SEEE1022 INTRODUCTION TO SCIENTIFIC PROGRAMMING

UTM UNIVERSITI TEKNOLOGI MALAYSIA

CH4
Control Flow

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

OBJECTIVES

1. To be able to write a decision algorithm into a program using if-elseif and switch-case statements.
2. To know the difference between the if-elseif and switch-case statements and which situation is best for each of the statement.
3. To be able to convert the if-elseif statement into logical vector decision method wherever possible.
4. To be able to write a loop algorithm into a program using for and while statements.
5. To know the difference between the for and while statements and which situation is best for each of the statement.
6. To be able to convert the for statement into array operations wherever possible.

OPERATION (RECAP)

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

CONTROL FLOW

INTRODUCTION

Control Flow

Conditional

- Decision statement.
- Tells the computer to pick and run one of several sets of statement based on certain condition.

Loop Control

- Repetition statement.
- Tells the computer to repeat certain set of statement for N number of repetition or while some condition is fulfil.
- Other than the above, repetition control such as skip and early exit from the repetition are also possible.

MATLAB SCRIPT

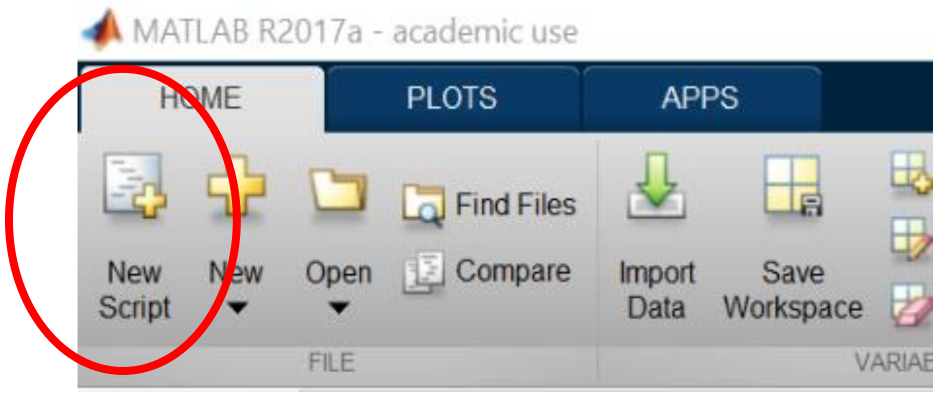
INTRODUCTION

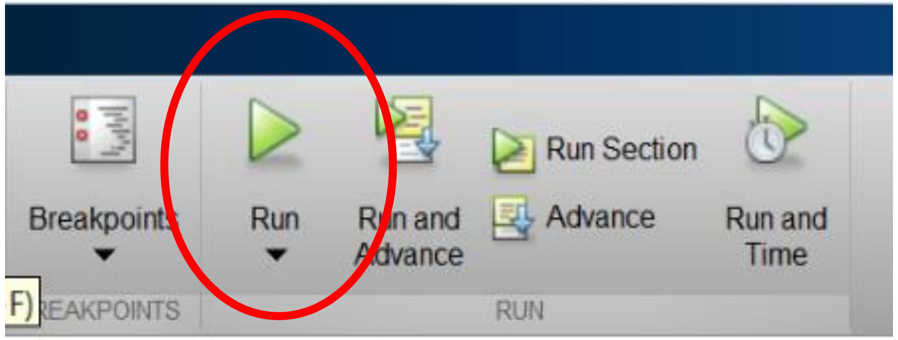
- Scripts are the simplest type of program file.
- Scripts have no input or output arguments.
- They are useful for automating a series of MATLAB commands, such as computations that you have to perform repeatedly from the command line or series of commands you have to reference.
- In this chapter, all examples are written as MATLAB script since we now need a series of commands to work with the control flow.

MATLAB SCRIPT

CREATE AND RUN A SCRIPT

- Clicking the New Script icon will open a new script editor.
- Write the code inside the editor and save it as .m file.
- Then run the code by either typing the file name on the command window or click the Run icon on the editor window.





CONDITIONAL

CONDITIONAL INTRODUCTION

if-elseif-else	switch-case-otherwise
EXECUTE STATEMENTS IF CONDITION IS TRUE	EXECUTES STATEMENTS BASED ON THE VALUE OF A VARIABLE

```
if condition_1
    statements
elseif condition_2
    statements
 ...
else
    statements
End
```

- Condition is normally express as Boolean expression
- If condition is express as arithmetic expression, condition is true when the value is nonzero

```
switch n
   case value_1
      statements
   case value_2
      statements
   ...
   otherwise
      statements
end
```

- Variable n is normally express as arithmetic expression
- Case value must be a scalar or string.
- To compare against multiple value, use cell array

CONDITIONAL

SINGLE IF-ELSE

EXAMPLE 1

```
x = 2;
if x<0
    disp('neg')
else
    disp('non-neg')
end
```

This code will display 'non-neg' at the command window. Try change the value as `x=-2`

EXAMPLE 2

```
if 79
    disp('true')
else
    disp('false')
end
```

This code will display 'true' at the command window. Try other values, including zero & negative values.

CONDITIONAL

MULTIPLE CONDITIONS WITH SINGLE IF

EXAMPLE 3

For quadratic equation of $ax^2 + bx + c = 0$, it has equal roots, given by $-b/(2a)$ provided that $b^2 - 4ac = 0$ and $a \neq 0$.

Below is the MATLAB code to compute the equal roots

```
a = input('Input a value: ');
b = input('Input b value: ');
c = input('Input c value: ');

d = b^2-4*a*c;

if (d==0) && (a~=0)
    x = -b/(2*a)
else
    disp('different root')
end
```

This code use logical operator `&&` to set a multiple conditions for when to compute the equal roots `x`.

CONDITIONAL

MULTIPLE CONDITIONS USING ELSEIF

EXAMPLE 4

Other than the equal roots, there are two other types of roots for the quadratic equation provided that $a \neq 0$: the nonequal roots and complex roots.

Below is the MATLAB code to compute only for the real roots.

```
d = b^2-4*a*c;

if (d==0) && (a~=0)
    x1 = -b/(2*a);
    x2 = x1;
elseif (d>0) && (a~=0)
    x1 = (-b+sqrt(d))/(2*a);
    x2 = (-b-sqrt(d))/(2* a);
else
    disp('complex root')
end
```

This code use `elseif` since there are more than 2 conditions.

CONDITIONAL

NESTED IF

EXAMPLE 5

To complete the problem, nested ifs is use where the first if is dedicated to variable a while the second if is for the variable d. Thus, now the root when $a = 0$ is also being computed.

```
d = b^2-4*a*c;
if a~=0
    if (d==0)
        x1 = -b/(2*a);
        x2 = x1;
    elseif (d>0)
        x1 = (-b+sqrt(d))/(2*a);
        x2 = (-b-sqrt(d))/(2* a);
    else
        disp('complex root')
    end
else
    x1 = -c/b;
end
```

When `a=0`, the quadratic equation becomes linear equation. Thus, only one root available and computed as `x1` in the outer `else`.

CONDITIONAL

SWITCH - CASE

elseif : great for variable conditions that result into a Boolean.

switch : great for fixed data values.

EXAMPLE 6

```
n = input('Enter a number: ');
switch n
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```

Every `case` is referring to a given fixed value. In this example, `-1`, `0` and `1`

```
Enter a number: 1
positive one
```

CONDITIONAL

DESIGN USING LOGICAL

- Instead of if and switch statements, logical approach is another method to execute decision statement.
- It is useful in avoiding the if and switch statements when involving mathematical equations, which is common in scientific programming.

ADVANTAGE

- They are almost always faster than if and switch methods.
- Can be easier to read since the expression can be written very close to the mathematical equation form.

- In logical approach, condition in if and switch statement is replaced with logical multiplier array.

CONDITIONAL DECISION USING LOGICAL

EXAMPLE 7

A simple version of how income tax is calculated could be based on the following table:

Taxable Income	Tax Payable
$10000 or less	10% of taxable income
Between $10000 and $20000	$1000 + 20% of amount by which taxable income exceeds $10,000
More than $20000	$3000 + 50% of amount by which taxable income exceeds $20,000

The tax payable on a taxable income of $30000, for example, is:

Tax = 3000 + 0.5*(30000-20000) = 8000

CONDITIONAL

DECISION USING LOGICAL VECTOR

Below is how the tax payable calculation is solve using elseif.

```
inc = input('Input an income: ');

if inc <= 10000
    tax = 0.1*inc;
elseif inc <= 20000
    tax = 1000 + 0.2*(inc-10000);
else
    tax = 3000 + 0.5*(inc-20000);
end

disp(['Tax Payable = ', num2str(tax)])
```

```
Input an income: 30000
Tax Payable = 8000
```

CONDITIONAL

DECISION USING LOGICAL VECTOR

- To convert the elseif method to logical method, the conditions are multiply with their respective formulas. Then, the results of all condition multiplication are added to get the final answer.
- Below is the coding structure:

```
var_1 = (formula_1)*(condition_1)
var_2 = (formula_2)*(condition_2)


var_n = (formula_n)*(condition_n)

var = var_1 + var_2 + … + var_n
```

HOW IT WORK

- Only one condition will be true at one time. Thus, the addition at the last line will result the output of the formula with the true condition.

CONDITIONAL

DECISION USING LOGICAL VECTOR

EXAMPLE 8

Below is the mathematical equation of the tax payable calculation, followed by the MATLAB code using the logical vector method.

$$tax = \begin{cases} 0.1 \times inc & for\ inc \leq 10000 \\ 1000 + 0.2(inc - 10000) & for\ 10000 < inc \leq 20000 \\ 3000 + 0.5(inc - 20000) & for\ inc > 20000 \end{cases}$$

```
inc = input('Input an income: ');

tax1 = (0.1*inc)                  *(inc<=10000);
tax2 = (1000 + 0.2*(inc-10000)) *(inc>10000 & inc<=20000);
tax3 = (3000 + 0.5*(inc-20000)) *(inc>20000);

tax = tax1 + tax2 + tax3;

disp(['Tax Payable = ', num2str(tax)])
```

```
Input an income: 30000
Tax Payable = 8000
```

LOOP CONTROL

LOOP CONTROL

FOR
DETERMINATE LOOPS

SYNTAX

```
for index=[array]
  Statements;
end
```

Repeat for every index value

```
j:m:k
j:k
[2 4 6 1]
[0.1 0.3; 2 10]
```

j – starting value
m – increment value
k – limit value

*or any other method of creating an array

WHILE
INDETERMINATE LOOPS

SYNTAX

```
while condition
  Statements;
end
```

Boolean expression

```
a>0
a<=100
a<100&a>10
a>0|b==10
```

Repeat while condition is true

Or value is not 0 when arithmetic expression is used (not recommended).

CONTINUE

PASS CONTROL TO NEXT ITERATION

BREAK

TERMINATE LOOPS EXECUTION

LOOP CONTROL

FOR LOOP

EXAMPLE 9

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

$$B = a(1 + r)^{n}$$

If B is to be evaluated for a=$100 on 5 different total year n (2,4,6,8,10) and interest rate of r=8%, below is how the for loop is use to compute all of the B values.

```
a=100;
r=0.08;

for n=2:2:10
    B=a*(1+r)^n;
    disp([n B])
end
```

WHILE LOOP

EXAMPLE 10

Below is how the same equation is coded with while loop:

```
a=100;
r=0.08;
n=2;
while n<=10
     B=a*(1+r)^n;
     disp([n B])
     n=n+2;
end
```

```
 2.0000  116.6400
 4.0000  136.0489
 6.0000  158.6874
 8.0000  185.0930
10.0000  215.8925
```

LOOP CONTROL

RETURNING VECTORS AS OUTPUT

EXAMPLE 11

When a resistor (R), capacitor (C) and battery (V) are connected in series, a charge Q builds up on the capacitor according to the formula:

$$Q(t) = CV\left(1 - e^{-\frac{t}{RC}}\right)$$

where t is the charging time starts at 0. The problem is to monitor the charge on the capacitor every 0.5 second in order to detect when it reaches a level of 2 units of charge, given that V=9, R=4 and C=1.

a) Write a program which display the time and charge every 0.5 seconds until the charge first exceeds 2 units (i.e. the last charge displayed must exceed 2).

Next slide shows how the problem is coded with while loop:

LOOP CONTROL

RETURNING VECTORS AS OUTPUT

EXAMPLE 11

The code for Example 11:

```
V=9;  R=4;  C=1;     %set V,R and C value
t=0;                 %initialise charge time
q=0;                 %initialise charge q at time 0
n=1;                 %initialise index for the output vectors
while q<=2
    q = C*V*(1-exp(-t/(R*C)));
    Q(n) = q;
    T(n) = t;

    t = t+0.5;
    n = n+1;
end
disp(['Q : ', num2str(Q)])
disp(['T : ', num2str(T)])
```

Q(1): 0	Q(2): 1.0575	Q(3): 1.9908	Q(4): 2.8144
T(1): 0	T(2): 0.5	T(3): 1	T(4): 1.5

- `n` is used for index instead of `t` since `t` is not integer.

NESTED LOOP

EXAMPLE 12

- Lets again consider a formula of compound interest. Now, B is to be evaluated for 3 values of a (\$100,\$500,\$800) on 5 different total year of n (2,4,6,8,10). This time, let r=0.09.
- Since now we also need to compute for several values of a, we need two loop statements.

```
r=0.08;
for a=[100 500 800]
    disp(['a= ',num2str(a)]);
    for n=2:2:10
        B=a*(1+r)^n;
        disp([n B])
    end
end
```

LOOP CONTROL

CONTINUE STATEMENT

Continue **passes control** to the **next iteration** and **skips remaining statements**. In nested loops, continue skips remaining statements only in the body of the loop in which it occurs.

EXAMPLE 13

```
for n = 1:50
    if mod(n,7)
        continue
    end

    disp(['Divisible by 7: ' num2str(n)])
end
```

```
Divisible by 7:  7
Divisible by 7:  14
Divisible by 7:  21
Divisible by 7:  28
Divisible by 7:  35
Divisible by 7:  42
Divisible by 7:  49
```

- The program skip displaying `n` when it is not divisible by seven.
- Check MATLAB documentation for function mod() and try to figure out how the `if` statement is performed.

LOOP CONTROL

BREAK STATEMENT

- Break **terminates the execution** of a **for** or **while** loop. Statements in the loop after the break statement do not execute.
- In nested loops, break exits only from the loop in which it occurs. Control passes to the statement that follows the end of that loop.

EXAMPLE 14

- Lets write a simple random number guessing game where you need to continuously guess one random number until your guessed number is correct.
- The random number can be generated using function randi(). In this example, to have a random number between 1 to 6, we write the function as randi(6).
- From the program on the next slide, observe that the endless loop is terminated with break statement.

LOOP CONTROL

BREAK STATEMENT

Below is the code for the guessing number game:

```
x = randi(6);
n = 0;
while 1
    guess = input('Guess the number: ');
    n = n+1;
    if guess==x
        disp(['"You got it right after ',num2str(n),' try"']);
        break
    end
end
```

```
Guess the number: 3
Guess the number: 2
Guess the number: 6
"You got it right after 3 try"
```

- `while 1` will execute endless loop.

LOOP CONTROL

VECTORIZING FOR LOOP

Given the way MATLAB has been designed, for loops tend to be inefficient in terms of computing time.

EXAMPLE 15

- Lets recap Example 12 where a formula of compound interest B is evaluated for 3 values of a (\$100,\$500,\$800) on 5 different total year of n (2,4,6,8,10) and r=0.09.
- By vectorizing a, next slide shows how one of the for loop can be removed to compute all of the B values.
- At the output matrix, row is dedicated to a and column is dedicated to n, representing the output such in a table of a versus n.

LOOP CONTROL

VECTORIZING FOR LOOP

The code for Example 15

```
a=[100 500 800];
r=0.09;

for n=2:2:10
    B(1:3,n/2)=a*(1+r)^n;
end
disp(B)
```

```
1.0e+03 *
      n=2       n=4       n=6       n=8       n=10
    0.1188    0.1412    0.1677    0.1993    0.2367
    0.5941    0.7058    0.8386    0.9963    1.1837
    0.9505    1.1293    1.3417    1.5941    1.8939
```

LOOP CONTROL

AVOID FOR LOOP BY ARRAY OPERATION

Although for loop is essential for many cases, there are cases especially in scientific programming where it is possible to convert all the indices into array and use array operation to replace the for loop.

EXAMPLE 16

- As from the previous Example 13, the remaining for loop can be removed by extending the vectorization method as shown in the next slide.
- The code use meshgrid function to duplicate vector a and n into two-dimensional arrays of A and N. Then, the arrays are entered to the equation to obtain the two-dimensional array of B.
- Recap:

  Array operation must be performed using array operator, i.e. with symbol period (.)

LOOP CONTROL

AVOID FOR LOOP BY ARRAY OPERATION

Below is the code for Example 16

```
a=[100 500 800];
n=[2 4 6 8 10];
r=0.09;
[N,A] = meshgrid(n,a)
B = A.*(1+r).^N
```

```
N =
     2     4     6     8    10
     2     4     6     8    10
     2     4     6     8    10

A =
   100   100   100   100   100
   500   500   500   500   500
   800   800   800   800   800

B =
   1.0e+03 *

    0.1188    0.1412    0.1677    0.1993    0.2367
    0.5941    0.7058    0.8386    0.9963    1.1837
    0.9505    1.1293    1.3417    1.5941    1.8939
```

LOOP CONTROL

LOOP STATEMENT VS ARRAY OPERATION

EXAMPLE 17

```
a = randi(1000,1,10000);
n = 2:2:20000;
r=0.09;

tic
[N,A] = meshgrid(n,a);
B = A.*(1+r).^N;
time1 = toc;

tic
for i=1:length(a)
    for j=1:length(n)
        B(i,j) = a(i)*(1+r)^n(j);
    end
end
time2 = toc;

disp(['Array Operation Elapse Time : ', num2str(time1)])
disp(['           Loop Elapse Time : ', num2str(time2)])
```

```
Array Operation Elapse Time : 7.703
           Loop Elapse Time : 15.3738
```

- Based on the resulting elapse time, it shows that the loop statement for this program consume almost double the time compared to the array operation method.

AVOIDING LOOP & DECISION STATEMENTS

EXAMPLE 18

Back to Example 6, lets write a program to calculate the tax for several values of income, e.g. [4000 12000 18000 23000 30000] by avoiding both the loop and decision statements.

```
inc = [4000 12000 18000 23000 30000];

tax1 = (0.1*inc)                 .*(inc<=10000);
tax2 = (1000 + 0.2*(inc-10000)) .*(inc>10000 & inc<=20000);
tax3 = (3000 + 0.5*(inc-20000)) .*(inc>20000);

tax = tax1 + tax2 + tax3;

disp(['Income : ', num2str(inc)])
disp(['   Tax : ', num2str(tax)])
```

```
Income : 4000      12000      18000      23000      30000
   Tax :  400       1400       2600       4500       8000
```

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

univteknologimalaysia

utm_my

utmofficial

Thank You

www.utm.my

innovative • entrepreneurial • global