1. Expression

Une "expression" MATLAB est une construction valide faisant usage de nombres, de variables, d'opérateurs et de fonctions.

Ex : rayon=4.5; pi*rayon^2 et sqrt((b^2)-(4*a*c)) sont des expressions

1.1. Nombres

1.1.1. Nombres complexes

MATLAB est aussi capable de manipuler des nombres complexes. Les nombres complexes sont écrits sous deux forme:

- ✓ la forme cartésienne $a + bi$, comme dans 1+2i
- ✓ la forme polaire: $r \times e^{it}$, comme 10*exp(pi/4*i)

Ex de nombres complexes valides (avec partie réelle et imaginaire) : 4e-13 - 5.6i , -45+5*j

1.1.2. Nombres réels

De façon interne, MATLAB stocke par défaut tous les nombres en virgule flottante "double précision" (8 octets ou 64 bits). On peut saisir les nombres réels selon:

- ✓ La convention de notation décimale: avec un point décimal facultatif '.' et le signe '+' ou '–' pour les nombres signés.
- ✓ La notation scientifique qui utilise la lettre 'e' pour spécifier le facteur d'échelle en puissance de 10.

Ex de nombres réels valides : 3 , -99 , 0.000145 , -1.6341e20 , 4.521e-5

1.1.3. Types entiers, 64/32/16/8 bits

On peut aussi manipuler des variables de types entiers stockées sur 8bits, 16 bits, 32 bits et 64 bits. Les opérations arithmétiques sur des entiers sont plus rapides que les opérations analogues réelles.

On dispose, pour cela:

- ✓ les fonctions de conversion int8 , int16 , int32 et int64 génèrent des variables entières signées stockées respectivement sur 8 bits, 16 bits, 32 bits ou 64 bits; les valeurs réelles sont arrondies au nombre le plus proche.
- ✓ les fonctions de conversion uint8 , uint16 , uint32 et uint64 génèrent des variables entières non signées (unsigned) stockées respectivement sur 8 bits, 16 bits, 32 bits ou 64 bits. Les valeurs réelles positives sont arrondies au nombre le plus proche (équivalent de round)

```
>> int8(7.8)
ans =
   8
>> int8(-9.2)
ans =
  -9
>> int8(200)
ans =
  127
>> int16(200)
ans =
   200
```

```
uint8(-9.7)
ans =
   0
>> uint8(10.7)
ans =
  11
>> uint8(200.5)
ans =
  201
>> uint8(300)
ans =
  255
```

IMPORTANT: Lorsque l'on utilise des opérateurs ou fonctions mélangeant des opérandes/paramètres de types

1.2. Variables

Le variable est définie lors de la première affectation, l'instruction = , opération qui consiste à spécifier son nom (identificateur) et à assigner sa valeur numérique ou son expression mathématique.

```
>>r=7.213   % Valeur
r=
   7.2130
```

```
>>s=r ^2*pi   % expression
s =
 163.4488
```

Si la variable existe déjà, MATLAB change son contenu et, si nécessaire, lui alloue une nouvelle capacité mémoire en cas de redimensionnement de tableau ou changer le type de la variable.

```
>> x= 17.09
x =
  17.0900
>> whos  x
 Name      Size       Bytes     Class           Attributes
 x          1x1          8        double
```

```
>> x= int32(10.07), whos  x
 x =
      10
Name      Size       Bytes    Class    Attributes

x       1x1           4         int32
```

Un nom de variable valide est commencé par une lettre. Il peut ensuite contenir des chiffres, des lettres et ou caractère souligné _.

Ex : noms de variables valides : **x_min** , **COEff55a** , **nom_de_variable, pr_var3int**

Ex : noms non valides : **91ab** (commence par un chiffre), **coe3f-55** (est considéré comme une expression), **cal_effectué** (contient un caractère accentué), **etud..per?** (contient des symboles)

Matlab est sensible à la case des variables. C'est à dire il différencie majuscules et minuscules: X33 et x33 désignent deux variables distinctes.

Il est inutile de spécifier le type ou la dimension de la variable que l'on manipule. 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.

Il existe cinq grands types de variables sous Matlab : les entiers, les réels, les complexes, les chaînes de caractères et le type logique.

Définissons une variable de chaque type :

```
>> x = 1.3; y = 3+i; s = 'bonjour';
>> b1 = (x>0); b2 = ischar(s);
>> e = int8(7.9);
```

La variable x est un réel, y un complexe, s une chaine de caractères, b1 et b2 sont deux manières de définir une variable logique et e est un entier codé sur 8 bits.

Toute variable est considérée comme étant un tableau des éléments d'un type donné. Trois formes particulières de tableaux sont existe:

- ✓ 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.

```
>> a = -17; b=9.13; z = 4-10j; w=14.05*exp(i*pi/10);      % Des scalaires
>> v =[-3.5  0  12  -7.025 cos(pi/7)  4^2/sqrt(9)]        % Vecteur ligne
>> w =[10; -14 ; log10(7.5); 7*(7+a/2)]                   % Vecteur Colone
>> Mat =[10 7.5 sqrt(7)*2^3; 20 45.16 0.27]               % Matrice
```

Pour se référer à un ensemble de variables (principalement avec commandes who , clear ...), on peut utiliser les caractères de substitution * (remplace 0, 1 ou plusieurs caractères quelconques).

Ex : si l'on a défini les variables x=14 ; ax=56 ; abx=542 ; , alors :

who *x liste toutes les variables x , ax et abx

Nous décrivons ci-dessous les commandes de base relatives à la gestion des variables.

1. Affectation

```
var = val   % affecte la valeur val à la variable var
var = expr % Affecte à variable var le résultat de l' expression expr, et affiche celui-ci
var = expr ;  % Affecte à variable var le résultat de l'expression expr, mais effectue cela sans afficher
              %du résultat à l'écran
```

2. La fonction deal

a) [var1, var2, ... varn] = deal(v1, v2, ... vn)

b) [var1, var2, ... varn] = deal(v)

La fonction deal permet d'affecter plusieurs variables en une seule instruction. Il faut que l'on ait exactement le même nombre d'éléments à gauche et à droite de l'expression (forme a)), ou que l'on ait un seul élément à droite (forme b)), sinon deal retourne une erreur.

Ex : [a, b, c] = deal(11, 12, 13) est identique à a=11, b=12, c=13

[a, b, c] = deal(10) est identique à a=b=c=10

3. Affichage

Pour afficher le contenu d'une variable var on saisie son nom ou bien on utilise la commande disp

var

disp(var)

4. Lister les variables

who {variable(s)}

Liste le nom de toutes les variables couramment définies dans le workspace (ou de la (des) variable(s) spécifiée(s))

whos {variable(s)}

Affiche une liste plus détaillée que who de toutes les variables couramment définies dans le workspace (ou de la (des) variable(s) spécifiée(s)) : nom de la variable, dimension, espace mémoire, classe.

5. Supprimer les variables

clear {variable(s)}

Efface du workspace toutes les variables (ou la(les) variable(s) spécifiée(s), séparées par des espaces et non pas des virgules !)

Ex : clear mat* détruit toutes les variables dont le nom commence par "mat"

1.2.1.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.

Exemple

```
>> x = 123; y = exp(log(x)); tst = ( x==y );
>> format long, x, y, tst
x =
   123
y =
   1.229999999999999e+002
```

```
tst =
    0
```

Quelque fonctions qui retournent une résultat de type logique (booléenne)

ischar(var)	Retourne 1 si var est une caractère, 0 sinon	ischar(3.15) retourne 0 ischar('a') retourne 1
isletter(var)	Retourne 1 si var est une caractère de l'alphabet, 0 sinon	ischar('J'ai 12 sur 20 ')
isspace(var)	Retourne 1 si var est une caractère de séparation (espace), 0 sinon	isspace('axt hh de ') retourne 0 0 0 1 0 0 1 0 0 1
isreal (var)	Retourne 0 si var est de type complexe, 0 sinon	x=7+2j, isreal(x) retourne 0 isreal('r') et isreal(5>0) retournent 1
isinteger(var)	Retourne 1 si var est un entier, 0 sinon	isinteger(5) retourne 0 isinteger(int16(5)) retourne 1
islogical(var)	Retourne 1 si var est de type booléenne, 0 sinon	islogical(0>=5) retourne 1 islogical(15) retourne 0
isnumeric(var)	Retourne 1 si var est une valeur numérique, 0 sinon	isnumeric (10>=5) retourne 0 isnumeric(10+9j) retourne 1
isequal(var1, var2)	Tester l'égalité entre var1 et var2	isequal('abc', 'abct') retourne 0 isequal([4 7 3], [4 9 3]) retourne 0

1.2.2.Le type chaîne de caractères

Une chaîne de caractères est définie comme un vecteur ligne contenant des caractères. Une donnée de type chaîne de caractères (char) est représentée sous la forme d'une suite de caractères encadrée d'apostrophes simples (’), où chaque caractère de la chaine occupe une cellule du vecteur.

Les apostrophes faisant partie de la chaine doivent êtres doublés, (sinon interprétés comme signe de fin de chaine, la suite de la chaine provoquant alors une erreur).

```
Ex: s1= 'la faculte de science exacte'
    s2= 'le departement d''economie'
```

La commande [s1 s2 s3...] concatène horizontalement les chaînes s1, s2, s3.....

```
Ex : soit s1=' AAA ', s2='CCC ', s3='EEE ' alors [s1 s2 s3] retourne '' AAA CCC EEE ''
```

Il est possible de manipuler (afficher, utiliser, modifier ou supprimer) chaque caractère de la chaîne en faisant référence à sa position dans la chaîne.

s(i)	signifie le caractère numéro i dans la chaine s
s(i: j)	signifie des caractères dans les cellules adjacents numéroté de i jusqu'à j dans la chaine s, s(i), s(i), s(i+1), s(i+2), s(i+3), ... s(j)
s(i : h : j)	séquence des caractères séparées par un pas fixé, de i jusqu'à j dans la chaine s, s(i), s(i+h), s(i+2*h), s(i+3*h), ...s(j)
s([i k m j])	Les caractères contenant dans les cellules i, k, m et j, s(i), s(k), s(m), s(j)

Exemple 01

```
>> ch = 'Module: outils de programmation 2019/2020'
ch =
Module: outils de programmation 2019/2020
>> length(ch)
ans =
41
>> ch(1)
ans =
M
>> a=ch(end)
a =
0
>> ch(length(ch))
ans =
0
>> ch(42)
??? Index exceeds matrix dimensions.
>> ch(0)
??? Subscript indices must either be real positive integers or logicals.
```

```
>> a=ch(3)
a =
d
>> ch(4) = 't'
ch =
Modtle: outils de programmation 2019/2020
>> ch(6) = 'a'
ch =
Modtla: outils de programmation 2019/2020
>> ch(9) = a
ch =
Modtla: dutils de programmation 2019/2020
>> ch(3)='tts'
??? In an assignment  A(:) = B, the number of elements in A and B must be the same.'
>> ch(3)=[]
ch =
Motla: dutils de programmation 2019/2020
```

Exemple 02

```
>> s ='chapitre 02 programmation avec
Matlab'
s =
chapitre 02 programmation avec Matlab
>> ch1=s(1:5)
ch1 =
chapi
>> s(6:9)='r'
s =
chapirrrr02 programmation avec Matlab
>> s(1:3)='abc'
s =
abcpirrrr02 programmation avec Matlab
>> s(length(s)-3:end)=''
s =
abcpirrrr02 programmation avec Ma
```

```
>> s ='chapitre 02 programmation avec
Matlab';
>> s(1:2:6)='x'
s =
xhxpxtre 02 programmation avec Matlab
>> s(10:3:17)='xwt'
s =
xhxpxtre x2 wrotrammation avec Matlab
>> s([7 1 3])=[]
s =
hpxte x2 wrotrammation avec Matlab
```

Ce tableau contient des fonctions manipulant les chaines de caractère:

length(ch)	Retourne le **nombre de caractères** de la chaîne *ch*
findstr(ch, s1)	Retourne, sur un vecteur ligne, la **position** dans *ch* de toutes les chaînes *s1* qui ont été trouvées. **Ex** : >>str='la faculte de la science de la nature et de la vie' >> findstr(str, 'd') ans = 12 26 42 >> findstr(str, 'la') ans = 1 15 29 45
strrep(ch, s1, s2)	Retourne une copie de la chaîne ch dans laquelle toutes les occurrences de s1 sont remplacées par s2 >>str='la faculte de la science de la nature et de la vie' >> strrep(str,'de','des') ans = la faculte des la science des la nature et des la vie
strcmp(ch1,ch2) isequal(ch1,ch2) strcmpi(ch1,ch2)	Compare les 2 chaînes ch1 et ch2: retourne 1 si elles sont identiques, 0 sinon. La fonction strcmpi ignore les différences entre majuscule et minuscule (''casse'') >> strcmp('abc','abc') ans = 1 >> strcmp('abc','abC') ans = 0 >> isequal('abc','abc') ans = 1 >> isequal('abc','abC') ans = 0 >> strcmpi ('abc','abC') ans = 1
strncmp(ch1, ch2, n) strncmpi(ch1,ch2,n)	Ne compare que les n premiers caractères des 2 chaînes >> strncmp('abc uuiitrp9***?', 'abcuuosc,,d2693*%', 3) ans = 1 La fonction strncmpi ignore les différences entre majuscule et minuscule (''casse'')
lower(ch)	Convertit la chaîne ch en minuscules, >> lower('1per Annee') ans = 1per annee
upper(ch)	Convertit la chaîne ch en majuscule, >> upper('1per Annee') ans = 1PER ANNEE

1.2.3.Type complexe

L'unité imaginaire est désignée par i ou j. Les nombres complexes peuvent être écrits sous forme cartésienne $a + i \times b$ ou sous forme polaire $r \times e^{it}$.

Les différentes écritures possibles sont a+j*b, a+b*i, et r*exp(i*t) ou r*exp(j*t) avec a, b, r et t des variables de type réel.

Opérations sur les nombres complexes:

```
>> z1 = 4.15 - 2.25i; z2= -5.25 + 7.25i;
1. Addition de nombres complexes:
>> z1+z2
ans =
 -1.1000 + 5.0000i
2. Soustraction de nombres complexes:
>> z1-z2
ans =
  9.4000 - 9.5000i
3. Multiplication de nombres complexes:
>> z1*z2
ans =
 -5.4750 +41.9000i
4. Division de nombres complexes:
>> z1/z2
ans =
  -0.4755 - 0.2281i
>> z1^z2
ans =
   0.0029 + 0.0102i
>> z1^2
ans =
  12.1600 -18.6750i
```

Fonctions relatives aux nombres complexes

— real et imag renvoient respectivement la partie réelle et la partie imaginaire du complexe passé en paramètre,

```
>> z1 = 4.15 - 2.25i;
>> x=real(z1), y= imag(z1)
x =
    4.1500
y =
   -2.2500
>> r =abs(z1), th= angle(z1)
r =
    4.7207
th =
   -0.4968
>> z1 = 4.15 - 2.25i;
>> w =conj(z1)
w =
   4.1500 + 2.2500i
>> z2=complex(4, 7)
ans =
   4.0000 + 7.0000i
>> x=real(w); y= imag(w);
>>[th, r] = cart2pol (x, y)
th =
    0.4968
r =
    4.7207
>> [x, y] =pol2cart(th, r)
x =
    4.1500
y =
    2.2500
```

— abs et angle renvoient respectivement le module et l'argument du complexe passé en paramètre,

— conj renvoie le complexe conjugué du nombre complexe passé en paramètre.

— z=complex(x, y): construire le variable z de type complexe à partir de la partie réelle x et la partie imaginaire y.

— [th, r] = cart2pol (x, y): calcule th (angle) et r (module) à partir de la partie réelle x et la partie imaginaire y.

—[x, y] =pol2cart(th, r): calcule x (la partie réelle) et y (la partie imaginaire) à partir de l'angle th et le module r.

1.3. Opérateurs

Les opérateurs arithmétiques sont

Fonctions	Définition
+	Addition
-	Soustraction
*	Multiplication
^	Puissance
/	Division
\	Division à gauche
‘	Transposé conjugué

Il utilise aussi des opérateurs logiques et relationnels :

Fonctions	Définition
< >	**différent de**
<= >=	**inférieur (supérieur) ou égal**
==	**égal**
~=	**différent**
&	**et logique**
|	**ou**
~	**complément logique (not)**
Xor	**ou exclusif**

1.4. Fonctions

MATLAB fournit un grand nombre de fonctions mathématiques élémentaires standard, y compris sin (sinus), cos (cosinus), tan (tangente), abs (valeur absolue ou module), sqrt (racine carrée), exp (exponentiel), ….. Pour obtenir une liste des fonctions mathématiques élémentaires, tapez :

>> help elfun

2. Formatage des nombres dans la fenêtre de commandes

Dans tous les calculs numériques, MATLAB travaille par défaut avec le format court avec quatre chiffres après la virgule.

On peut choisir le format d'**affichage des nombres** dans la fenêtre "Command Window" à l'aide de la commande **format**

Voici les commandes d'affichage

format short Affichage par défaut : notation décimale fixe à 4 chiffres significatifs

format short e : notation décimale (4 chiffres après virgule) avec exposant

format long Affichage précision max : notation décimale fixe à 15 chiffres significatifs

format long e => notation décimale (15 chiffres après virgule) avec exposant

format bank Format monétaire (2 chiffres après virgule) 72.35

format rat afficher les nombres sous forme d'une ration $\frac{a}{b}$

Exemple

```
>> 8/3
ans =
   2.6667
>> format long
>> 8/3
ans =
  2.666666666666667
>> format short e, 0.00079369
ans =
 7.9369e-004
```

```
>> format long e, 30000796347.34796
ans =
   3.000079634734796e+010
>> format bank, 8/3
ans =
        2.67
>> format rat, ans
ans =
      8/3
```

3. Caractères spéciaux dans les commandes MATLAB et Octave

Les caractères spéciaux ci-dessous sont particulièrement importants.

Caractère	Utilisation
;	1. Une commande terminée par ce caractère, sera normalement exécutée, mais son résultat ne sera pas affiché. **x=147.22;** 2. Ce caractère est utilisé comme de séparateur de commandes lorsque l'on saisit plusieurs commandes sur la même ligne. a=3*pi/19; b=tan(a); c=sind(30) 3. Utilisé pour séparer les éléments de vecteur colonne. **v=[14; 0.7 ; -3]** 4. Utilisé aussi comme caractère de séparation des lignes d'une matrice lors de la définition de ses éléments **M=[14 7; 0.7 0.14; -3 10]**

,	• séparateur de commande si on a plusieurs commandes sur la même ligne a=4 , b=5 • Utilisé aussi pour délimiter les indices de ligne et de colonne d'une matrice **A=[22.7 -2 7; 3.7 0.14 101; 7 -13 9], A(2, 3)** A(2,3) désigne l'élément de la matrice A situé à la 2e ligne et 3e colonne • Utilisé également pour séparer les différents paramètres d'entrée et de sortie d'une fonction. mod(3, 2)
:	• Opérateur de définition de séries et de plage d'indices des chaines de caractère, de vecteurs et matrices ch='bonjour', ch(1:3)
' (apostrophe)	• Caractère utilisé pour délimiter le début et la fin d'une chaîne de caractère ch='bonjour' • Également utilisé comme opérateur de transposition de matrice
.	Point décimale
%	• Ce qui suit est considéré comme un commentaire. Utile pour documenter un script ou une fonction (M-file) r=5.5 % r represente un rayon en [cm]
[ ]	Utiliser Pour définir de matrices ou vecteurs; v=[10 -4.5 2] Mat=[2 4; -3.14 -4.6]
( )	Gère la priorité des opérations arithmétique