Introduction to programming in MATLAB

Dr. G.H.J. Lanel

Lecture 2

Outline

Outline

1. Starting MATLAB
   - Introduction

2. Basic Constructs of Structured Programming

3. Flow of Control (Branch and Loop Structures)
   - Branch Structure (If)
   - Loop Structure (for, while)

- Nowadays, every programmer is facing lots of mathematical problems
- Solving them on a paper takes time and puts us in danger of making mistakes and getting wrong solutions
- The material of this course covers a popular mathematics computation system MATLAB
- Youtube Vedio- Introduction to MATLAB: https://www.youtube.com/watch?v=jTS5ZmrrzMs

- Origins: Founded in 1984 to create a more productive computation environment beyond that provided by the languages Fortran and C. Headquartered in Natick, Massachusetts. Privately held company.
- List price: $1,900/per copy. Tens of thousands of dollars in ad-dons are available.
- Employees: " More than 1,500 people worldwide "
- Approximate annual revenue: About $100 million (i.e., roughly three times the size of Mathematica).
- Estimated number of users:I guess 5 million, though they write"Our customers are 1,000,000 of the worldâĂŹs leading technical people, in over 100 countries, on all seven continents."

MATLAB windows

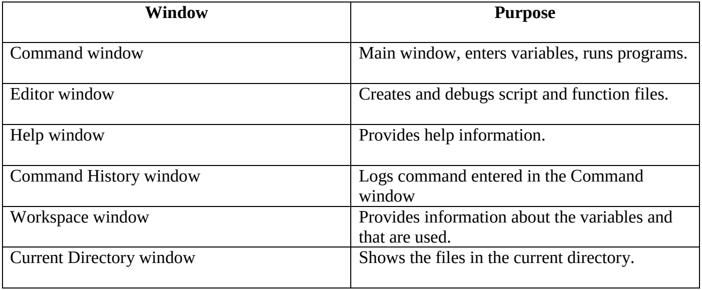
- When you start MATLAB, a special window called the MATLAB desktop appears.
- The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

MATLAB windows

- When you start MATLAB, a special window called the MATLAB desktop appears.
- The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

MATLAB windows

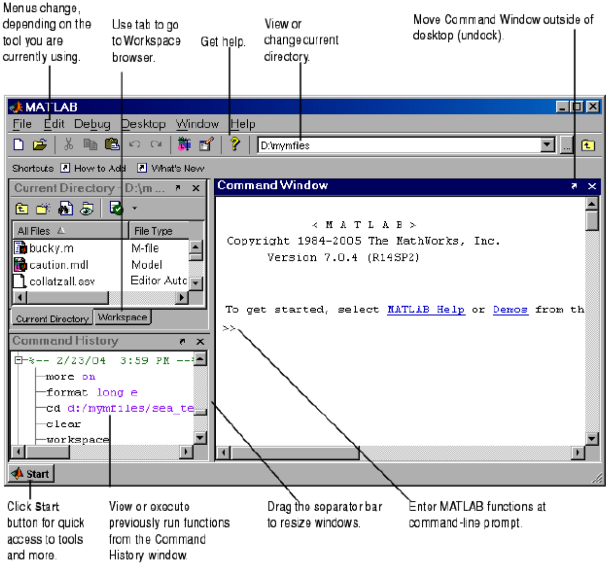
- When you start MATLAB, a special window called the MATLAB desktop appears.
- The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:



Window	Purpose
Command window	Main window, enters variables, runs programs.
Editor window	Creates and debugs script and function files.
Help window	Provides help information.
Command History window	Logs command entered in the Command window
Workspace window	Provides information about the variables and that are used.
Current Directory window	Shows the files in the current directory.

Graphical interface of the MATLAB workspace

Graphical interface of the MATLAB workspace


MATLAB
File Edit Debug Desktop Window Help
D:\mymfiles
Shortcuts How to Add What's New
Current Directory - D:\m ...
Command Window
Command History
Current Directory
Workspace
Start

Menus change, depending on the tool you are currently using.

Use tab to go to Workspace browser.

Get help.

View or change current directory.

Move Command Window outside of desktop (undock).

All Files	File Type
bucky.m	M-file
caution.mdl	Model
collatzall.asv	Editor Auto

```
%-- 2/23/04  3:59 PM --
  more on
  format long e
  cd d:/mymfiles/sea_te
  clear
  workspace
```

```
          < M A T L A B >
Copyright 1984-2005 The MathWorks, Inc.
      Version 7.0.4 (R14SP2)


To get started, select MATLAB Help or Demos from th
>>
```

Click Start button for quick access to tools and more.

View or execute previously run functions from the Command History window.

Drag the separator bar to resize windows.

Enter MATLAB functions at command-line prompt.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Notes for working in the Command window

- To type a command the cursor must be placed next to the command prompt(»)
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously is unchanged.
- Typed commands can be recalled to the command prompt with the up and down arrow keys(↑ & ↓).

The semicolon (;)

- Any output that the command generates is displayed in the Command window.
- If the semicolon is typed at the end of command the output of the command is not displayed.

Comments (%)

- When the symbol % is typed in the beginning of a line, the line is designated as a comment.

Order of precedence

Comments (%)

- When the symbol % is typed in the beginning of a line, the line is designated as a comment.

Order of precedence

Comments (%)

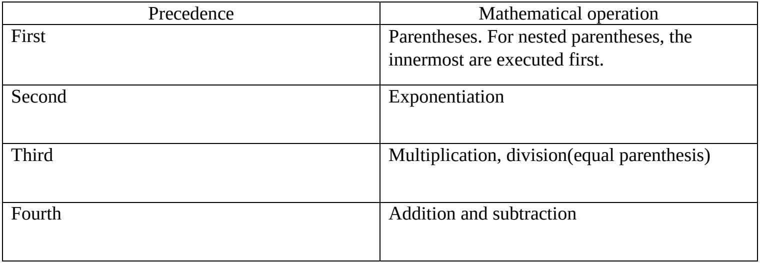
- When the symbol % is typed in the beginning of a line, the line is designated as a comment.

Order of precedence

Comments (%)

- When the symbol % is typed in the beginning of a line, the line is designated as a comment.

Order of precedence



Precedence	Mathematical operation
First	Parentheses. For nested parentheses, the innermost are executed first.
Second	Exponentiation
Third	Multiplication, division(equal parenthesis)
Fourth	Addition and subtraction

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing help format in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing help elfun and help specfun calls up full lists of elementary and special functions respectively.

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing **help format** in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing **help elfun** and **help specfun** calls up full lists of elementary and special functions respectively.

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing **help format** in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing **help elfun** and **help specfun** calls up full lists of elementary and special functions respectively.

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing **help format** in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing help elfun and help specfun calls up full lists of elementary and special functions respectively.

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing **help format** in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing **help elfun** and **help specfun** calls up full lists of elementary and special functions respectively.

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing **help format** in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing **help elfun** and **help specfun** calls up full lists of elementary and special functions respectively.

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing **help format** in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing **help elfun** and **help specfun** calls up full lists of elementary and special functions respectively.

Display formats

- The user can control the format in which MATLAB displays output on the screen.
- The format can be changed with the **format** command.
- MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing **help format** in the Command window.

Mathematical functions

- MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.
- Typing **help elfun** and **help specfun** calls up full lists of elementary and special functions respectively.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with latter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function cannot be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with latter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function can not be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with latter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function cannot be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with letter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function cannot be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with letter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function can not be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with letter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function cannot be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with letter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function can not be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with letter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function can not be used.

Defining scalar variables

- A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value.
- Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statement and commands.
- A variable is actually a name of memory location.

Rules about variable names

Variable name:

- Can be up to 63 characters long, can contain letters, digits, and the underscore character.
- Must begin with letter.
- Avoid using the names of a built-in function for a variable. Once a function name is used to define a variable, the function can not be used.

Predefined variables

- A number of frequently used variable are already defined when MATLAB is started. Some of the predefined variables are **ans, pi, eps, inf, i, j, NaN**

Useful commands for managing variables

- The following are commands that can be used to eliminate variables or to obtain information about variables that have been created

Predefined variables

- A number of frequently used variable are already defined when MATLAB is started. Some of the predefined variables are **ans, pi, eps, inf, i, j, NaN**

Useful commands for managing variables

- The following are commands that can be used to eliminate variables or to obtain information about variables that have been created.

Predefined variables

- A number of frequently used variable are already defined when MATLAB is started. Some of the predefined variables are **ans, pi, eps, inf, i, j, NaN**

Useful commands for managing variables

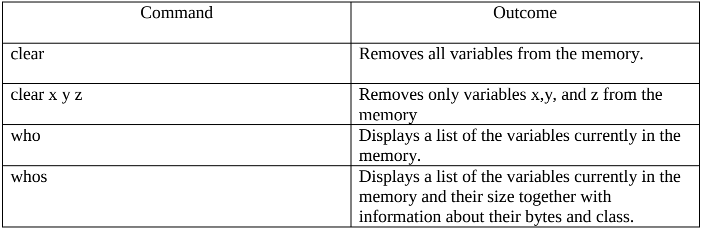
- The following are commands that can be used to eliminate variables or to obtain information about variables that have been created.

Predefined variables

- A number of frequently used variable are already defined when MATLAB is started. Some of the predefined variables are **ans, pi, eps, inf, i, j, NaN**

Useful commands for managing variables

- The following are commands that can be used to eliminate variables or to obtain information about variables that have been created.



Command	Outcome
clear	Removes all variables from the memory.
clear x y z	Removes only variables x,y, and z from the memory
who	Displays a list of the variables currently in the memory.
whos	Displays a list of the variables currently in the memory and their size together with information about their bytes and class.

The **fprintf** Function

- The general form of the fprintf function is: **fprintf (format, data)**
- **format** is a string that controls the way the data is to be printed, and **data** is one or more scalars or arrays to be printed.
- The format is a character string containing text to be printed plus special characters describing the format of the data.

Example:

```
temp = 78.234567989;
fprintf('The temperature is %f degrees. \n',temp)
```

The **fprintf** Function

- The general form of the fprintf function is: **fprintf (format, data)**
- **format** is a string that controls the way the data is to be printed, and **data** is one or more scalars or arrays to be printed.
- The format is a character string containing text to be printed plus special characters describing the format of the data.

Example:

```
temp = 78.234567989;
fprintf('The temperature is %f degrees. \n',temp)
```

The **fprintf** Function

- The general form of the fprintf function is: **fprintf (format, data)**
- **format** is a string that controls the way the data is to be printed, and **data** is one or more scalars or arrays to be printed.
- The format is a character string containing text to be printed plus special characters describing the format of the data.

Example:

```
temp = 78.234567989;
fprintf('The temperature is %f degrees. \n',temp)
```

The **fprintf** Function

- The general form of the fprintf function is: **fprintf (format, data)**
- **format** is a string that controls the way the data is to be printed, and **data** is one or more scalars or arrays to be printed.
- The format is a character string containing text to be printed plus special characters describing the format of the data.

Example:

```
temp = 78.234567989;
fprintf('The temperature is %f degrees. \n',temp)
```

The **fprintf** Function

- The general form of the fprintf function is: **fprintf (format, data)**
- **format** is a string that controls the way the data is to be printed, and **data** is one or more scalars or arrays to be printed.
- The format is a character string containing text to be printed plus special characters describing the format of the data.

Example:

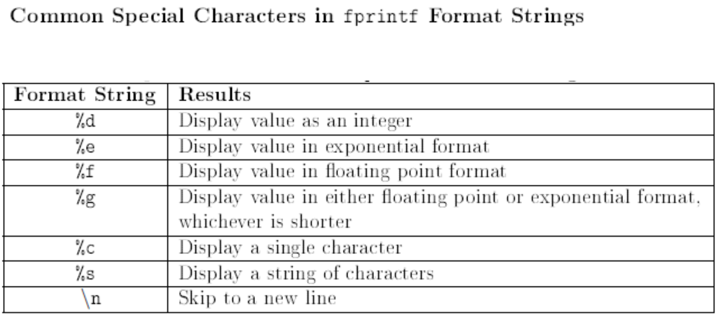
```
temp = 78.234567989;
fprintf('The temperature is %f degrees. \n',temp)
```

The **fprintf** Function

- The general form of the fprintf function is: **fprintf (format, data)**
- **format** is a string that controls the way the data is to be printed, and **data** is one or more scalars or arrays to be printed.
- The format is a character string containing text to be printed plus special characters describing the format of the data.

Example:

```
temp = 78.234567989;
fprintf('The temperature is %f degrees. \n',temp)
```



Common Special Characters in fprintf Format Strings

Format String	Results
%d	Display value as an integer
%e	Display value in exponential format
%f	Display value in floating point format
%g	Display value in either floating point or exponential format, whichever is shorter
%c	Display a single character
%s	Display a string of characters
\n	Skip to a new line

Outline

1. Starting MATLAB
   - Introduction
2. Basic Constructs of Structured Programming
3. Flow of Control (Branch and Loop Structures)
   - Branch Structure (If)
   - Loop Structure (for, while)

There are three main programming constructs:

- **Sequence**
  The Sequence construct refers to writing a group of programming statements in a sequence.

- **Branch (Selection)**
  The Branch construct enables us to change the flow of control if a given condition is satisfied.

- Repetition (Loop)
  The Loop construct enables the program to run a statement (or a group of statements) a number of times.

There are three main programming constructs:

- **Sequence**
  The Sequence construct refers to writing a group of programming statements in a sequence.
- **Branch (Selection)**
  The Branch construct enables us to change the flow of control if a given condition is satisfied.
- **Repetition (Loop)**
  The Loop construct enables the program to run a statement (or a group of statements) a number of times.

There are three main programming constructs:

- **Sequence**
  The Sequence construct refers to writing a group of programming statements in a sequence.
- **Branch (Selection)**
  The Branch construct enables us to change the flow of control if a given condition is satisfied.
- **Repetition (Loop)**
  The Loop construct enables the program to run a statement (or a group of statements) a number of times.

There are three main programming constructs:

- **Sequence**
  The Sequence construct refers to writing a group of programming statements in a sequence.
- **Branch (Selection)**
  The Branch construct enables us to change the flow of control if a given condition is satisfied.
- **Repetition (Loop)**
  The Loop construct enables the program to run a statement (or a group of statements) a number of times.

MATLAB script files

- MATLAB programming codes are saved in files with extension.m. This gives rise to the so-called MATLAB M-files.
- An M-file may contain a Matlab script or a MATLAB function.
- A script file is a sequence of MATLAB commands, also called a program.
- When a script file is executed it runs in the order that they are written just as they typed in the command window.
- When script file has a command that generates and output, the output is displayed in the command window.
- Using a script file is convenient because it can be edited and executed in many times.

MATLAB script files

- MATLAB programming codes are saved in files with extension.m. This gives rise to the so-called MATLAB M-files.
- An M-file may contain a Matlab script or a MATLAB function.
- A script file is a sequence of MATLAB commands, also called a program.
- When a script file is executed it runs in the order that they are written just as they typed in the command window.
- When script file has a command that generates and output, the output is displayed in the command window.
- Using a script file is convenient because it can be edited and executed in many times.

MATLAB script files

- MATLAB programming codes are saved in files with extension.m. This gives rise to the so-called MATLAB M-files.
- An M-file may contain a Matlab script or a MATLAB function.
- A script file is a sequence of MATLAB commands, also called a program.
- When a script file is executed it runs in the order that they are written just as they typed in the command window.
- When script file has a command that generates and output, the output is displayed in the command window.
- Using a script file is convenient because it can be edited and executed in many times.

MATLAB script files

- MATLAB programming codes are saved in files with extension.m. This gives rise to the so-called MATLAB M-files.
- An M-file may contain a Matlab script or a MATLAB function.
- A script file is a sequence of MATLAB commands, also called a program.
- When a script file is executed it runs in the order that they are written just as they typed in the command window.
- When script file has a command that generates and output, the output is displayed in the command window.
- Using a script file is convenient because it can be edited and executed in many times.

MATLAB script files

- MATLAB programming codes are saved in files with extension.m. This gives rise to the so-called MATLAB M-files.
- An M-file may contain a Matlab script or a MATLAB function.
- A script file is a sequence of MATLAB commands, also called a program.
- When a script file is executed it runs in the order that they are written just as they typed in the command window.
- When script file has a command that generates and output, the output is displayed in the command window.
- Using a script file is convenient because it can be edited and executed in many times.

MATLAB script files

- MATLAB programming codes are saved in files with extension.m. This gives rise to the so-called MATLAB M-files.
- An M-file may contain a Matlab script or a MATLAB function.
- A script file is a sequence of MATLAB commands, also called a program.
- When a script file is executed it runs in the order that they are written just as they typed in the command window.
- When script file has a command that generates and output, the output is displayed in the command window.
- Using a script file is convenient because it can be edited and executed in many times.

Outline

1 Starting MATLAB
- Introduction

2 Basic Constructs of Structured Programming

3 Flow of Control (Branch and Loop Structures)
- Branch Structure (If)
- Loop Structure (for, while)

If statement

- The if statement in MATLAB allows us to change the flow of control in our computer programs based on a specified condition.
- The simplest way of using the if statement is as follows:

```
if <condition>
    statement 1
    statement 2
    .
    .
    .
end
```

If statement

- The if statement in MATLAB allows us to change the flow of control in our computer programs based on a specified condition.
- The simplest way of using the if statement is as follows:

```
if <condition>
     statement 1
     statement 2
     .
     .
     .
end
```

If statement

- The if statement in MATLAB allows us to change the flow of control in our computer programs based on a specified condition.
- The simplest way of using the if statement is as follows:

```
if <condition>
      statement 1
      statement 2
      .
      .
      .
end
```

If statement

- The if statement in MATLAB allows us to change the flow of control in our computer programs based on a specified condition.
- The simplest way of using the if statement is as follows:

```
if <condition>
    statement 1
    statement 2
    .
    .
    .
end
```

Second form of using the if statement provides a way to test for a condition and execute the appropriate statement (or set of statements) if a condition is true or false.

```
if <condition>
    statement 1
    statement 2
    .
    .
    .
    else
    statement 1
    statement 2
    .
    .
    .
end
```

Second form of using the if statement provides a way to test for a condition and execute the appropriate statement (or set of statements) if a condition is true or false.

```
if <condition>
    statement 1
    statement 2
    .
    .
    .
    else
    statement 1
    statement 2
    .
    .
    .
end
```

The most general way of using the if statement is outlined below.

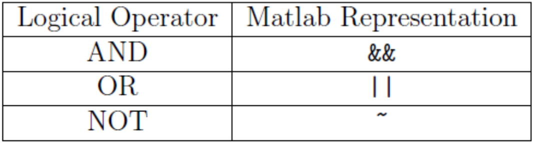
```
if <condition>
     statements
     elseif <condition>
         statements
     elseif <condition>
         statements
     .
     .
     else
         statements
end
```

The most general way of using the if statement is outlined below.

```
if <condition>
    statements
    elseif <condition>
        statements
    elseif <condition>
        statements
    .
    .
    else
        statements
end
```

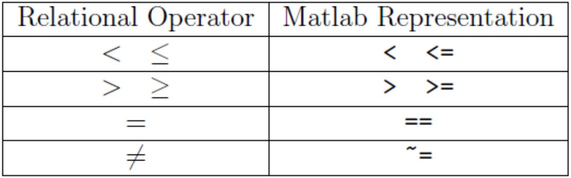
- When discussing the if statement it is natural to discuss the logical operators, AND, OR, and NOT. In addition, we can use the relational operators in the conditional expressions.

- When discussing the if statement it is natural to discuss the logical operators, AND, OR, and NOT. In addition, we can use the relational operators in the conditional expressions.



Logical Operator	Matlab Representation
AND	&&
OR	||
NOT	~

Logical Operators in Matlab



Relational Operator	Matlab Representation
$<$ $\leq$	< <=
$>$ $\geq$	> >=
$=$	==
$\neq$	~=

Relational Operators in Matlab

Repetition(Loops)

- One of the strongest attributes of a computer is its ability to do fast repetitive operations on a set of data.
- we use this feature through loops when we want to repeat certain parts of our program over and over again.
- In MATLAB there are two basic forms of loop constructs: **for** loops and **while** loops.
- The major difference between these two types of loops is in how the repetition is controlled.
- The code in a for loop is repeated a specified number of times, and the number of repetitions is known before the loops starts.
- The code in a while loop is repeated an indefinite number of times until some user-specified condition is satisfied.

Repetition(Loops)

- One of the strongest attributes of a computer is its ability to do fast repetitive operations on a set of data.
- we use this feature through loops when we want to repeat certain parts of our program over and over again.
- In MATLAB there are two basic forms of loop constructs: **for** loops and **while** loops.
- The major difference between these two types of loops is in how the repetition is controlled.
- The code in a for loop is repeated a specified number of times, and the number of repetitions is known before the loops starts.
- The code in a while loop is repeated an indefinite number of times until some user-specified condition is satisfied.

Repetition(Loops)

- One of the strongest attributes of a computer is its ability to do fast repetitive operations on a set of data.
- we use this feature through loops when we want to repeat certain parts of our program over and over again.
- In MATLAB there are two basic forms of loop constructs: **for** loops and **while** loops.
- The major difference between these two types of loops is in how the repetition is controlled.
- The code in a for loop is repeated a specified number of times, and the number of repetitions is known before the loops starts.
- The code in a while loop is repeated an indefinite number of times until some user-specified condition is satisfied.

Repetition(Loops)

- One of the strongest attributes of a computer is its ability to do fast repetitive operations on a set of data.
- we use this feature through loops when we want to repeat certain parts of our program over and over again.
- In MATLAB there are two basic forms of loop constructs: **for** loops and **while** loops.
- The major difference between these two types of loops is in how the repetition is controlled.
- The code in a for loop is repeated a specified number of times, and the number of repetitions is known before the loops starts.
- The code in a while loop is repeated an indefinite number of times until some user-specified condition is satisfied.

Repetition(Loops)

- One of the strongest attributes of a computer is its ability to do fast repetitive operations on a set of data.
- we use this feature through loops when we want to repeat certain parts of our program over and over again.
- In MATLAB there are two basic forms of loop constructs: **for** loops and **while** loops.
- The major difference between these two types of loops is in how the repetition is controlled.
- The code in a for loop is repeated a specified number of times, and the number of repetitions is known before the loops starts.
- The code in a while loop is repeated an indefinite number of times until some user-specified condition is satisfied.

Repetition(Loops)

- One of the strongest attributes of a computer is its ability to do fast repetitive operations on a set of data.
- we use this feature through loops when we want to repeat certain parts of our program over and over again.
- In MATLAB there are two basic forms of loop constructs: **for** loops and **while** loops.
- The major difference between these two types of loops is in how the repetition is controlled.
- The code in a for loop is repeated a specified number of times, and the number of repetitions is known before the loops starts.
- The code in a while loop is repeated an indefinite number of times until some user-specified condition is satisfied.

for Loop

- To execute a statement (or group of statements) a specified number of times we use the for loop.
- The basic usage of the for loop is as follows.

```
for index = expression
    statement group (body of the loop)
end
```

- The expression usually takes the form of a vector in shortcut notation **first:increment:last**.
- The index of the for loop must be a variable.

for Loop

- To execute a statement (or group of statements) a specified number of times we use the for loop.
- The basic usage of the for loop is as follows.

```
for index = expression
     statement group (body of the loop)
end
```

- The expression usually takes the form of a vector in shortcut notation **first:increment:last**.
- The index of the for loop must be a variable.

for Loop

- To execute a statement (or group of statements) a specified number of times we use the for loop.
- The basic usage of the for loop is as follows.

```
for index = expression
    statement group (body of the loop)
end
```

- The expression usually takes the form of a vector in shortcut notation **first:increment:last**.
- The index of the for loop must be a variable.

for Loop

- To execute a statement (or group of statements) a specified number of times we use the for loop.
- The basic usage of the for loop is as follows.

```
for index = expression
    statement group (body of the loop)
end
```

- The expression usually takes the form of a vector in shortcut notation **first:increment:last**.
- The index of the for loop must be a variable.

Example 1

```
for i = 1 : 10
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 2

```
for i = 1 : 2 : 10
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 3

```
for i = 10 : -1 : 1
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 1

```
for i = 1 : 10
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 2

```
for i = 1 : 2 : 10
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 3

```
for i = 10 : -1 : 1
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 1

```
for i = 1 : 10
    fprintf(’%d ’, i);
end
fprintf(’\n’);
```

Example 2

```
for i = 1 : 2 : 10
    fprintf(’%d ’, i);
end
fprintf(’\n’);
```

Example 3

```
for i = 10 : -1 : 1
    fprintf(’%d ’, i);
end
fprintf(’\n’);
```

Example 1

```
for i = 1 : 10
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 2

```
for i = 1 : 2 : 10
    fprintf('%d ', i);
end
fprintf('\n');
```

Example 3

```
for i = 10 : -1 : 1
    fprintf('%d ', i);
end
fprintf('\n');
```

Example

Calculate the summation of 1 + 2 + ... + 100.

```
sum = 0;
    for i=1:100
    sum = sum + i;
end
fprintf(' The summation is %d \n ' ,sum);
```

```
» The summation is 5050
```

Example

Calculate the summation of 1 + 2 + ... + 100.

```
sum = 0;
    for i=1:100
    sum = sum + i;
end
fprintf(' The summation is %d \n ' ,sum);
```

```
» The summation is 5050
```

Example

Calculate the summation of 1 + 2 + ... + 100.

```
sum = 0;
    for i=1:100
    sum = sum + i;
end
fprintf(' The summation is %d \n ' ,sum);
```

```
» The summation is 5050
```

Example

Given a natural number n form an n× n Hilbert matrix whose (i, j)-component is defined as

H(i,j) = $\frac{1}{(i+j-1)}$, display the matrix.

```
n = input('Enter the number of terms := '); % change n to any value
for i = 1 : n
    for j = 1 : n
        H(i,j) = 1/(i+j-1);
    end
end
disp(H);
```

Example

Given a natural number n form an n× n Hilbert matrix whose (i, j)-component is defined as

$$H(i,j) = \frac{1}{(i+j-1)}$$, display the matrix.

```
n = input('Enter the number of terms := '); % change n to any value
for i = 1 : n
    for j = 1 : n
        H(i,j) = 1/(i+j-1);
    end
end
disp(H);
```

while Loop

- To execute a statement (or group of statements) a specified condition we use the while loop.
- The basic usage of the while loop is as follows.

```
while expression
      statement group
end
```

- The controlling expression produces a logical value.
- If the expression is always true (for example, we made an mistake in the expression), the loop becomes an infinite loop and we need to use the **Ctrl+C** key to abort it.

while Loop

- To execute a statement (or group of statements) a specified condition we use the while loop.
- The basic usage of the while loop is as follows.

```
while expression
      statement group
end
```

- The controlling expression produces a logical value.
- If the expression is always true (for example, we made an mistake in the expression), the loop becomes an infinite loop and we need to use the **Ctrl+C** key to abort it.

while Loop

- To execute a statement (or group of statements) a specified condition we use the while loop.
- The basic usage of the while loop is as follows.

```
while expression
    statement group
end
```

- The controlling expression produces a logical value.
- If the expression is always true (for example, we made an mistake in the expression), the loop becomes an infinite loop and we need to use the **Ctrl+C** key to abort it.

while Loop

- To execute a statement (or group of statements) a specified condition we use the while loop.
- The basic usage of the while loop is as follows.

```
while expression
    statement group
end
```

- The controlling expression produces a logical value.
- If the expression is always true (for example, we made an mistake in the expression), the loop becomes an infinite loop and we need to use the **Ctrl+C** key to abort it.

- If the expression is true, the statement group will be executed. The process will be repeated until the expression becomes false.
- If the expression is false, the program will execute the first statement after the end of while loop.

Example: Calculate the summation of 1 + 2 + ... + n; where n(>0) is given

```
n = input('Input n : ');
sum = 0;
current = 1;
while current <= n
        sum = sum + current;
        current = current + 1;
end
fprintf(' The summation is %d \n ',sum);
```

- If the expression is true, the statement group will be executed. The process will be repeated until the expression becomes false.
- If the expression is false, the program will execute the first statement after the end of while loop.

Example: Calculate the summation of 1 + 2 + ... + n. where n(>0) is given

```
n = input('Input n : ');
sum = 0;
current = 1;
while current <= n
    sum = sum + current;
    current = current + 1;
end
fprintf(' The summation is %d \n ' ,sum);
```

- If the expression is true, the statement group will be executed. The process will be repeated until the expression becomes false.
- If the expression is false, the program will execute the first statement after the end of while loop.

Example: Calculate the summation of 1 + 2 + ... + n. where n(>0) is given

```
n = input('Input n : ');
sum = 0;
current = 1;
while current <= n
    sum = sum + current;
    current = current + 1;
end
fprintf(' The summation is %d \n ' ,sum);
```

- If the expression is true, the statement group will be executed. The process will be repeated until the expression becomes false.
- If the expression is false, the program will execute the first statement after the end of while loop.

Example: Calculate the summation of 1 + 2 + ... + n. where n(>0) is given

```
n = input(’Input n : ’);
sum = 0;
current = 1;
while current <= n
     sum = sum + current;
     current = current + 1;
end
fprintf(’ The summation is %d \n ’ ,sum);
```

- If the expression is true, the statement group will be executed. The process will be repeated until the expression becomes false.
- If the expression is false, the program will execute the first statement after the end of while loop.

Example: Calculate the summation of 1 + 2 + ... + n. where n(>0) is given

```
n = input(’Input n : ’);
sum = 0;
current = 1;
while current <= n
    sum = sum + current;
    current = current + 1;
end
fprintf(’ The summation is %d \n ’ ,sum);
```

End!