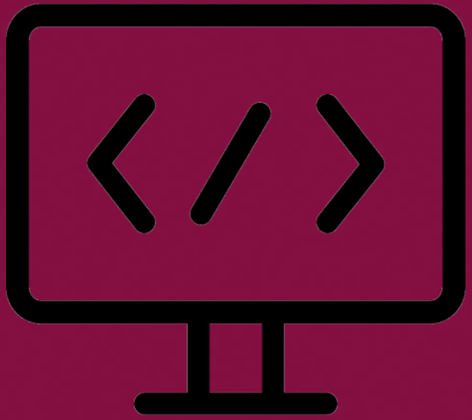
SEEE1022 INTRODUCTION TO SCIENTIFIC PROGRAMMING

UNIVERSITI TEKNOLOGI MALAYSIA

CH3
Operation



Dr. Mohd Saiful Azimi Mahmud (azimi@utm.my)
P19a-04-03-30, School of Electrical Engineering, UTM

OBJECTIVES

1. To know the three types of operation (statement, function, command).
2. To know the three types of statement (assignment, repetition, decision).
3. To understand the operation of arithmetic expression on both matrix and array operators.
4. To understand the operation of Boolean expression on both relational and logical operators.

OPERATION

STATEMENT

1. ASSIGNMENT

Assign array to a variable using EQUAL OPERATOR

syntax

```
variable = expression
```

TYPE
- **Arithmetic**
- **Boolean**

2. REPETITION

Execute statements specified number of times using BOOLEAN OPERATOR or EQUAL OPERATOR

```
while
for
```

3. DECISION

Execute statements if condition is TRUE using BOOLEAN OPERATOR

```
if
if-else
if-elseif
Switch-case
```

FUNCTION

Group of statement that perform a task

syntax

```
[output1,…,outputM]
     = functionName(input1 … inputN)
```

COMMAND

Function that mostly used to change the environment

syntax

```
functionName input1 … inputN
```

Input, if exist, must only be string

ARITHMETIC OPERATION

ARITHMETIC OPERATIONS

INTRODUCTION

- MATLAB has two different types of arithmetic operations: array operations and matrix operations.

Matrix Operations

- Follow the rules of linear algebra.

Array Operations

- Execute element by element operations and support multidimensional arrays.

- The period character (.) distinguishes the array operations from the matrix operations.
- However, since the matrix and array operations are the same for addition and subtraction, the character pairs .+ and .- are unnecessary.

ARITHMETIC OPERATOR

OPERATOR TYPES

Matrix Operator

Operation	Algebraic	MATLAB
Addition	$a + b$	$a + b$
Subtraction	$a - b$	$a - b$
Multiplication	$a \times b$	$a * b$
Right Division	a/b	a/b
Left Division	b/a	$a\backslash b$
Power	a^b	a ^ b

Array Operator

Operation	MATLAB
Multiplication	$a.* b$
Right Division	$a./b$
Left Division	$a.\backslash b$
Power	$a.$^b
Pure Transpose	$a.'$

*A.K.A. Element-by-Element Operator

Other Matrix Operator

Operation	MATLAB
Colon	$a: b$
Transpose	a'

TYPE

PRECEDENT

No	Operation
1	Transpose
2	Parentheses
3	Power, left to right
4	Multiplication & Division, left to right
5	Addition & subtraction
6	Colon

MATRIX OPERATOR

MATRIX OPERATOR : ADDITION

EXAMPLE 1

```
>> a = [1 2 3]
a =
     1     2     3

>> b = a + 5
b =
     6     7     8
```

- Addition with scalar is performed on all elements of the array.

EXAMPLE 2

```
>> a = [1 2;3 4];
>> b = [4 5;6 7];

>> c = a + b
c =
     5     7
     9    11
```

- Two matrices are added according to the corresponding elements of the two matrices.
- Both matrices must have the same size.

MATRIX OPERATOR : ADDITION

EXAMPLE 3

```
>> a = [1 2 3]
a =
     1     2     3

>> b = a + a'
Error using  +
Matrix dimensions must agree.
```

- Addition between array needs both arrays to be compatible in size. In this example, `a` is of size `1x3` and `a'` is of size `3x1` are not compatible.

* Please check MATLAB documentation for the full list of Compatible Array Sizes for Basic Operations.

MATRIX OPERATOR : MULTIPLICATION

EXAMPLE 4

```
>> x = 2*(3+2)
x =
    10

>> x = 2(3+2)
2(3+2)
  |
Error: Unbalanced or unexpected parenthesis or bracket.
```

- Parentheses alone does not represent multiplication.

EXAMPLE 5

```
>> a = [1 2 3];
>> b = [4 5 6];

>> c = a*b
Error using  *
Inner matrix dimensions must agree.
```

- * mean matrix multiplication. a and b are not the correct size for matrix multiplication in this example.

MATRIX OPERATOR : MULTIPLICATION

EXAMPLE 6

```
>> a = [1 2 3];
>> b = [4;5;6];

>> c = a*b
c =
     32
```

- This is a valid matrix multiplication where `1x3` column vector `a` is multiplied with `3x1` row vector `b`.

EXAMPLE 7

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

>> c = a*b
c =
     40     46
     76     88
```

- In general, the number of column in the first matrix must be equal to the number of rows in the second matrix

MATRIX OPERATOR : TRANSPOSE

EXAMPLE 8

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

>> b = a'
b =
     4     3     2
     5     1     9
     2     7     6
```

- Transpose of a matrix is obtained by interchanging the rows and column.

EXAMPLE 9

```
>> a = [1 2 3];
>> b = a*a'
b =
    14
```

MATRIX OPERATOR : PRECEDENCE

EXAMPLE 10

```
>> y = 5*6/6*5
y =
    25

>> z = 6*5/5*6
z =
    36
```

- Since multiply and division have the same precedence, the operation compute from left to right

EXAMPLE 11

```
>> y = -4^2
y =
    -16
```

- ^ has higher precedence than –

MATRIX OPERATOR : PRECEDENCE

EXAMPLE 12

```
>> y = (5*6)/(6*5)
y =
     1

>> z = (6*5)/(5*6)
z =
     1
```

- To avoid confusing on the precedence, use parentheses.

EXAMPLE 13

```
>> y = (-4)^2
y =
     16
```

- Parentheses () has higher precedence than ^

MATRIX OPERATOR : STRING

EXAMPLE 14

```
>> y = 'A' + 1
y =
     66

>> z = char(y)
z =
     B
```

- Operation between string and numeric will return numeric value.

EXAMPLE 15

```
>> y = 'hello' + 1
y =
     105    102    109    109    112
```

ARRAY OPERATOR

ARRAY OPERATOR : SCALAR

EXAMPLE 16

```
>> a = 2 + [2 4 5 6]
a =
     4      6      7      8

>> a = 2 .+ [2 4 5 6]
a =
     4      6      7      8


>> b = 2 * [2 4 5 6]
b =
     4      8     10     12

>> b = 2 .* [2 4 5 6]
b =
     4      8     10     12
```

- Any operation between scalars and non-scalars can be written with or without the period (.).

ARRAY OPERATOR : ARRAY SIZE

EXAMPLE 17

```
>> a = [2 4 5 6];
>> b = [2 1 2 1];

>> c = a.*b
c =
     4      4      10      6
```

- Elements in a and b are multiplied accordingly.
- Both array must have equal size.
- Recap: a*b need the column no. of a and row no. of b to be equal

EXAMPLE 18

```
>> a = [2 4 5 6];

>> c = a.*a'
Error using  .*
Matrix dimensions must agree.
```

- a and a' are having equal number of elements but not at equal size.

VECTORIZING : COMPOUND INTEREST

- With data in MATLAB constructed as array, a formula can be evaluated for a large set of values at once. This is called **vectorizing**. In vectorizing, element by element operation will be used.

Example 19

- Lets consider a formula of compound interest as below where $A =$ invested money, $r =$ interest rate, $n =$ total year, and $B =$ final balance:

$$B = A(1 + r)^n$$

- If $A = 100$, this scalar value will result $B = 236.7$ for $r = 0.09$ and $n = 10$.
- To compute for several values of A, vectorizing will be the most useful instead of computing for several times. Now, lets represent A as a vector with 5 values:

$$A = [100, 200, 500, 1000, 4000]$$

- Thus, evaluating B based on the vector A will give below result:

$$B = [236.7, 473.5, 1183.7, 2367.4, 9469.5]$$

VECTORIZING : COMPOUND INTEREST

- The MATLAB code is as follows:

```
>> r = 0.09;
>> n = 10;

>> A1 = 100; %single invested value
>> B1 = A1*(1+r)^n
B1 =
   236.7364

>> A2 = [100,200,500,1000,4000]; %vectorizing invested value
>> B2 = A2*(1+r)^n
B2 =
   1.0e+03 *

   0.2367     0.4735     1.1837     2.3674     9.4695
```

VECTORIZING : VERTICAL DISPLACEMENT

EXAMPLE 20

If a stone is thrown vertically upward, its vertical displacement `s` after an elapsed time `t` is given by the formula $s = gt^2/2$ where `g` is the acceleration due to gravity with value `9.81`. The structure plan for this problem is as follows:

1. Assign the data (`g` and `t`) to MATLAB variables.
2. Calculate the value of `s` according to the formula.

```
>> g = 9.81;
>> t = 0:5   %vectorizing t for 6 values of elapsed time
t =
     0      1      2      3      4      5

>> s = g*t.^2/2
s =
     0    4.9050   19.6200   44.1450   78.4800  122.6250
```

- Since the square operation must be done to every `t` value, array operation is used.

VECTORIZING : VOLUME OF CONES

EXAMPLE 21

- Supposed the diameter and height of a cone is D and H respectively. Then, the volume (V) of the cones can be computed as:

$$V = \frac{1}{12}\pi D^2 H$$

- If you have 5 different size of cones and want to calculate their volumes, in most programming language, you need to set up a loop where D and H are constructed as vectors:

for $n = 1\ to\ 5$

$$V(n) \leftarrow \frac{1}{12}\pi D(n)^2 H(n)$$

end

- By using array operator, you can avoid the loop and calculate all the volumes at once.
- For some values of D and H, the MATLAB code will be as follow:

```
>> D = [1.00, 0.50, 3.00, 1.20, 2.00];
>> H = [2.00, 4.00, 1.00, 1.00, 2.00];
>> V = 1/2*pi*(D.^2).*H
V =
    0.5236      0.2618      2.3562      0.3799      2.0944
```

COMPLEX NUMBER ARITHMETIC

INTRODUCTION

- Complex numbers are numbers that consist of two parts, a real number and an imaginary number in the form of $a+bi$.
- a and b are real numbers, and i is the imaginary component where $i=\sqrt{(-1)}$
- For complex matrices, the operations ' and .' behave differently.
- The ' operator is the complex conjugate transpose where the signs of imaginary parts are changed.
- The .' operator does a pure transpose.

COMPLEX NUMBER ARITHMETIC

EXAMPLE 22

```
>> a=[1+i 2+2i; 3+3i 4+4i]
a =
    1+1i     2+2i
    3+3i     4+4i

>> a'
ans =
    1-1i  3-3i
    2-2i  4-4i

>> a.'
ans =
    1+1i  3+3i
    2+2i  4+4i
```

- The array operator will return the original values while matrix operator return the matrix conjugates.

ARITHMETIC OPERATOR DRILL

$$a = 2; \quad b = [2 \quad 4]; \quad c = \begin{bmatrix} 7 & 4 \\ 1 & 3 \end{bmatrix};$$

Based on the above, evaluate by hand of the following expressions:

```
1. a+2-3
2. a+2*3
3. 2*a^3+a
4. a+b
5. b*c
6. c'
7. b*c'
8. b'+a/2+2
9. c'*c
10.a+1:a^3/2
11.a+(1:a^3)/2
12.(a:a+1)*c
13.c^2^a
14.c*c
15.c.*c
16.b.^b
17.a+b.^b*c'
18.(a+2i)'
19.(a+2i).'
20.[a+2i,a+3i]'
21.[a+2i;a+3i].'
```

BOOLEAN OPERATION

BOOLEAN OPERATOR

INTRODUCTION

- Boolean algebra is a mathematical operation that return a logical value, which are either `true(1)` or `false(0)`.
- Thus, in programming language, the assignment from Boolean expression will return a logical data type.
- There are two types of Boolean operator in programming language:
  1. Relational Operator.
  2. Logical Operator.

- Common usage of Boolean operators are:
  1. Identify particular elements from an array.
  2. Describing decision and repetition statements (next week topic).

BOOLEAN OPERATOR

INTRODUCTION

Relational Operator

Symbol	Meaning
<	Less than
<=	Less than or equal
==	Equal
~=	Not equal
>	Greater than
>=	Greater than or equal

Logical Operator

Logical	Element-wise	Short-circuiting
AND	&	&&
OR	|	||
NOT	~	~

Precedence

No	Operation
1	() Parentheses
2	^ .^ Power, ' .' Transpose
3	~ NOT
4	* .* Multiply, / ./ \ .\ Divide
5	+ Addition, − Subtraction
6	: Colon
7	Relational
8	& AND
9	| OR

Logical Value

Function	Value
`true`	1
`false`	0

TRUTH TABLE FOR LOGICAL OPERATIONS

INTRODUCTION

Input A	Input B	AND A&B	OR A|B	NOT ~A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

- If the input is numeric data type, all nonzero values will be assign as logical `1` and zero value as logical `0` .

RELATIONAL OPERATOR

EXAMPLE 23

```
>> a = 5;
>> b = 10;
>> c = a>b
c =
     0
```

- Since `a>b` is wrong, then the output is `0(false).`

EXAMPLE 24

```
>> a = [1 4 5 2];
>> b = 4;

>> c = a<b
c =
     1      0      0      1

>> d = b<=a
d =
     0      1      1      0
```

- Operation with scalar is performed on all elements of the array.

LOGICAL OPERATOR

EXAMPLE 25

```
>> a = [0 0 1 0 1];
>> b = [1 0 1 1 0];
>> c = a|b
c =
     1     0     1     1     1
```

- OR operation will return 1 if either of the input is 1.

EXAMPLE 26

```
>> a = [1 4;5 2]
a =
     1     4
     5     2

>> b = a & [0 2;4 0]
b =
     0     1
     1     0
```

- For operation on numeric data type, all nonzero values will be assign as logical 1 and zero value as logical 0 .

OPERATORS PRECEDENCE

EXAMPLE 27

```
>> a = [1 6 5];
>> b = [3 4 5];

>> c = a>b~=5
c =
     1     1     1

>> c = a~=b>5
c =
     0     0     0
```

- The operation executed from left to right.

EXAMPLE 28

```
>> a = 1&0|0
a =
     0
>> a = 1|0&0
a =
     1
```

- & has higher precedence than |

MIX OPERATORS

- In decision and repetition statements, which will be discuss in the next chapter, mixing the relational and logical operator will be very useful in having more than one condition at once.
- For now, we will discuss on how those two can be mixed into single expression.

EXAMPLE 29

```
>> a = [1 4 5 2 9 2 5 7];

>> b = a>2 & a<8
b =
     0     1     1     0     0     0     1     1
```

- `b` is identifying elements of `a` with values between `3` & `7`

MIX WITH ARITHMETIC OPERATORS

EXAMPLE 30

```
>> a = [0 0 1 0 1];
>> b = [1 0 3 2 0];
>> c = a|b-3
c =
     1     1     1     1     1

>> d = (a|b)-3
d =
    -2    -3    -2    -2    -2


>> whos c
  Name      Size            Bytes  Class      Attributes
  c         1x5                 5  logical

>> whos d
  Name      Size            Bytes  Class     Attributes
  d         1x5                40  double
```

- When both arithmetic and Boolean operators are used in single expression, the output data type depends on the last operator executed.

IDENTIFY ELEMENTS

EXAMPLE 31

```
>> A = randi(15,3)
A =
    12    10     1
    12     3     5
     6    11     1

>> B = A<9
B =
     0     0     1
     0     1     1
     1     0     1

>> A(B)
ans =
     6
     3
     1
     5
     1
```

- Relational operator can be use to identify elements based on some condition.
- In this example, `B` has the information on which elements are required.
- Then, by using logical indexing, elements of `A` which are `<9` can be extracted.
- Note that when no equal operator present, MATLAB automatically assign the value to variable name `ans`.

IDENTIFY ELEMENTS

EXAMPLE 32

```
>> A = randi(15,3)
A =
    12    10     1
    12     3     5
     6    11     1

>> B = A<=4 | A>11
B =
     1     0     1
     1     1     0
     0     0     1

>> A(B)
ans =
    12
    12
     3
     1
     1
```

- By mixing relational and logical operator, elements of `a` which are not between certain range can be identified.
- In this example, the range is between `4&11`.

BOOLEAN OPERATOR DRILL

Determine the following value of `x` before checking your answer with MATLAB.

1. `x = 3>2`
2. `x = 2>3`
3. `x = -4<=-3`
4. `x = 1<1`
5. `x = 2~=2`
6. `x = 3==3`
7. `x = 4>=3~=2`
8. `x = 0<0.5<1`
9. `x = 0<1&1<2`
10. `x = 1<2|1>4`
11. `x = a<2|a>4`
    lists values of `a` that gives `false` to `x`
12. `x = 1|0&1`
13. `x = 2&3|0`
14. `x = ~2&3|0`
15. `x = 2&~3|0`
16. `x = 2&3|3>2`
17. `x = 2+3>2`
18. `x = 3-~2==4<2^2`
19. `x = [1 2]&[1;2]`
20. `x = [1:0.2:2]<2`

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

univteknologimalaysia

utm_my

utmofficial

Thank You

www.utm.my

innovative ● entrepreneurial ● global