ENGR 1100

Week 05- Programming in MATLAB

Learning Objectives

Upon completion this module, students will be able to:

1. Demonstrate how to use relational and logical operators in MATLAB,
2. Apply conditional statement in MATLAB programming
3. Implement different types of Loops in MATLAB programming
   - For Loops
   - While Loops
   - Double Loops

Programming in MATLAB

In this module, we will study how to make MATLAB programs run sections of code

- If something is true
- While something is true
- For a certain number of times

We will also learn how to run different sections of code depending on

- The value of a variable
- Which particular condition is true
- What combination of conditions is true
  - If this and that are true
  - If this or that is true, etc.
- What relationship two things have
  - For example, one is less than the other; greater than; equal to; not equal to; etc.

RELATIONAL AND LOGICAL OPERATORS

Relational Operations

MATLAB: Relational functions

MATLAB: Array Comparison with Relational Operators

MATLAB: Operator Precedence

Relational operator

Relational operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not Equal to

- Can't put space between operators that have two characters
- "Not equal to" is "~=", not "!=" as in C or C++
- "Equal to" comparison is two equal signs (==), not one.
  - Remember, "=" means "assign to" or "put into"

Relational operator (Cont’d)

- Result of comparing with a relational operator is always "true" or "false"
  - If "true", MATLAB gives the comparison a value of one (1)
  - If "false", MATLAB gives the comparison a value of zero (0)

Remark: This may be different than convention in other programming languages. For example, C gives an expression that is false a value of zero, but it can give a true expression any value but zero, which you can't assume will be one

Example

Try these in MATLAB

```
>> 5>8
>> x=5<8
>> a= 1:9;
>> b= 8-a;
>> a==b
>> a~=b
>> a>b
```

When comparing arrays

- They must be the same dimensions
- MATLAB does an elementwise comparison
- Result is an array that has same dimensions as other two but only contains 1's and 0's

Examples –Cont'd

Example 1: How many of the numbers from 1-20 are prime?

- Use MATLAB `isprime` command, which returns true (1) is number is prime and false (0) if it isn't

```
>> numbers = 1:20;
>> sum( isprime(numbers) )
ans =
     8
```

Example 2: What are the prime numbers from 1-20?

```
>> numbers = 1:20;
>> numbers( isprime(numbers) )
ans =
     2   3   5   7   11   13   17   19
```

Remark: Logical indexing is very useful.

Logical Operator

MATLAB has three common logical operators: `&`, `|`, `~`

- `a&b` does the logical AND operation on `a` and `b`
- `a|b` does the logical OR operation on `a` or `b`
- `~a` does the logical NOT operation on `a`
- Arguments to all logical operators are numbers
  - Zero is "false"
  - Any non-zero number is "true"
- Result (output) of logical operator is a logical one (true) or zero (false)

Logical Operator- Cont’d

When using logical operator on arrays

- The arrays must be the same dimensions
- MATLAB does an element-wise evaluation of operator
- Result is an array that has same dimensions as other two but only contains 1's and 0's

Example

Define:

- Child – 12 or less years
- Teenager – more than 12 and less than 20 years
- Adult – 20 or more years

Who is a teenager?

```
>> age=[45 47 15 13 11]
age =  45    47    15    13    11
>> age>=13
ans =     1     1     1     1     0
>> age<=19
ans =     0     0     1     1     1
>> age>=13 & age<=19
ans =     0     0     1     1     0
```

These mark the two teenagers

Example- Cont'd

Who is not a teenager?

```
>> ~(age>=13 & age<=19)
ans =    1     1     0     0     1
```

Who is an adult or a child?

```
>> age>19 | age<13
ans =    1     1     0     0     1
```

Built-in Logical Functions

MATLAB has some built-in functions or commands for doing logical operations and related calculations. Three are equivalent to the logical operators

- `and(A,B)` – same as `A&B`
- `or(A,B)` – same as `A|B`
- `not(A)` – same as `~A`

MATLAB also has other Boolean functions

Function	Description	Example

`xor(a,b)`	Exclusive or. Returns true (1) if one operand is true and the other is false.	`>> xor(7,0)`
		`ans =`
		`1`
		`>> xor(7,-5)`
		`ans =`
		`0`

`all(A)`	Returns 1 (true) if all elements in a vector A are true (nonzero). Returns 0 (false) if one or more elements are false (zero).
	If A is a matrix, treats columns of A as vectors, and returns a vector with 1s and 0s.
		`>> A=[6 2 15 9 7 11];`
		`>> all(A)`
		`ans =`
		`1`
		`>> B=[6 2 15 9 0 11];`
		`>> all(B)`
		`ans =`
		`0`

`any(A)`	Returns 1 (true) if any element in a vector A is true (nonzero). Returns 0 (false) if all elements are false (zero).
	If A is a matrix, treats columns of A as vectors, and returns a vector with 1s and 0s.
		`>> A=[6 0 15 0 0 11];`
		`>> any(A)`
		`ans =`
		`1`
		`>> B = [0 0 0 0 0 0];`
		`>> any(B)`
		`ans =`
		`0`

`find(A)`
 	
`find(A>d)`
	If A is a vector, returns the indices of the nonzero elements.
	If A is a vector, returns the address of the elements that are larger than d (any relational operator can be used).
		`>> A=[0 9 4 3 7 0 0 1 8];`
		`>> find(A)`
		`ans =`
		`2 3 4 5 8 9`
		`>> find(A>4)`
		`ans =`
		`2 5 9`

CONDITIONAL STATEMENT

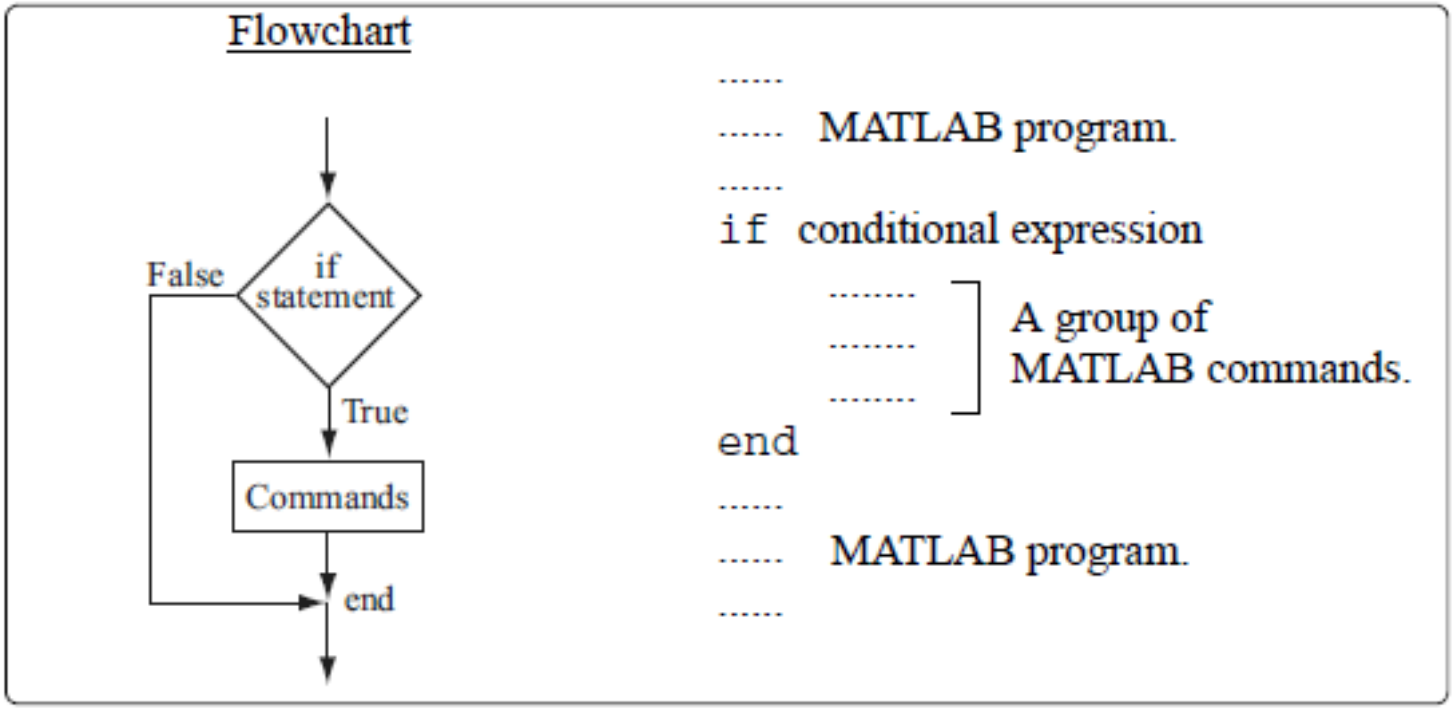
Conditional Statement

MATLAB: Conditional Statement

A ***conditional statement*** is a command that allows MATLAB to decide whether or not to execute some code that follows the statement

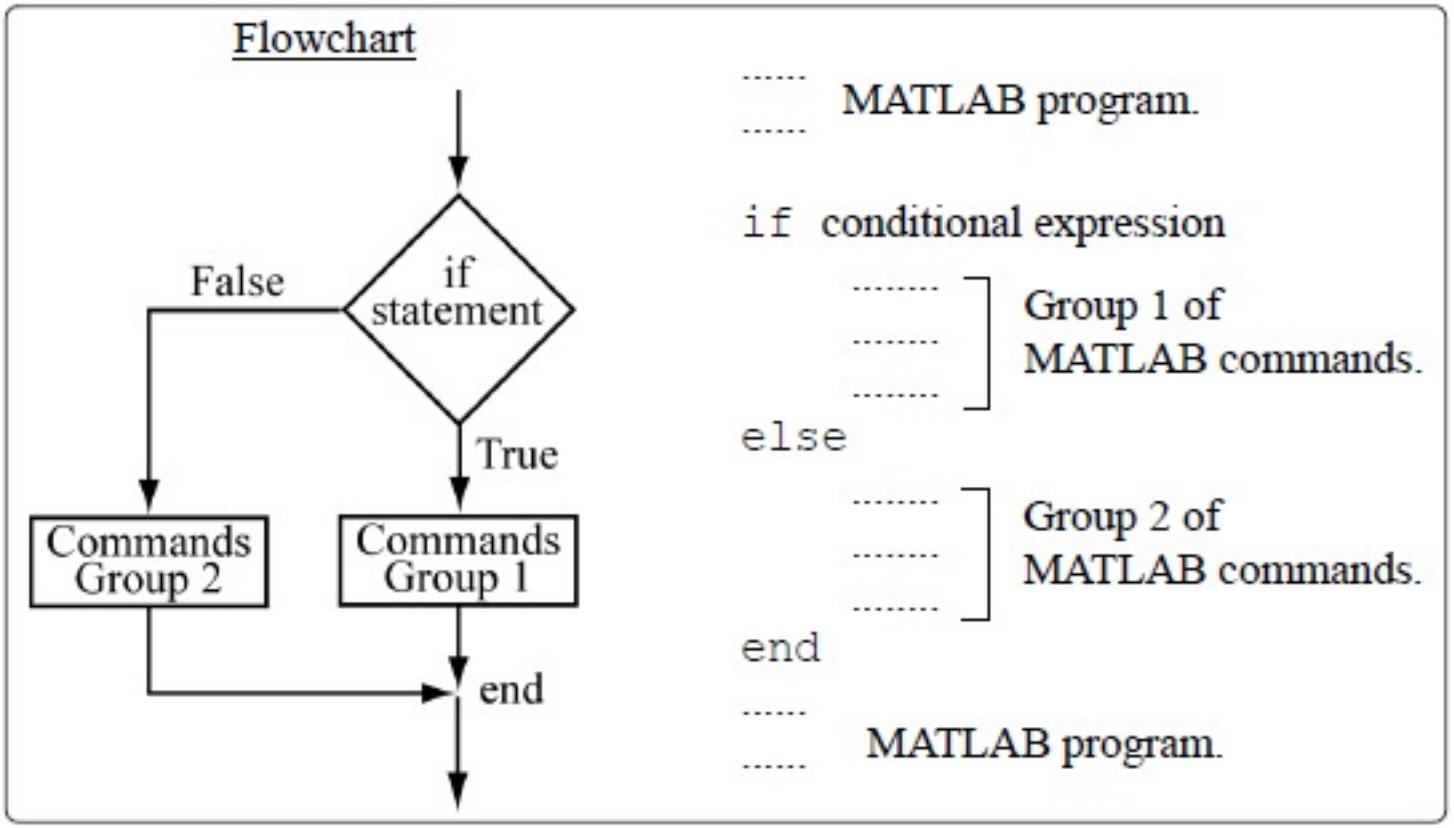
- Conditional statements almost always part of scripts or functions
- They have three general forms
  - `if-end`
  - `if-else-end`
  - `if-elseif-else-end`

if-end statement



The structure of the `if-end` conditional statement.

if-else-end



The structure of the `if-else-end` conditional statement.

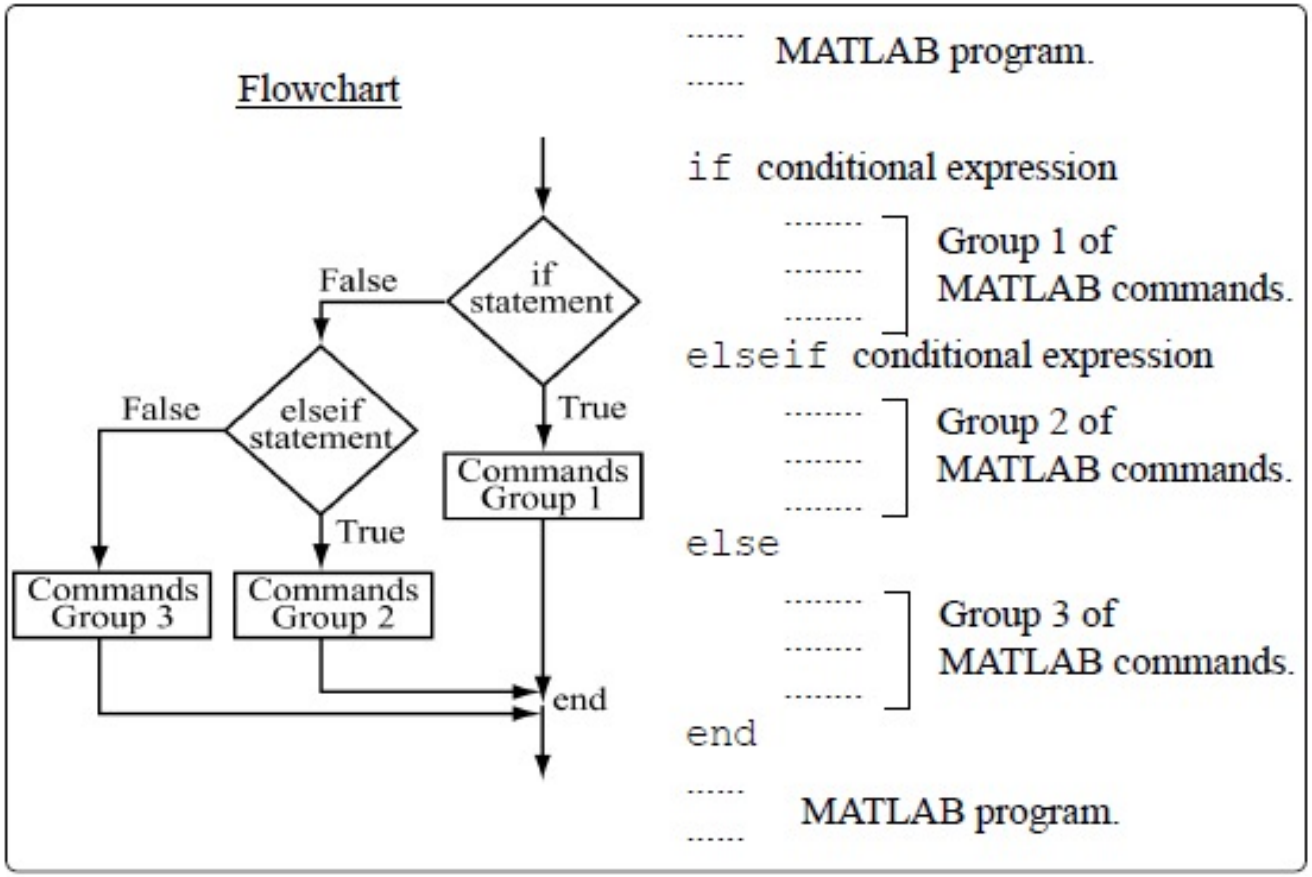
Example

`if-else-end` structure lets you execute one section of code if a condition is true and a different section of code if it is false.

EXAMPLE - answering your phone

```
if the caller is your best friend
    talk for a long time
else
    talk for a short time
end
```

if-elseif-else-end



The structure of the `if-elseif-else-end` conditional statement.

Example

Can have as many `elseif` statements as you want

EXAMPLE

```
if the caller is your best friend
   talk for a long time
elseif the caller is a potential date
   talk for a little bit and then set a time to meet
elseif the caller is your study-mate
   talk until you get the answer to the hard problem
elseif the caller is your mom
   say you're busy and can't talk
else
   have your room-mate say you'll call back later
end
```

Switch-case

`if-elseif-else-end` structure gets hard to read if more than a few `elseif` statements. A clearer alternative is the `switch-case` structure

- `switch-case` slightly different because choose code to execute based on value of scalar or string, not just true/false

Switch-case

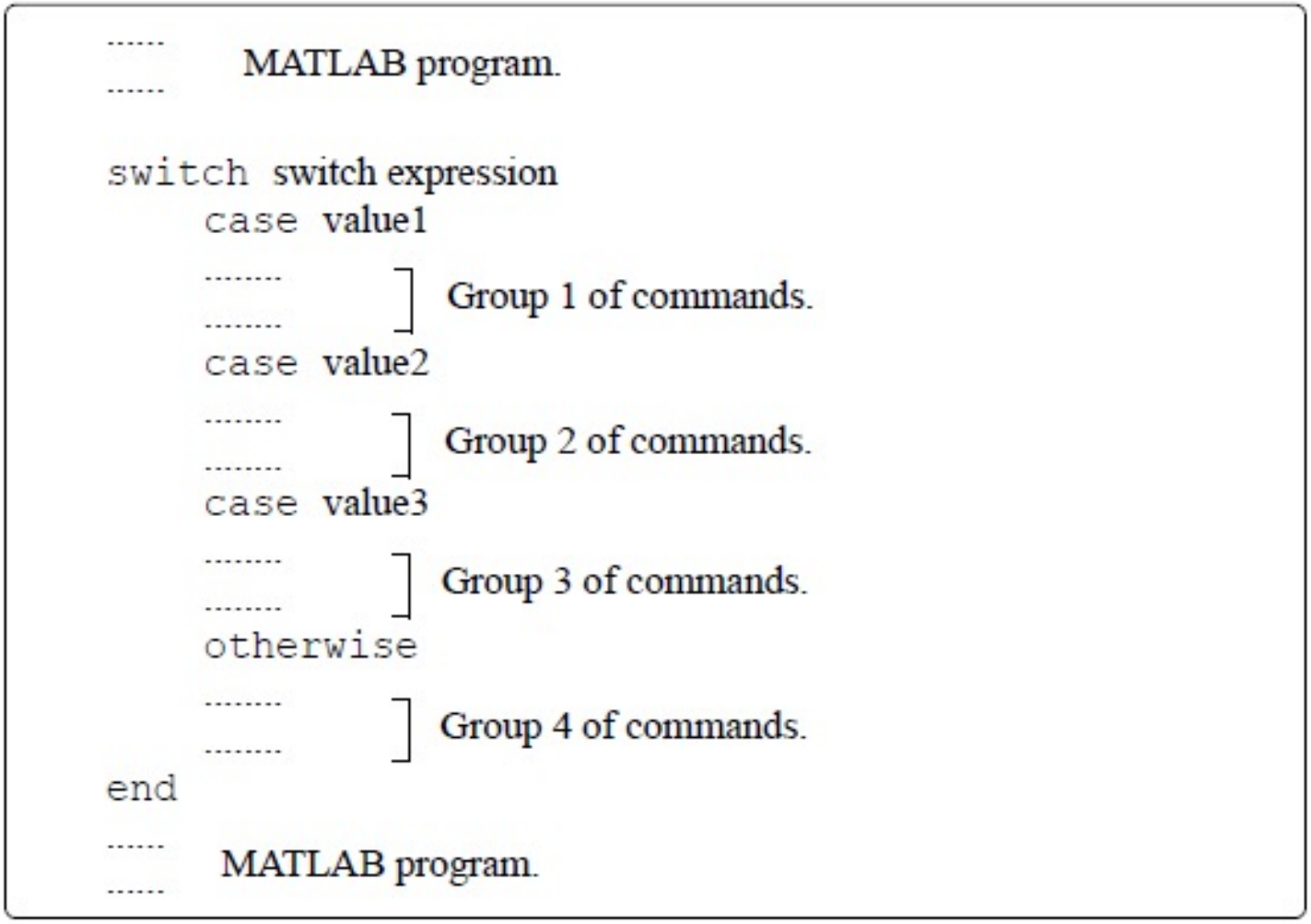


Figure 6-4: The structure of a `switch-case` statement.

Example

```
switch name
case 'Bobby'
  talk for a long time
case 'Susan'
  talk for a little bit and then set a time to meet
case 'Hubert'
  talk until you get the answer to the hard problem
case 'Mom'
  say you're busy and can't talk
otherwise
  have your room-mate say you'll call back later
end
```

LOOPS

MATLAB Loops

MATLAB: Loop Control Statement

MATLAB: For Loops

MATLAB: While Loops

The For Loops

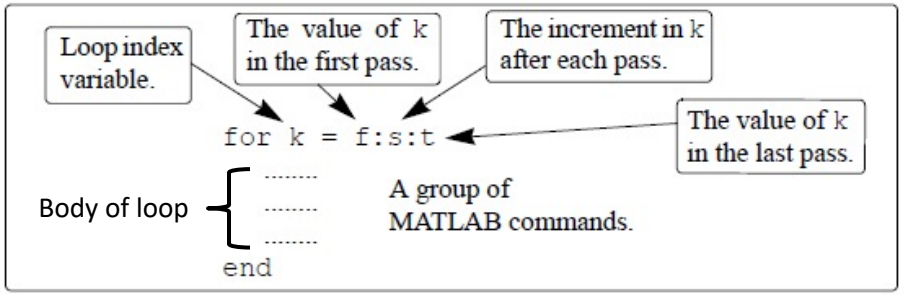


Figure 6-5: The structure of a `for-end` loop.

1. Loop sets `k` to `f`, and executes commands between `for` and the `end` commands, i.e., executes body of loop
2. Loop sets `k` to `f+s`, executes body
3. Process repeats itself until `k > t`
4. Program then continues with commands that follow `end` command

- `f` and `t` are usually integers
- `s` usually omitted. If so, loop uses increment of 1

The For Loops

- Increment `s` can be negative
  - For example, `k = 25:-5:10` produces four passes with `k = 25, 20, 15, 10`
- If `f = t`, loop executes once
- If `f > t` and `s > 0`, or if `f < t` and `s < 0`, loop not executed
- In the `for` command `k` can also be assigned specific value (typed in as a vector)
  - For example: `for k = [7 9 -1 3 3 5]`
- In general, loop body should not change value of `k`
- Each `for` command in a program must have an `end` command

The While Loop

`while-end` loop used when

- You don't know number of loop iterations
- You do have a condition that you can test and stop looping when it is false. For example,
  - Keep reading data from a file until you reach the end of the file
  - Keep adding terms to a sum until the difference of the last two terms is less than a certain amount

```
while conditional expression
   ........
   ........          A group of
   ........          MATLAB commands.
end
```

Figure 6-6: The structure of a `while-end` loop.

1. Loop evaluates conditional-expression
2. If conditional-expression is true, executes code in body, then goes back to Step 1
3. If conditional-expression is false, skips code in body and goes to code after `end`-statement

The While Loop

The conditional expression of a `while-end` loop

- Has a variable in it
  - Body of loop must change value of variable
  - There must be some value of the variable that makes the conditional expression be false

```
while condition
   statements
end
```

MATLAB Example 3

```
x = 1
while x <= 15
    x = 2*x
end
```

Makes this output

```
x =
     1
x =
     2
x =
     4
x =
     8
x =
    16
```

MATLAB Example 4

A worker is paid according to his hourly wage up to 40 hours, and 50% more for overtime. Write a program in a script file that calculates the pay to a worker. The program asks the user to enter the number of hours and the hourly wage. The program the displays the pay.
[Use if-end statement]

The For Loops

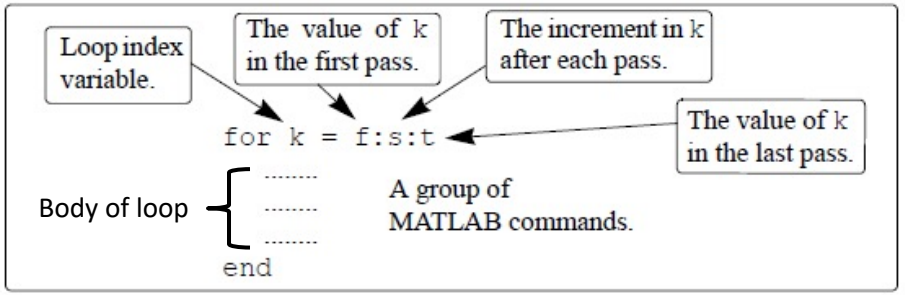


Figure 6-5: The structure of a `for-end` loop.

1. Loop sets `k` to `f`, and executes commands between `for` and the `end` commands, i.e., executes body of loop
2. Loop sets `k` to `f+s`, executes body
3. Process repeats itself until `k > t`
4. Program then continues with commands that follow `end` command

- `f` and `t` are usually integers
- `s` usually omitted. If so, loop uses increment of 1

MATLAB Example 5

```
for k=1:3:10
   k
   x = k^2
end
fprintf('After loop k = %d\n', k);
```

Output

```
k =    1
x =    1
k =    4
x =   16
k =    7
x =   49
k =   10
x =  100
After loop k = 10
```

MATLAB Example 5 –Cont'd

```
>> for j=1:4
     y(j)=j^2;
   end
```

What is the value of y? Try it yourself without using MATLAB.

The While Loops

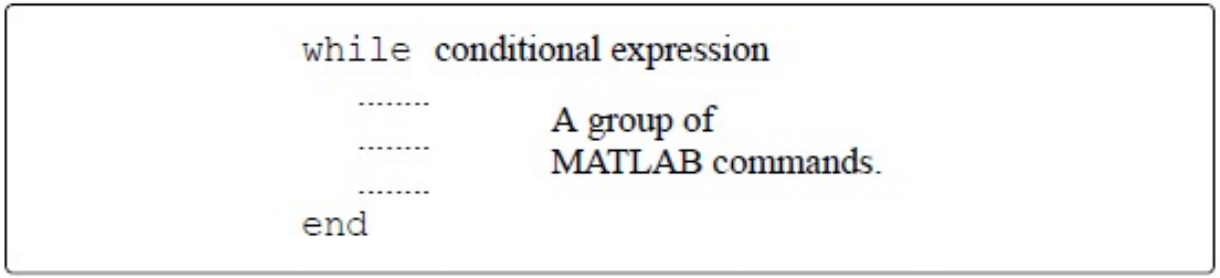


Figure 6-6: The structure of a `while-end` loop.

1. Loop evaluates conditional-expression
2. If conditional-expression is true, executes code in body, then goes back to Step 1
3. If conditional-expression is false, skips code in body and goes to code after `end`-statement

MATLAB Example 6

The following code ask a user to enter a value between 1 and 10, and if the number is not within this limits, the user is informed and he can re-enter the correct value. The loops continues till it get a correct input value.

```
value=input('Please Enter a value between 1 and 10');
while (value<1 || value >10)
  fprintf('Incorrect input, please try again. \n');
  value=input('Enter a Number between 1 and 10');
end
```

The While Loop

If the conditional expression never becomes false, the loop will keep executing... forever! We call this an ***indefinite loop***, but more commonly referred to as an ***infinite loop***. Your program will just keep running, and if there is no output from the loop (as if often the case), it will look like MATLAB has stopped responding.

What are some common causes of infinite loop?

Common causes of infinite loops -1

- No variable in conditional expression

```
distance1 = 1;
distance2 = 10;
distance3 = 0;
while distance1 < distance2
   fprintf('Distance = %d\n',distance3);
end
```

distance1 and distance2 never change

Common causes of infinite loops -2

- Variable in conditional expression never changes

```
minDistance = 42;
distanceIncrement = 0;
distance = 0;
while distance < minDistance
   distance=distance+distanceIncrement;
end
```

Typo – should be 10

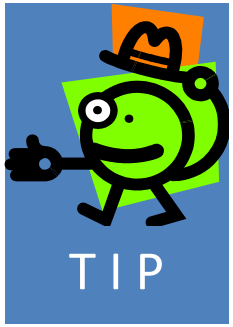
Common causes of infinite loops -3

- Conditional expression never becomes false

```
minDistance = 42;
x = 0;
y = 0;
while -sqrt( x^2+y^2 ) < minDistance
   x = x + 1;
   y = y + x;
end
```

Typo – shouldn't be any negative sign

How to stop infinite loop?



If your program gets caught in an infinite loop,

- Put the cursor in the Command Window
- Press CTRL+C

Double Loops

Double loops are mainly used to create matrices or perform operations between matrices. The simple form of double loops is:

```
for icounter= 1: n
    Statements
        for jcounter=1:m
            Statements
        end
end
```

Example

```
for k=1:4
    for h=1:3
        A(k,h)=k-h;
    end
end
```

MATLAB Example 7

Use double loops to generate the identity matrix I.

- First, ask user to specify the size of identity matrix
- Use double loops and if-then statement to generate 1 only at diagonal positions, 0 at all other positions.
- Display the identity matrix

MATLAB Example 8

Write a program in a script file that creates a $n \times m$ matrix with elements that have the following values.

- The value of each element in the first row is the number of the column.
- The value of each element in the first column is the number of the row.
- The rest of the elements each has a value equal to the sum of the element above it and the element to the left.

When executed, the program asks the user to enter values for n and m.

Nested Loops & Nested Conditional Statements

```
n=input('Enter the number of rows ');
m=input('Enter the number of columns ');
A=[];                                   Define an empty matrix A
for k=1:n                               Start of the first for-end loop.
   for h=1:m                            Start of the second for-end loop.
      if k==1                           Start of the conditional statement.
          A(k,h)=h;                     Assign values to the elements of the first row.
      elseif h==1
          A(k,h)=k;                     Assign values to the elements of the first column.
      else
          A(k,h)=A(k,h-1)+A(k-1,h);     Assign values to other elements.
      end                               end of the if statement.
   end                                  end of the nested for-end loop.
end                                     end of the first for-end loop.
A
```

The program is executed in the Command Window to create a 4×5 matrix.

```
Enter the number of rows 4
Enter the number of columns 5
A =
     1     2     3     4     5
     2     4     7    11    16
     3     7    14    25    41
     4    11    25    50    91
```