Chapter 2

Calculating with MATLAB

Contents

2.1 A simple calculation

2.1.1 Variables in MATLAB

To create a variable, we use the simple structure: **variable = definition** without worrying about the type of the variable.

Figure 4.1 illustrates an example of variables under MATLAB.

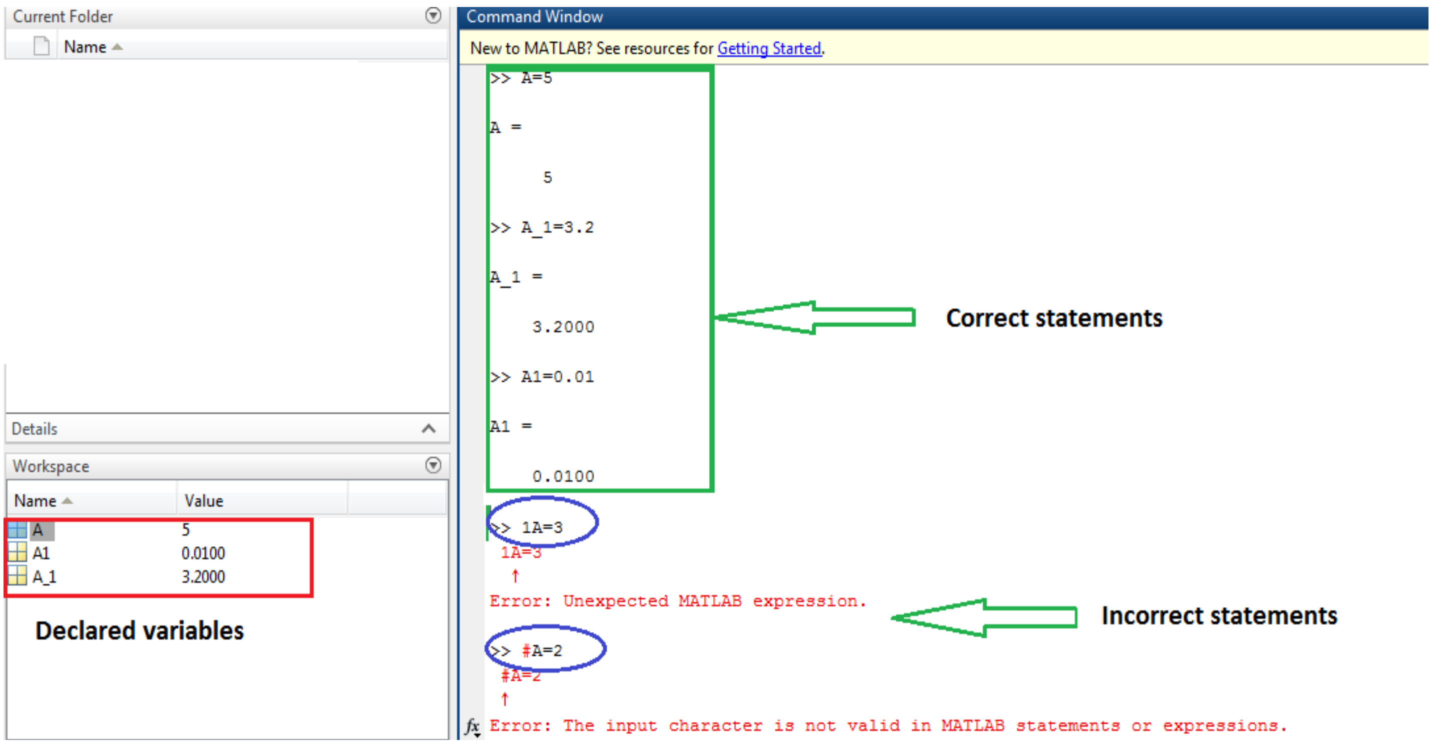

Figure 2.1: Example of variable reporting under MATLAB.

The name of a variable must contain only alphanumeric characters or the_ (underscore) symbol, and must begin with an alphabet.

We must also pay attention to capital letters because MATLAB is case sensitive (**A** and **a** are two different identifiers).

2.1.2 Numbers in MATLAB

— MATLAB uses conventional decimal notation, with an optional decimal point '.' and the sign '**+**' or '**–**' for signed numbers.

— Scientific notation uses the letter '**e**' to specify the power scale factor of 10.

— Complex numbers use the characters ' **i** 'and ' **j** '(indifferently) to designate the imaginary part.

There are five main types of variables in MATLAB: integers, reals, complexes, strings, and logical type.

```
» a = 1.3 ;
»b = 3+i;
»c = 'hello';
»d= logical(1);
»e = int8(2);
```

- *a* represents a real,
- *b* a complex,
- *c* a string,
- *d* is a logical variable (1=TRUE)
- *e* is an integer encoded on 8 bits.

The type of these different variables can then be checked using the **whos** function (See Figure 2.2).

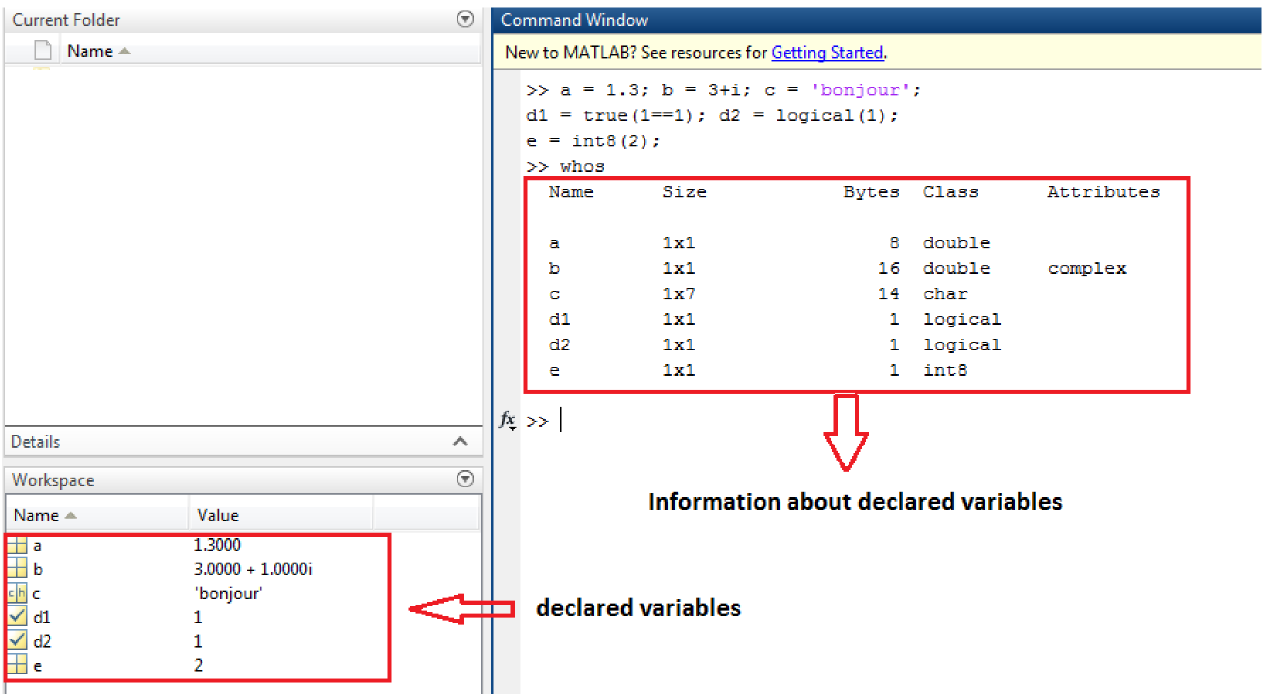

Figure 2.2: Example of variable reporting under MATLAB.

MATLAB always uses real numbers (double precision) to do the calculations, which makes it possible to obtain a calculation accuracy of up to 16 significant digits.

However, the following points should be noted:

- The result of a calculation operation is by default displayed with four digits after the decimal point.
- To display more digits, use the **long format** command (14 digits after the decimal point).
- To return to the default view, use the **short format** command.
- To display only 02 digits after the comma, use the **format bank** command.

— To display numbers as a ration, use the **format rat** command. Table 2.1 provides a summary:

Orders	Meaning
short format	Displays numbers with 04 digits after the decimal point
Long format:	Displays numbers with 14 digits after the decimal point
bank format	Displays numbers with 02 digits after comma
rat format	Displays numbers as a ration (a/b)

Table 2.1: Calculation accuracy.

2.1.3 Basic arithmetic operations

The basic operations in an expression are summarized in Table 2.3.

Transactions	Meaning
+	Addition
-	Subtraction
*	A multiplication
/	La division
\	Left division (or reverse division)
^	Power
ʻ	The transposed
( )	The parenthesis

Table 2.2: Basic operations under MATLAB.

2.1.4 Predefined Mathematical Functions

Some mathematical functions are illustrated are Table 2.3.

2.2 Vector calculation

2.2.1 Define a vector

A vector in MATLAB is a collection of elements of the same type. The simplest method to define a vector is to give its explicit description using the **[ ]** command, for example:

```
»vec1 = [1 2 11 0 0.3]
vec1 =
1.0000   2.0000   11.0000   0.0000   0.3000
```

A column vector can also be defined using the;

Function	usage
exp(x)	exponential of x
log(x)	natural logarithm of x
log10(x)	base 10 logarithm of x
x ∧ n	x raised to the power n
sqrt(x)	square root of x
abs(x)	absolute value of x
sign(x)	1 if x > 0 and 0 if x ≤ 0
sin(x)	sine of x
cos(x)	cosine of x
tan(x)	tangent of x
asin(x)	inverse sine of x (arcsin of x)
sinh(x)	hyperbolic sine of x
asinh(x)	inverse hyperbolic sine of x
round(x)	nearest integer to x
floor(x)	default rounding of x
ceil(x)	rounding up of x
rem(m,n)	remainder of the division of m by n
lcm(m,n)	least common multiple of m and n
gcd(m,n)	greatest common divisor of m and n
factor(n)	prime factorization of n
conj(z)	complex conjugate of z
abs(z)	modulus (magnitude) of z
angle(z)	argument (angle) of z
real(z)	real part of z
imag(z)	imaginary part of z

Table 2.3: Predefined mathematical functions in MATLAB.

```
»vec2 = [1; 2; 3;4 ]
vec2 =
1.0000
2.0000
3.0000
4.0000
```

2.2.2 Vector Operations

2.2.2.1 Vector concatenation

Two vectors can be concatenated:

```
»A = [1 2 3 4 5 6];
»B=[11 12 13];
»C=[A B]
C=
1   2   3   4   5   6   11    12      13
```

2.2.2.2 Vector Conversion

You can convert a row vector to a column or vice versa using transpose.

```
»A= [1 2 11 0 0.3];
»B =A'
B=
1.0000
2.0000
11.0000
0.0000
0.3000
```

2.2.2.3 Vector size

The *length()* function returns the size of a vector:

```
»A= [1 2 11 0 0.3];
length(A)
ans=
5
```

2.2.2.4 Generation of a vector of spaced elements

To generate a line vector of n elements linearly spaced between a and b, the *linspace(a,b,n)* function can be used:

```
»x=linspace(-5,5,7);
x=
-5.0000   -3.3333   -1.6667   0   1.6667   3.3333   5.0000
```

Another method for generating linearly spaced vectors is to use **[a:s:b]**. We then create a vector between a and b with a spacing s:

```
»vec=[1 :2 :10]
Vec=
1   3   5 7    9
```

2.2.2.5 Special vectors predefined in MATLAB

- *ones(1,n)*: line vector of length n all elements of which are equal to 1.

```
»X=ones(1.5) X
=
1   1   1   1   1
```

- *zeros(1,n)*: line vector of length n all elements of which are equal to 0.

```
»Y=zeros(1,4)
Y =
0   0   0   0
```

- *rand(1,n)*: line vector of length n whose elements are randomly generated between 0 and 1.

```
»Z=rand(1.6)
Z =
0.8147   0.0975   0.1576   0.1419   0.6557   0.7892
```

2.2.2.6 Arithmetic operations

The usual algebraic operations +, -, *, / should be taken with caution for vectors. Sum and difference are term-to-term operations, and therefore require vectors of the same dimension. The product * is the matrix product. We will come back to this in the section on matrices. To use multiplication or division terms to terms we must replace * by .* and / by ./

In the same way as for scalars, all the mathematical functions previously defined for vectors can be applied.

```
»A = [1 3 5 6]; B = [10 20 30 40];
»A + B
ans = 11   23   35    46
» A-B
ans= -9   -17    -25    -34
»A.* B
ans=  10    60     150    240
»B./A
ans=  10.0000        6.6667        6.0000              6.6667
```

Vector-specific mathematical functions:

There are also commands that are vector-specific (see Table 2.5)

Functions	use
sum(x)	sum of the elements of vector x
prod(x)	produces elements of vector x
max(x)	largest element of vector x
min(x)	smallest element of vector x
mean(x)	average of the elements of the vector x
spell(x)	orders the elements of the vector x in ascending order
fliplr(x)	reverses the order of the elements of vector x

Table 2.4: Vector-specific mathematical functions.

2.2.3 Manipulate a vector

It is also important to become familiar with vector manipulation, i.e. being able to extract subsets using clues. The k^{th} element of a vector A can be displayed using the command $A(k)$. k must be an integer otherwise MATLAB will return an error:

```
»A=[1 4 5 6 9 3 10 11];
» A(3)
ans= 5
» A(2.3)
Subscript indices must be real positive integers or logical.
```

Index vectors can also be used to extract a sub-vector:

```
»B=[1 2 0 3 5 6 9 3 10 11 15 16];
»B(3 :7)
ans=0   3   5   6   9
```

2.3 Matrix calculation

2.3.1 Define a matrix

A matrix will be defined in a similar way to a vector with the command [].
The matrix X is defined:

$$X = \begin{pmatrix} 0 & 8 & 1 & 9 \\ 1 & 3 & 7 & 6 \\ 4 & 0 & 11 & 2 \end{pmatrix}$$

```
»X=[0 8 1 9; 1 3 7 6; 4 0 11 2]
X=
0   8   1   9
1   3   7   6
4   0   5   2
```

A matrix is composed of m rows and n columns. If we want to know the value of m or n, we use the **size(X)** command:

```
»X=[0 8 1 9; 1 3 7 6; 4 0 11 2]
X=
0   8   1   9
1   3   7   6
4   0   5   2
»[m n]=size(X)
m=3
n = 4
```

A *block* matrix can be constructed very simply. If A, B, C, D designate 4 matrices (with compatible dimensions), we define the blocks matrix:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

by the instruction **M = [A B; C D]**.

```
»A=[1 2; 3 4];
»B=[5 6;7 8];
»C=[ 9 10; 11 12];
»D=[13 14; 15 16];
»M = [A B; C D]
M=
1   2   5   6
3   4   7   8
9   10   13   14
11 12    15   16
```

2.3.2 Matrix Operations

Addition and subtraction operations

These operations are only possible on matrices of identical size. These are term-to-term operations, similar to scalar operations. For example: $\begin{pmatrix} -1 & 5 \\ 0 & 2 \end{pmatrix} + \begin{pmatrix} 2 & 3 \\ -4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 8 \\ -4 & 3 \end{pmatrix}$

```
»A=[-1 5;0 2];
»B=[2 3;-4 1];
»C= A+B
 C=
  1   8
-4    3
```

The product operation

On the other hand, multiplication deserves special attention. There are two types of multiplication: so-called matrix multiplication and term-to-term multiplication.

Term-to-term multiplication: is the analog of addition and subtraction seen above. Under MATLAB, it is scored specifically to distinguish it from true matrix multiplication: A.*B.

```
»A=[-1 5;0 2];
»B=[2 3;-4 1];
»C= A.*B
C=
-2   15
0    2
```

Similarly, if it is desired to obtain the square of a matrix (in the sense of the product terms to terms of this matrix by itself) we write A.^2

```
»A=[-1 5;0 2];
»C= A.^2
ans=
1   25
0   4
```

The matrix product: it is a (non-commutative) product between the matrix A of size $m \times n$ and the matrix B of size $n \times p$ is a matrix $C = AB$ of size $m \times p$ (See Figure 2.3. So that this product is defined, it is necessary that the number of columns of A is equal to the number of rows of B. If we note the elements of A: a_{ij}, and those of B: b_{ij}, then the elements of the matrices C are given by the following formula: $c_{ij} = \Sigma_{0<k<n} \ \boldsymbol{a_{ik}b_{kj}}$

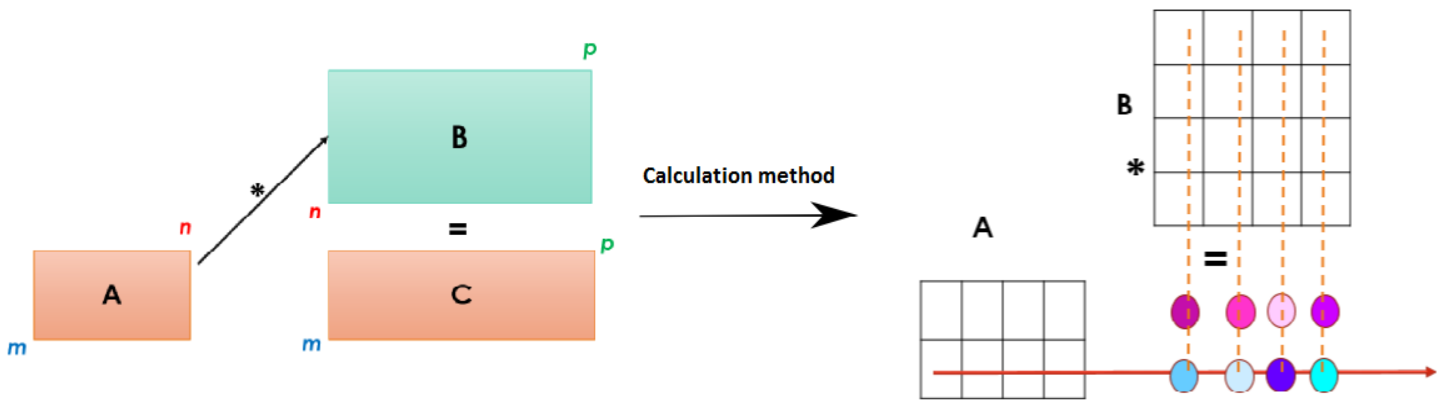

Figure 2.3: Matrix Product Principal.

Figure 2.4 shows an example of the matrix multiplication principle.
Under MATLAB, the matrix product is calculated by simply using the sign A*B:

```
»A = [4 2; 0 1];
»B = [1 2 3; 5 4 6];
»C=A*B
C =
14   16   24
5   4   6
```

Inverse operation and division

We denote A^{-1}, the inverse of A (when it exists) and we define A^{-1} by:
$\boldsymbol{A^{-1}A = AA^{-1} = I}$ where I is the identity matrix.

```
»A = [4 2; 0 1];
»X = inv(A)
X=
0.2500          -0.5000
0               1.0000
```

The division is defined from the reverse: $\boldsymbol{A/B = AB^{-1}}$
It therefore requires that B be invertible and that the dimensions of A and B be compatible.

```
»A = [4 2; 0 1];
»B = [1 2; 5 4];
»C = A/B
C=
-1.0000   1.0000
0.8333   -0.1667
```

Matrix-specific functions

As for vectors, there are predefined matrices:

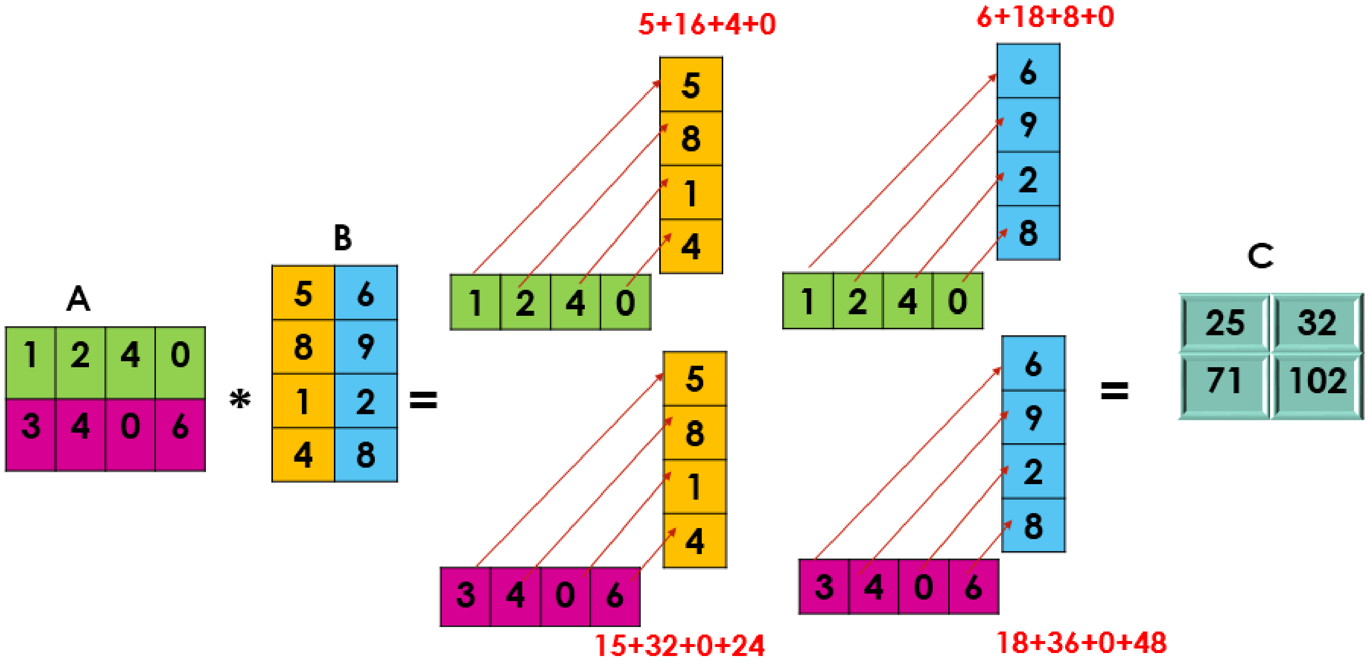

Figure 2.4: Example of the matrix product between two matrices *A* and *B*.

Function	use
eye(n)	the identity matrix (square of size n)
ones(m,n)	the matrix with m rows and n columns of which all the elements are equal to 1
zeros(m,n)	the matrix with m rows and n columns of which all the elements are equal to 0
rand(m,n)	a matrix with m rows and n columns whose elements are generated randomly between 0 and 1.
magic(n)	a magic matrix of dimension n.

Table 2.5: Predefined functions specific to the mat