Xi'an Jiaotong-Liverpool University
西交利物浦大学
UNIVERSITY OF LIVERPOOL

EEE116
Experimental, Computer Skills and Sustainability
Matlab Tutorial No.2
(2017 – 2018 Semester 2)

Yang DU[1], Haochuan JIANG[2]
[1]: yang.du@xjtlu.edu.cn, [2]: haochuan.jiang@xjtlu.edu.cn
Department of Electric and Electronic Engineering
Xi'an Jiaotong – Liverpool University
Mar. 23th, 2018

*Acknowledge: the teaching materials of this module are originally prepared by Dr. Siyi WANG

Outline

1. Arrays (vector / matrix / Tensor, and relevant operations);
2. Relational and logical operators;
3. Branch statement;
4. Loops;

Array (Vector / Matrix / Tensor)

Array is the fundamental data unit in Matlab.

Vector: 1-D
Matrix: 2-D
Tensor: N-D, N>=3;

Creating Vector

To create a row vector:

```
>> a = [3 7 9]

a =

     3     7     9
```

To create a column vector:

```
>> b = [3;7;9]

b =

     3
     7
     9
```

Separate elements by semicolons

or

```
>> b = [3
7
9]

b =

     3
     7
     9
```

[Enter]

Creating Matrix

First way to create a 2×3 matrix A

```
>> A = [2 4 10;3 5 7]

A =

     2     4    10
     3     5     7
```

Separate row by semicolons

Second way to create a 2×3 matrix A

```
>> A = [2 4 10
3 5 7]

A =

     2     4    10
     3     5     7
```

[Enter]

Transpose

Transpose: interchanges the rows and columns:

$$A = \begin{bmatrix} 2 & 4 & 10 \\ 3 & 5 & 7 \end{bmatrix} \qquad A^T = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 10 & 7 \end{bmatrix}$$

MATLAB operator: A'

Complex conjugate transpose (Hermitian):

$$B = \begin{bmatrix} 2+3j & 6+j \\ 1-5j & 5-3j \end{bmatrix} \qquad B^H = \begin{bmatrix} 2-3j & 1+5j \\ 6-j & 5+3j \end{bmatrix}$$

MATLAB operator: B'

Slicing Indices

Given vector $v = \begin{bmatrix} 2 & 3 & 7 & 9 & 0 \end{bmatrix}$

- $v(5)$–the 5th element in v

```
>> v(5)

ans =

     0
```

- $v(2:5)$–the 2nd through 5th elements

```
>> v(2:5)

ans =
```

Slicing Indices

Given matrix $C = \begin{bmatrix} 2 & 4 & 10 & 13 \\ 16 & 3 & 7 & 18 \\ 8 & 4 & 9 & 25 \\ 3 & 12 & 15 & 17 \end{bmatrix}$

- $C(:,3)$–all the elements in the 3rd column of C
- $C(:,2:4)$–all the elements in the 2nd through 4th columns of C
- $C([1 \quad 3],1:3)$–all the elements in the 1st and 3rd rows that are also in the 1st through 3rd columns

```
>> C(:,3)

ans =

    10
     7
     9
    15

>> C(:,2:4)

ans =

     4    10    13
     3     7    18
     4     9    25
    12    15    17

>> C([1 3],1:3)

ans =

     2     4    10
     8     4     9
```

Deleting Elements

- The empty or null array is expressed as [ ].
- To delete rows or columns of

$$C = \begin{bmatrix} 2 & 4 & 10 & 13 \\ 16 & 3 & 7 & 18 \\ 8 & 4 & 9 & 25 \\ 3 & 12 & 15 & 17 \end{bmatrix}$$

- $C(3,:)$=[ ]–deletes the 3rd row
- $C(:,2:4)$=[ ]–deletes the 2nd through 4th columns

```
>> C(3,:) = []

C =

     2     4    10    13
    16     3     7    18
     3    12    15    17

>> C(:,2:4) = []

C =

     2
    16
     3
```

Vector / Matrix Shape

- *length*(v) – returns the number of elements in vector $\boldsymbol{v}$.
- *size*(A) – returns a row vector $[m \quad n]$ containing the sizes of the $m \times n$ array $\boldsymbol{A}$.

```
>> v = [2 3 7 9 0];
>> length(v)

ans =

     5

>> A = [2 4 10;3 5 7];
>> size(A)

ans =

     2     3
```

Vector / Matrix Shape

Say M is a tensor (3-D)

length(M)?

Return the **longest dimension** of the matrix / tensor

Scalar-Array Operation

Scalar-Array Addition/Subtraction:

```
>> A = [2 5;6 9];
>> A+3

ans =

     5     8
     9    12
```

Scalar-Array Multiplication:

```
>> 5*A

ans =

    10    25
    30    45
```

Recall: Matrix / Element-wise Multiplication

Matrix multiplication:

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} * \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}1\times5+2\times7 & 1\times6+2\times8\\3\times5+4\times7 & 3\times6+4\times8\end{bmatrix} = \begin{bmatrix}19 & 22\\43 & 50\end{bmatrix}$$

Element-wise multiplication:

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} .* \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}1\times5 & 2\times6\\3\times7 & 4\times8\end{bmatrix} = \begin{bmatrix}5 & 12\\21 & 32\end{bmatrix}$$

Element-wise Operation

If $$\boldsymbol{A}=\begin{bmatrix}2 & 5\\ 6 & 9\end{bmatrix}, \qquad \boldsymbol{B}=\begin{bmatrix}9 & 8\\ -12 & 14\end{bmatrix}$$

$D = A. * B$ gives the array multiplication result

$$\boldsymbol{D}=\begin{bmatrix}2(9) & 5(8)\\ 6(-12) & 9(14)\end{bmatrix}=\begin{bmatrix}18 & 40\\ -72 & 126\end{bmatrix}$$

$E = A./B$ gives the array division result

$$\boldsymbol{E}=\begin{bmatrix}2/9 & 5/8\\ 6/(-12) & 9/14\end{bmatrix}=\begin{bmatrix}0.2222 & 0.6250\\ -0.5000 & 0.6429\end{bmatrix}$$

Element-wise Operation

To perform exponentiation on an element-by-element basis: If

$$\boldsymbol{A} = \begin{bmatrix} 2 & 5 \\ 6 & 9 \end{bmatrix}$$

$B = A.\hat{}3$ gives

$$\boldsymbol{B} = \begin{bmatrix} 2^3 & 5^3 \\ 6^3 & 9^3 \end{bmatrix} = \begin{bmatrix} 8 & 125 \\ 216 & 729 \end{bmatrix}$$

Array Addition / Subtraction

Array Addition:

$$\underbrace{\begin{bmatrix} 2 & 5 \\ 6 & 9 \end{bmatrix}}_{A} + \underbrace{\begin{bmatrix} 9 & 8 \\ -12 & 14 \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} 11 & 13 \\ -6 & 23 \end{bmatrix}}_{C}$$

```
>> A = [2 5;6 9];
>> B = [9 8;-12 14];
>> C = A+B

C =

    11    13
    -6    23
```

Array Subtraction: similar to addition

Array Addition / Subtraction

Be aware:

Matrix addition and subtraction are **IDENTICAL** to element-wise addition and subtraction;

However…

Matrix multiplication and division are **NOT THE SAME** as element-wise multiplication and division.

Vector-Matrix Multiplication

The product $\boldsymbol{Ax}$ is a column vector with m rows if $\boldsymbol{A}$ is of size $m \times p$, and x is a p-element column vector

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{bmatrix}$$

E.g., $w = \boldsymbol{A} * x$ gives

$$\underbrace{\begin{bmatrix} 2 & 5 \\ 6 & 9 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} 3 \\ 4 \end{bmatrix}}_{x} = \begin{bmatrix} 2(3)+5(4) \\ 6(3)+9(4) \end{bmatrix} = \underbrace{\begin{bmatrix} 26 \\ 54 \end{bmatrix}}_{\boldsymbol{w}}$$

```
>> A = [2 5;6 9];
>> x = [3;4];
>> w = A*x

w =

    26
    54
```

Vector-Matrix Multiplication

If A is an $m \times p$ matrix, and $\boldsymbol{B}$ is a $p \times n$ matrix, the product $\boldsymbol{AB}$ is of size $m \times n$.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}=\begin{bmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} & a_{11}b_{13}+a_{12}b_{23} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} & a_{21}b_{13}+a_{22}b_{23} \end{bmatrix}$$

E.g., $\boldsymbol{C} = \boldsymbol{A} * \boldsymbol{B}$ gives

$$\underbrace{\begin{bmatrix} 2 & 5 \\ 6 & 9 \end{bmatrix}}_{\boldsymbol{A}}\underbrace{\begin{bmatrix} 6 & 10 & 4 \\ -2 & 3 & 7 \end{bmatrix}}_{\boldsymbol{B}}=\begin{bmatrix} 2(6)+5(-2) & 2(10)+5(3) & 2(4)+5(7) \\ 6(6)+9(-2) & 6(10)+9(3) & 6(4)+9(7) \end{bmatrix}$$

$$=\underbrace{\begin{bmatrix} 2 & 35 & 43 \\ 18 & 87 & 87 \end{bmatrix}}_{\boldsymbol{C}}$$

Matrix Exponentiation

- $A^2 = AA$: A must be square. To find A^2, type $A\hat{}2$
- This must be distinguished from the array exponentiation $A.\hat{}2$

```
>> A = [2 5;6 9]

A =

     2     5
     6     9

>> A^2

ans =

    34    55
    66   111

>> A.^2

ans =

     4    25
    36    81
```

Special Matrix Creating

- Identity Matrix: A square matrix whose diagonal elements are all ones, and the remaining elements are zeros. E.g., a 2×2 identity matrix is

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Property of the identity matrix: $\boldsymbol{I}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{I} = \boldsymbol{A}$
- To create an $n \times n$ identity matrix, type eye(n).

Special Matrix Creating

- All-zero matrix:
  - zeros(n) creates an $n \times n$ matrix of zeros.
  - zeros(m, n) creates an $m \times n$ matrix of zeros.
- All-one matrix:
  - ones(n) creates an $n \times n$ matrix of zeros.
  - ones(m, n) creates an $m \times n$ matrix of zeros.

Break?

Relational Operators

Relational operators: Operators to make comparison between scalars or arrays.

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

Relational Operators

- The result of a comparison using the relational operators is 1 (ture) or 0 (false).

- Example: Given $x = 2, y = 5$,
  - $z = (x < y),\ 1$
  - $z = (x <= y),\ 1$
  - $z = (x == y),\ 0$
  - $z = (x \sim= y),\ 1$
  - $z = (x >= y),\ 0$
  - $z = (x >= 2),\ 1$

== / =

- The equivalence operator $(==)$ is used for comparison and returns a logical result.
  E.g., $>> 3 == 4$, the answer is 0
- The assignment operator $(=)$ is used to define a variable/function.
  E.g., $>> a = \exp(-5)$, the answer is 0.0067

Relational Operators

The relational operators compare the arrays on an element-by-element basis.

```
>> x=[6 3 9]; y=[14 2 9];
>> z=(x<y)
z =
     1     0     0
>> z=(x~=y)
z =
     1     1     0
>> z=(x>8)
z =
     0     0     1
```

Logical Operators

Operator		Name	Definition
Binary	&	AND	A&B returns an array that has ones where both A and B have nonzero elements and zeros otherwise.
	|	OR	A|B returns an array that has ones where at least one element in A or B is nonzero and zeros otherwise.
	xor	Exclusive OR	xor(A,B) returns an array that has zeros where A and B are either both nonzero or both zero, and ones where either A or B is nonzero, but not both.
Unary	~	NOT	~A returns an array that has ones where A is zero and zeros where A is nonzero.

Logical Operators: find

The find function finds indices of nonzero elements in an array.

```
>> x=[5 -3 0 0 8]; y=[2 4 0 5 7];
>> v=find(x>0)
v =
     1     5
>> z=find(x&y)
z =
     1     2     5
>> w=find(x==0&y==0)
w =
     3
```

Branch Statements

Branch statements: if, else and elseif.

```
if (x>0) && (y>0)
    z=log(x)-3*log(y)
end
```

```
if x>=0
    y=sqrt(x)
else
   disp ('Some of the elements of x are negative')
end
```

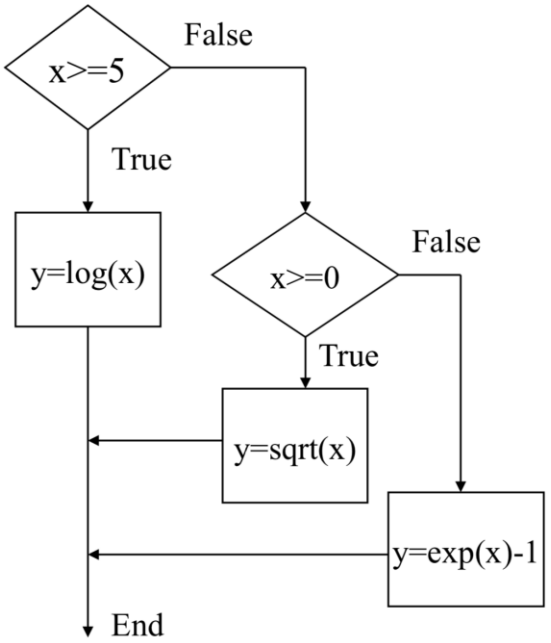
Branch Statements

E.g., Suppose that

$$y = \begin{cases} \ln x & x \geq 5 \\ \sqrt{x} & 0 \leq x < 5 \\ e^x - 1 & x < 0 \end{cases}$$

```
if x>=5
    y = log(x)
elseif x>=0
    y = sqrt(x)
else
    y = exp(x)-1
end
```

Flowchart:



Switch Case Statements

- To determine an integer x between 1 and 10 is odd or even,

```
function oddeven(x)
%oddeven determines an integer x between 1 and 10 is
% odd or even.
switch (x)
case {1, 3, 5, 7, 9}
   disp('The value is odd');
case {2, 4, 6, 8, 10}
   disp('The value is even');
otherwise,
   disp('The value is out of range');
end
```

Loops

- Loops are MATLAB constructs that permit us to execute a sequence of statements more than once.
  - for loops
  - while loops

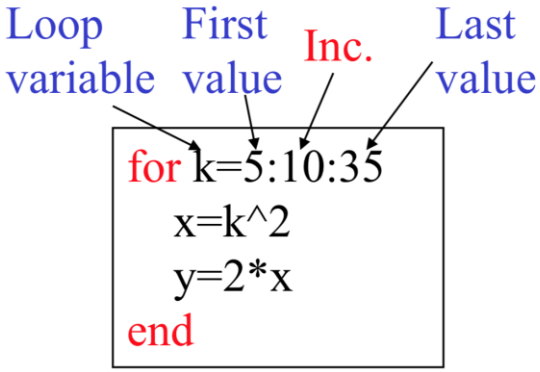
Loops: for

- The for loop executes a block of statements a specified number of times
- The for loop has the form

```
for loop variable=first: increment : last
        Statement 1
        ....
        Statement n
end
```

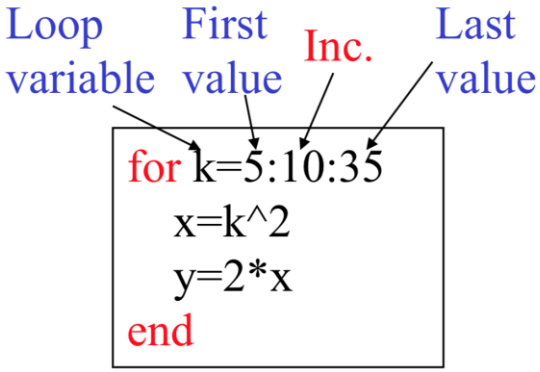
Control expression

Loops: for



- Each for statement must be matched by end.
- Loop variable:
  - Avoid using i or j (MATLAB built-in variables) as a loop variable.
  - Never modify a loop variable within the loop.

Loops: for



- Each for statement must be matched by end.
- Loop variable:
  - Avoid using i or j (MATLAB built-in variables) as a loop variable.
  - Never modify a loop variable within the loop.

Loops: for

```
function y=fun_factorial(n)
%fun_fraction: returns y=n!

%Initialising y for n=0 y=1;

for ii=1:n
        %n!=n*(n-1)!
        y=y*ii;
end
```

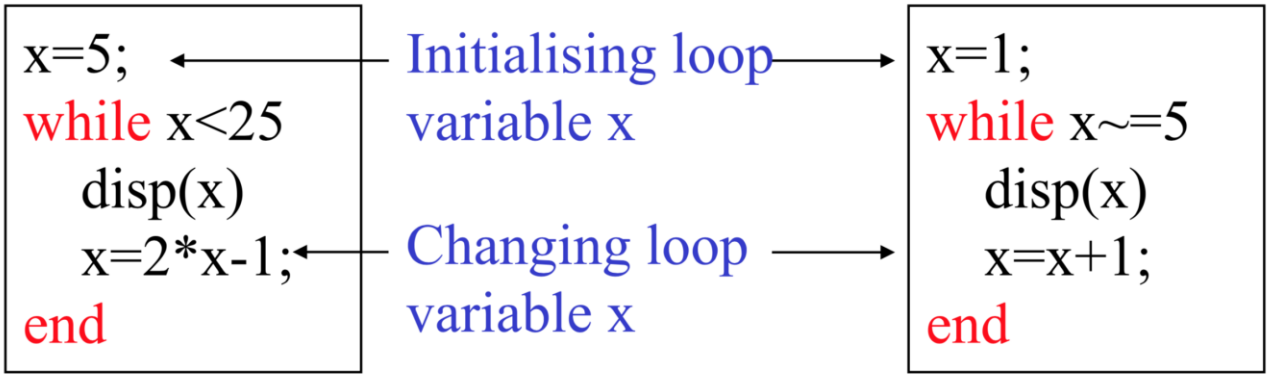
Loops: while

- The while loop repeats a block of statements indefinitely as long as some condition is satisfied.
- The while loop has the form

```
while expression
        Statement 1
        ....
        Statement n
end
```

If the *expression* is non-zero (true), the statements will be executed, until the *expression* becomes zero (false)

Loops: while



- Each while statement must be matched by end.
- Loop variable:
  - Must be initialized before the while loop.
  - Must be changed somehow by the statements.

Thanks for your attention.