la phrase x est égal a y ou la phrase x est different de y. Dans l'exemple proposé, compte-tenu des erreurs d'arrondis lors du calcul de exp(log(123)), la valeur de la variable y ne vaut pas exactement 123. On pourra également considérer le cas où x = 12.

```
>> x = 123; y = exp(log(x));
>> tst = ( x==y );
>> if tst, disp('x est egal a y '), else disp('x est different de y '), end
x est different de y
>> format long
>> x, y, tst
x =
    123
y =
    1.229999999999999e+02
tst =
    0
>> format, clear
>>
```

4. Les vecteurs

- 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 (touche Entrée/Enter).
- On peut obtenir la longueur d'un vecteur donné grâce à la commande `length`.
- On peut transformer un vecteur ligne x en un vecteur colonne et réciproquement en tapant x' (' est le symbole de transposition).

```
>> 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), length(x3)
ans =
     4
ans =
     3
```

```
>> x3'
ans =
     8 9 10
```

- Un vecteur peut également être défini « par blocs » selon la même syntaxe.

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

- Les éléments d'un vecteur peuvent être manipulés grâce à leur indice dans le tableau. Le éme élément du vecteur *x* est désignée par `x(k)`. Il est possible de manipuler plusieurs éléments d'un vecteur simultanément. Ainsi les éléments *k* à *l* du vecteur *x* sont désignés par `x(k :l)`. On peut également manipuler facilement les éléments d'un vecteur dont les indices sont en progression arithmétique. Ainsi si l'on souhaite extraire les éléments k, k + p, k + 2p, · · · , k + Np = l on écrira `x(k :p :l)`.

```
>> X(5)
ans =
     5
>> X(4:10)
ans =
     4 5 6 7 8 9 10
>> X(2:2:10)
ans =
     2 4 6 8 10
>> K = [1 3 4 6]; X(K)
ans =
     1 3 4 6
>>
```

- Pour définir un vecteur x dont les composantes forment une suite arithmétique de raison *h*, de premier terme *a* et de dernier terme *b*, on écrira `x = a :h :b`.
- La commande `linspace` permet de définir un vecteur x de longueur *N* dont les composantes forment une suite arithmétique de premier terme *a* et de dernier terme *b* .La syntaxe est `x = linspace(a,b,N)`.

```
>>  x = 1.1:0.1:1.9

x =

    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000
  1.8000    1.9000

>>  x = 1.1:0.2:2

x =

    1.1000    1.3000    1.5000    1.7000    1.9000
>> x = linspace(1.1,1.9,9)

x =

    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000
  1.8000    1.9000
```

```
>>
```

4.1. Vecteurs spéciaux

Les commandes `ones`, `zeros` et `rand` permettent de définir des vecteurs dont les éléments ont respectivement pour valeurs 0, 1 et des nombres générés de manière aléatoire.

ones(1,n)	vecteur ligne de longueur n dont tous les éléments valent 1
ones(m,1)	vecteur ligne de longueur n dont tous les éléments valent 1
zeros(1,n)	vecteur ligne de longueur n dont tous les éléments valent 1
zeros(m,1)	vecteur colonne de longueur m dont tous les éléments valent 0
rand(1,n)	vecteur ligne de longueur n dont les éléments sont générés de manière aléatoire entre 0 et 1
rand(m,1)	vecteur colonne de longueur m dont les éléments sont générés de manière aléatoire entre 0 et 1

5. Les Matrices

Définition :

De façon similaire aux vecteurs (un vecteur n'est en fait qu'une matrice à 1 colonne ou 1 ligne) les matrices se définissent comme suit :

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

A =

     1     2     3
     4     5     6
     7     8     9

>>
```

et même à partir de vecteurs :

```
>> V1=[1,2,3 4 5 6];
>> V2=[7 8 9 1 2 3];
>> V=[V1;V2]

V =

     1     2     3     4     5     6
     7     8     9     1     2     3

>>
```

5.1. Quelques matrices particulières :

eye(n)	Matrice identité de taille n.
zeros(n) ou **zeros(n,m)**	Matrices de taille n ou n×m remplie de zeros.
ones(n), ones(n,m)	Matrices de taille n ou n×m remplie de uns.
rand(n), rand(n,m)	Matrices de taille n ou nm remplie de nombres aléatoires.

5.2. Accès aux éléments :

On peut accéder à un élément ou à une partie des éléments d'une matrice :

- On accède à l'élément qui se trouve sur la i^{éme} ligne et la j^{éme}colonne par l'instruction *A(i,j)*
- On extrait la n^{éme} ligne de A grâce à l'instruction : *A(n,:)*. Le résultat est donc un vecteur ligne.
- On extrait la n^{éme} colonne grâce à l'instruction *A(:, n)*. Le résultat est donc un vecteur colonne.

5.3. Opérations sur les matrices :

A'	Transposée de A
inv(A)	Inverse de A
A + B	Addition de A et B
A - B	Soustraction de A et B
A * B	Multiplication

- La suppression d'une colonne (Ligne) se fait par substitution avec une colonne (Ligne) vide.
- La commande `fliplr` (respectivement `flipud`) a pour effet de retourner les colonnes (respectivement les lignes) d'une matrice, par exemple si la i^{èmé} ligne de A est [a,b,c], alors la i ^{ème} ligne de fliplr(A) sera [c,b,a].
- La commande `reshape (x, m, n)` construit une matrice de taille m×n à partir du vecteur x (x doit avoir m*n éléments)
- - Une autre fonction utile est la fonction `repmat` qui (c permet de construire une matrice à partir de la répétition en ligne et en colonne d'une autre matrice.

[cf. Matlab]

6. Activité d'auto-évaluation

Exercice

C'est le module et l'argument du nombre complexe v.

C'est la matrice identité.

Le tri ascendant du variable v.

Matrice carré de dimension NxN contient que des uns.

Donne une descriptive sur la fonction cherchée.

Ones(N,N)	Sort (v)	Eye(N,N)	Abs(v) ; angle(v)	Help nom_du fonction

Exercice

Quel est l'effet de ce programme?

A=[2 3 4 ;3 -2 -2 ; 1 4 -3] ;

B=[17 ;3 ;1] ;

X=inv(A)*B

- Il ne joue aucun rôle
- Réaliser une multiplication entre deux variables
- Résoudre un système d'équation linéaire de trois variables

7.

Exercice 1 :

1. Générez la matrice M.
2. Générez la transposé de M.
3. trouvez le déterminant de M.

$$M = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

Exercice 2 :

1. Définir la matrice M.
2. Extraire la deuxième colonne de la matrice M, sa deuxième ligne et sa sous-matrice 3×4 centrale.
3. Inverser les 2 colonnes centrales de la matrice M et ensuite les deux premières lignes.
4. Extraire de cette matrice la matrice N , La matrice P , puis la matrice Q.

$$M = \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 \end{matrix}, N = \begin{matrix} 1 & 2 \\ 11 & 12 \\ 21 & 22 \end{matrix}, P = \begin{matrix} 8 & 9 & 10 \\ 18 & 19 & 20 \\ 28 & 29 & 30 \end{matrix}, Q = \begin{matrix} 3 & 7 \\ 23 & 27 \end{matrix}$$