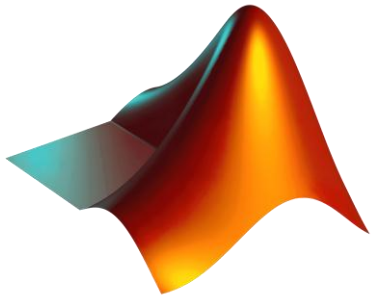


ELE 4311 – MATLAB for Engineers (OE -1)

SHAILESH K R (E) **BHARATHI R B (F)** **SHWETHA V (G)**

Department of Electrical and Electronics Engineering

Manipal Institute of Technology, Manipal

About the course

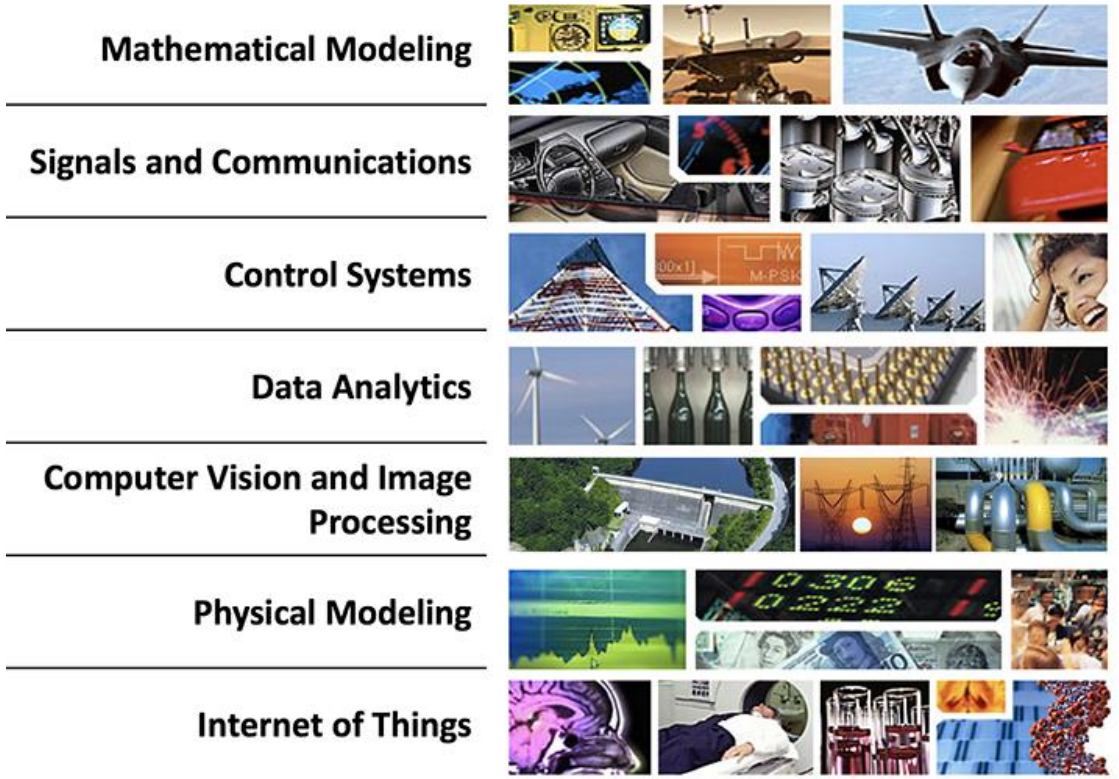
- ELE 4311 – MATLAB for Engineers (2 1 0 3)
- **36** hours = **24** Lecture hours + **12** Tutorials
- Open Elective
- Quiz and Assignment as per notification.
- **MATLAB Documentation : https://in.mathworks.com/help/matlab/**

About MATLAB

- **MAT**rix **LAB**oratory

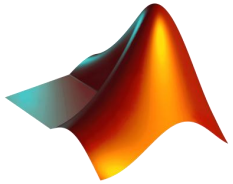
- All variables are treated as matrices

- 1x1, rows x 1,
- 1 x cols,
- rows x cols,
- rows x cols x pages



Millions of engineers and scientists worldwide use MATLAB to analyze and design the systems and products transforming our world.

About MATLAB



- **Built-in graphics** make it easy to **visualize** and gain insights from **data**.
- MATLAB helps you take your ideas beyond the desktop.
- **Large data-sets** can be analyzed
- MATLAB **code** can be **integrated with other languages**.
- Current Version : **R2024b**

Typical applications of MATLAB

- **Mathematics** and **computation**
- **Algorithm** development
- **Data acquisition**
- **Modeling**, **simulation**, and **prototyping**
- **Data analysis**, **exploration**, and **visualization**
- **Scientific** and engineering **graphics**
- **Application development**, including graphical user interface (GUI) building

Choosing a Computer to Run MATLAB and Simulink Products

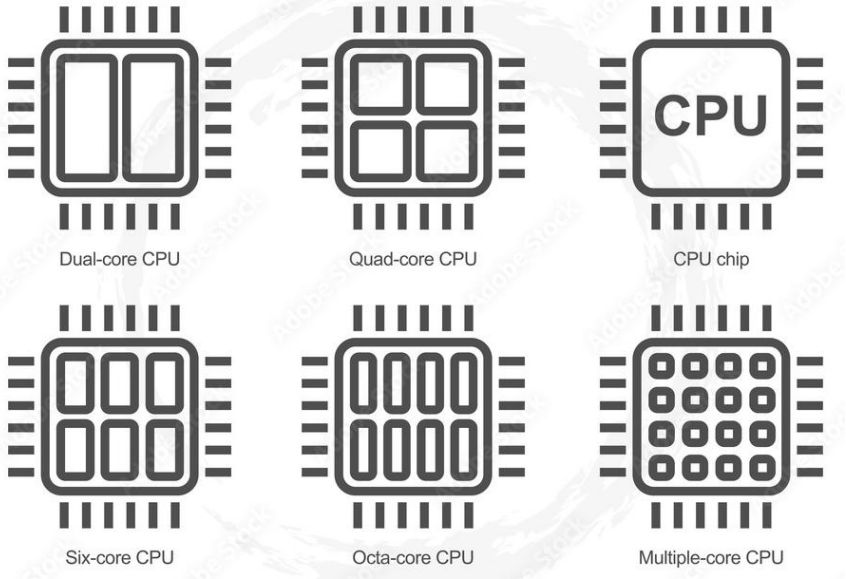
- "**Predicting** how **MATLAB** will **perform** while **running** an **application** on a **particular computer is difficult**."

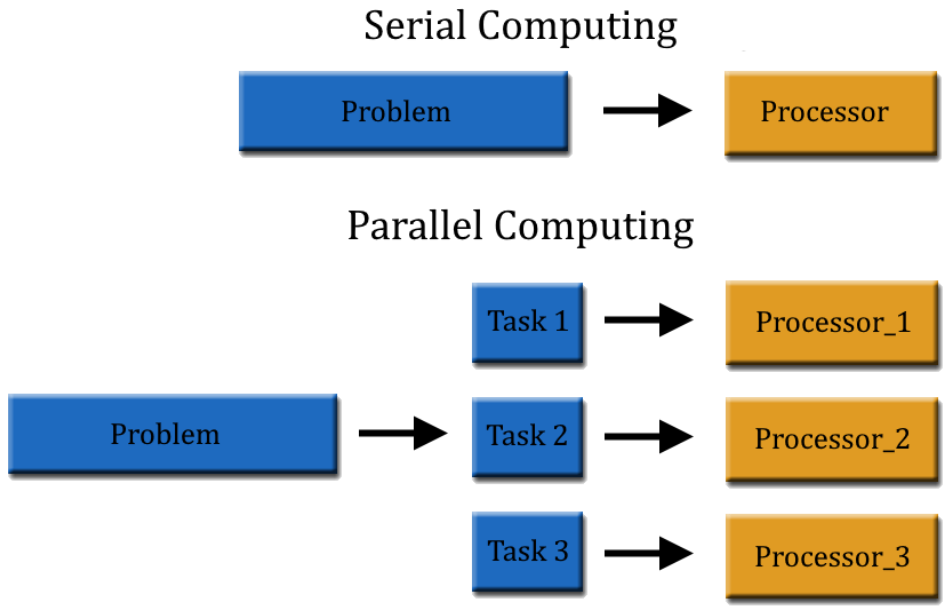
- **Each component** of a **typical computer configuration** has an **impact on MATLAB performance**.

- MATLAB performance is similar on Windows, Mac OS, and Linux, although differences can occur among platforms because of
- **compiler**,
- **third-party libraries**,
- **disk- or graphics-intensive operations**

Hardware Considerations (**CPU**)

- **Central Processing Unit (CPU)**
- Computers with more CPU cores (**MULTIPLE PROCESSORs**) can outperform those with a lower core (**ONE PROCESSOR**) count, but results will vary with the MATLAB application.





For additional capability, **Parallel Computing Toolbox** offers parallel programming constructs that more directly leverage multiple computer cores.

Hardware Considerations

- **Memory** – MATLAB and other concurrent programs, how they use available **physical memory** and need for **virtual memory**.
  - **Thrashing** occurs when a computer's virtual memory resources are overused thereby inhibiting most application-level processing.
- **Hard disk** - The hard disk speed is a significant factor in MATLAB start-up time.
  - **Solid-state drives** are recommended
- **Graphics Processing Unit (GPU) for display** – MATLAB uses OpenGL technology for rendering graphics - **graphics card with superior OpenGL support**
- **Graphics Processing Unit (GPU) for computation** - NVIDIA GPUs are recommended

The MATLAB System

- The MATLAB system consists of five main parts:

- **Development Environment (GUI)** – Set of tools that facilitate using MATLAB functions and files.
  - MATLAB desktop
  - Command Window,
  - Command history,
  - File Editor
- Documentation/Help

- **The MATLAB Mathematical Function Library -** Vast collection of computational algorithms

The MATLAB System

- **The MATLAB Language** - high-level matrix/array language - with **control flow statements**, **functions**, **data structures**, and **object-oriented programming features**

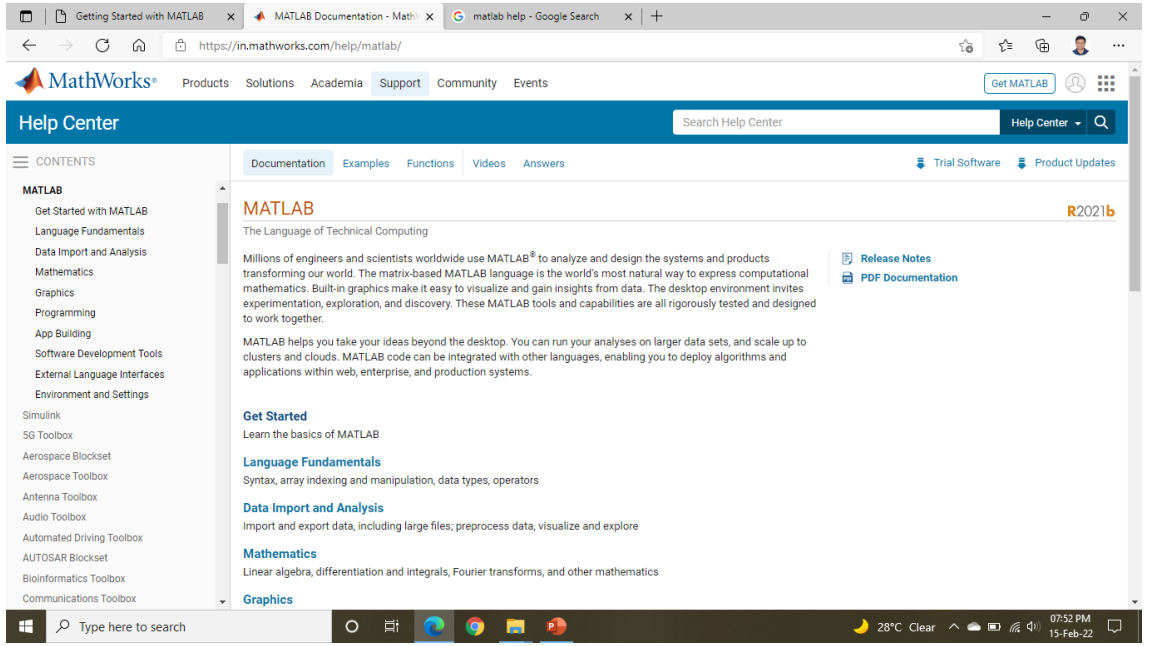
- **Graphics** - Two-dimensional and three-dimensional data visualization, image processing

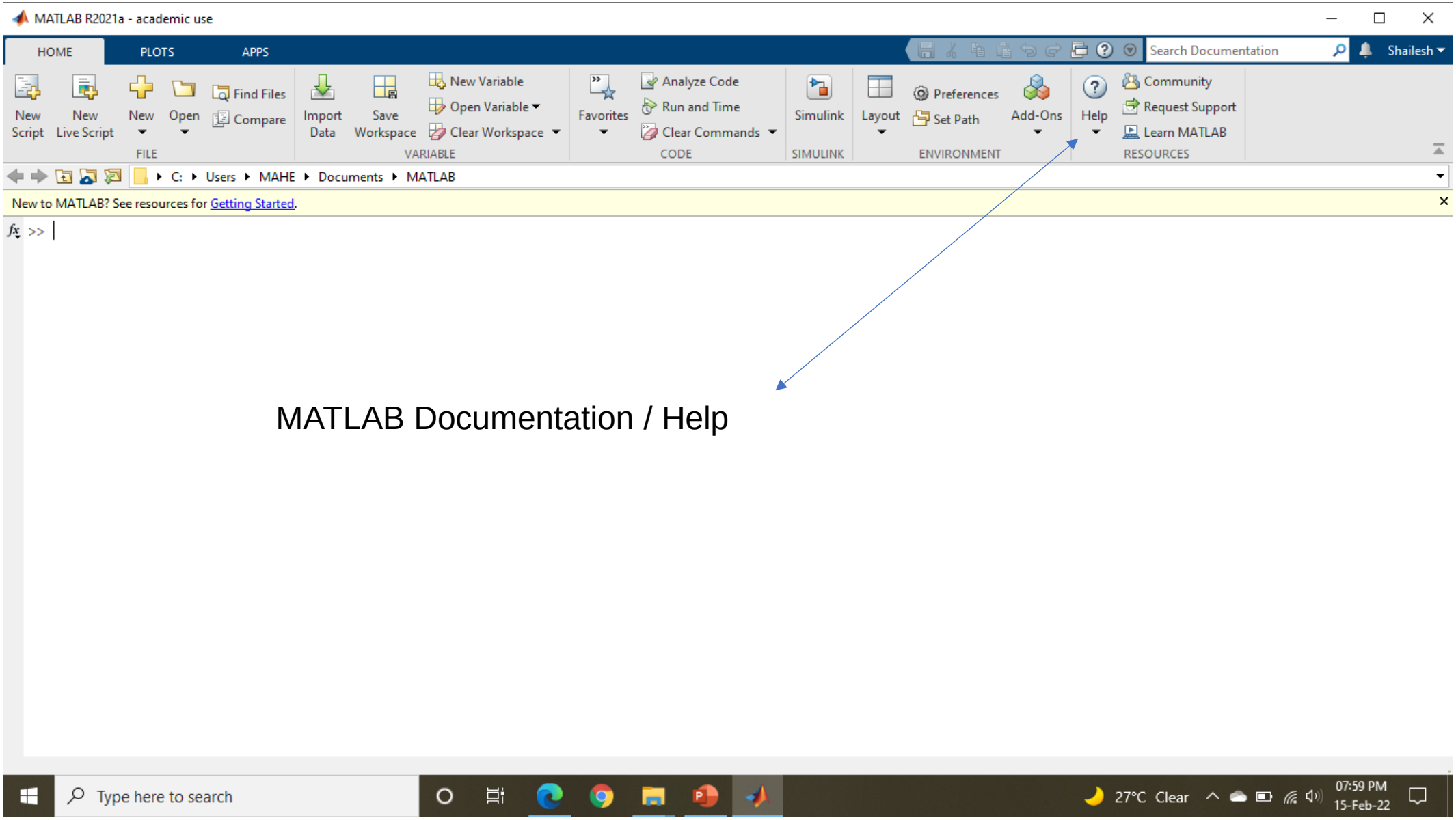
- **The MATLAB External Interfaces/API –** Allow write code in C and interact with MATLAB

MATLAB Documentation

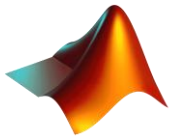
- MATLAB provides extensive online documentation.
- https://in.mathworks.com/help/matlab/

- The MATLAB online help provides task-oriented and reference information about MATLAB features.




MATLAB R2021a - academic use
HOME
PLOTS
APPS
Search Documentation
Shailesh
New Script
New Live Script
New
Open
Find Files
Compare
FILE
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
VARIABLE
Favorites
Analyze Code
Run and Time
Clear Commands
CODE
Simulink
SIMULINK
Layout
Preferences
Set Path
Add-Ons
ENVIRONMENT
Help
Community
Request Support
Learn MATLAB
RESOURCES
C: ▸ Users ▸ MAHE ▸ Documents ▸ MATLAB
New to MATLAB? See resources for Getting Started.
>>
MATLAB Documentation / Help
Type here to search
27°C Clear
07:59 PM
15-Feb-22

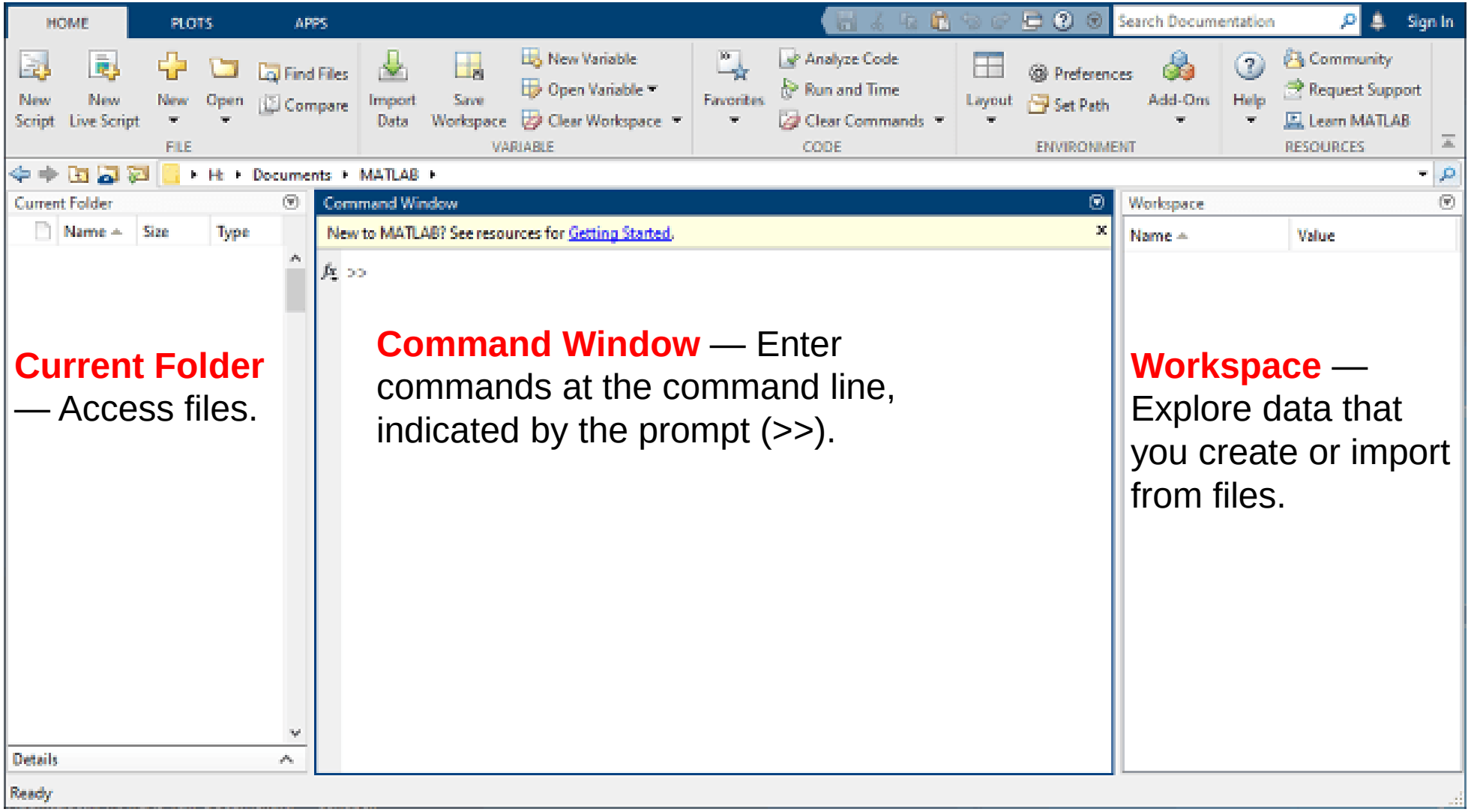
Starting and Quitting MATLAB

- On Windows platforms, start MATLAB by double-clicking the MATLAB shortcut icon on your Windows desktop. 

- Quitting MATLAB To end your MATLAB session, type **quit** in the Command Window.

MATLAB Desktop

When MATLAB is started , the desktop appears in its default layout.



Current Folder — Access files.

Command Window — Enter commands at the command line, indicated by the prompt (>>).

Workspace — Explore data that you create or import from files.

Rules on Variables

- Variable, function, file names are **case sensitive**, e.g., **NAME** and **Name** are 2 distinct names.

- **variable begins with a letter**, e.g., A2z or a2z

- can be a mix of letters, digits, and underscores (e.g., vector_A)

- reserved characters: **% = + – ~ ; : ! ' [ ] ( ) , @ # $ & ^**

Rules on File name

- MATLAB command files should be named with a suffix of **".m"**, e.g., **myfile.m**.

- An **m-file** typically contains a **sequence** of **MATLAB** commands that will be **executed in order**

- A file may contain a **collection** of **commands**, **functions**

- Note: To run, enter **m-file**, without .m, e.g.,
- >> **myfile**

Reserved Characters % = ; ,

- Some characters are reserved by MATLAB for various purposes.
- Some as arithmetic or matrix operators: **=, +, - , *, / , ** and others are used to perform a multitude of operations.
- **Reserved characters** cannot be used in **variable** or **function names**.
- anything after **%** until the end of line is treated as comments
- >> a = 3 *% a is defined a to have the value 3*

Reserved Characters % = ; ,

- >> a = 3; **% ";" suppresses printing**
- >>

- >> b = 4; c = 5; **% ";" enables multiple commands on same line**
- >>

- >> d = 6, e = 7; **% "," delimits commands but enables printing**
- d = 6

Reserved Characters : [ ] ( )

- >> x = 1:2:9 **% define vector x with : operator (begin : interval : end)**
- x =
- 1 3 5 7 9

- >> y = 3:5 **% interval is defaulted to 1; same as y=[3:5]**
- y =
- 3 4 5

Reserved Characters : [ ] ( )

- >> X = [1, 2, 3; 4, 5, 6] % 2D array. **The ; is vertical concatenation**.
- % [ ] for arrays. Prevents ambiguity
- **% ; concatenates vertically (new row)**
- **% , concatenates horizontally (new columns)**
- X =
- 1 2 3
- 4 5 **6**
- >> X(2,3) % ( ) for subscripting;
- ans =
- 6

Reserved Characters … and '

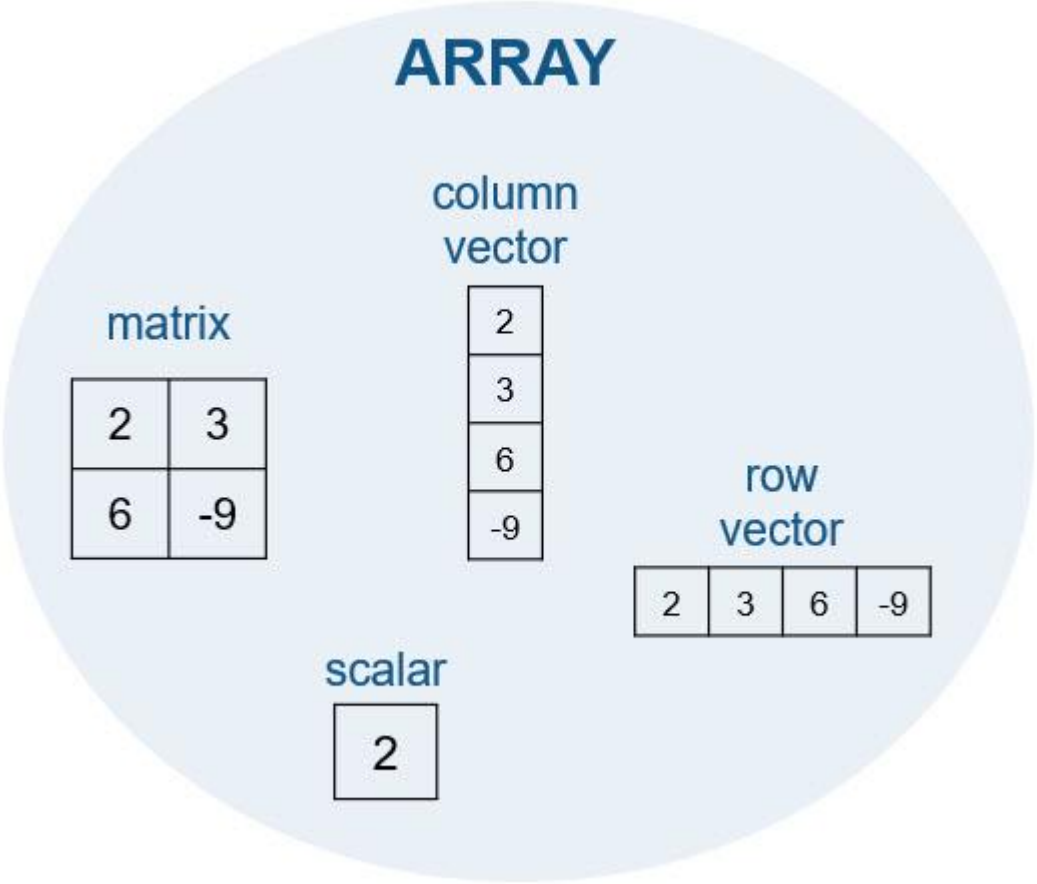
- >> x = [1 2 3 … **% ellipses … means to be continued on the next line**
- 4 5 6]
- x =
- 1 2 3 4 5 6

- >> s = 'this is a character string'; **% blanks preserved within quotes**

Reserved Characters … and '

- >> x = [1 2 3]' **% ' performs transpose (e.g., turns row into column)**
- x =
- 1
- 2
- 3
- >> X = [1 2 3; 4 5 6]; size(X) **% figure out the size (dimensions) of X**
- ans =
- 2 3
- >> X = [1 2 3; 4 5 6]; numel(X) **% total number of entries in X**
- ans =
- 6

Matrices and Arrays



Some useful commands

- **clc**; Clear command window. (**home**)
- **clear**; Clear variables and functions from memory.
- **format**; Set output format. **Short** (4 digits) **Long** (15 digits)
- Try : **help disp;**

Creating variable at the command line

- Create a variable named **'a'** by typing this statement at the command line (>>):
- >> a= 1 ;

- When you do not specify an output variable, MATLAB uses the variable **ans**, short for answer, to store the results of your calculation.

- >> sin(a)

- ans =

- 0.8415

MATLAB R2021a - academic use

HOME PLOTS APPS

Search Documentation | Shailesh

New Script | New Live Script | New | Open | Find Files | Compare — FILE

Import Data | Save Workspace | New Variable | Open Variable | Clear Workspace — VARIABLE

Favorites | Analyze Code | Run and Time | Clear Commands — CODE

Simulink — SIMULINK

Layout | Preferences | Set Path | Add-Ons — ENVIRONMENT

Help | Community | Request Support | Learn MATLAB — RESOURCES

C: ▸ Users ▸ MAHE ▸ Documents ▸ MATLAB

Current Folder

Name

Details

Command Window

New to MATLAB? See resources for Getting Started.

```
>> a=1

a =

     1

>> sin(a)

ans =

    0.8415

>> b=3

b =

     3

>> c=a+b;
>> c

c =

     4

>>
```

Workspace

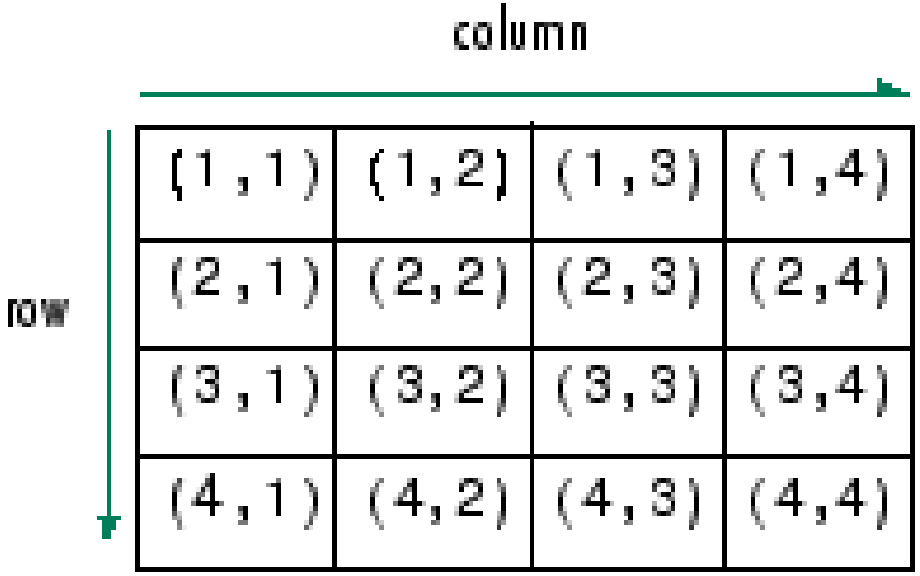
Name	Value
a	1
ans	0.8415
b	3
c	4

The **Workspace** window (**on the right**) shows all the **variables** currently in the workspace.

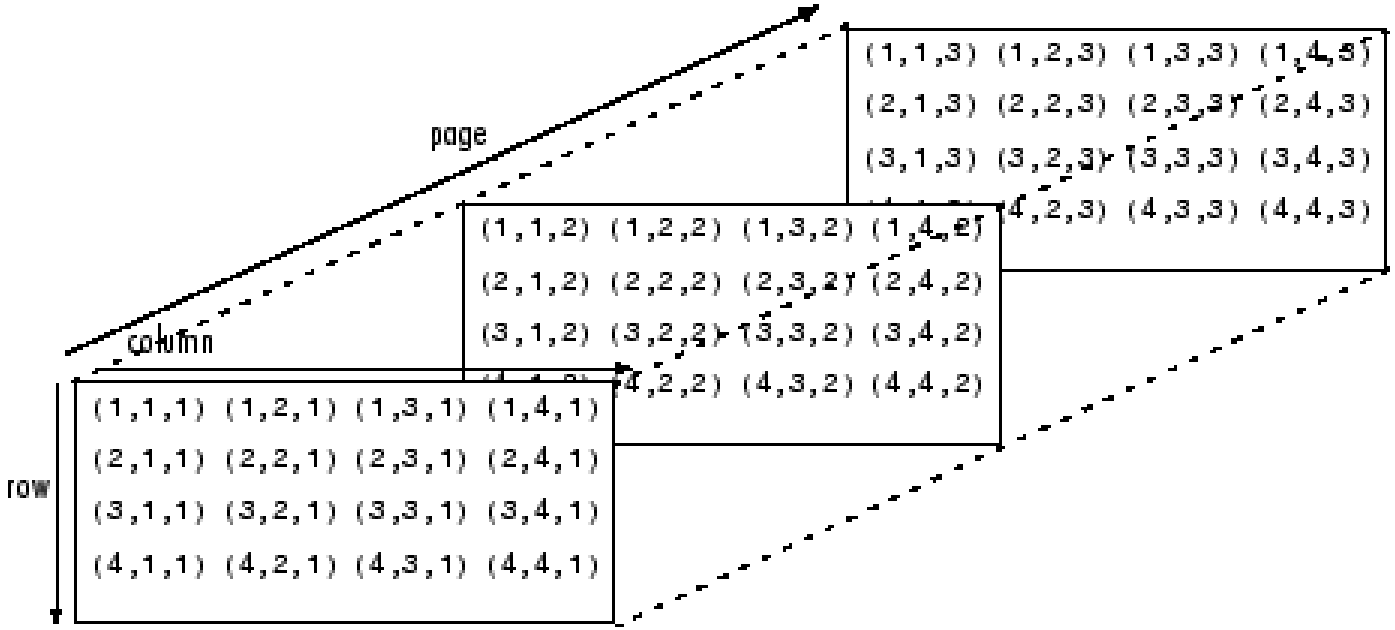
Type here to search | 26°C Fog | 09:50 AM 21-Feb-22

Matrices and Arrays

- All MATLAB variables are multidimensional arrays, no matter what type of data.



2D – Array



3D – Array

Array Creation : >> a = [1 2 3 4] ; creating a **row** vector

c = [3+4i, 4+3j; -i, 10j] ; Array with **complex** numbers

Examples (Try it on MATLAB command window)

- Creating a matrix that has multiple rows, separate the rows with semicolons.
- >> a = [1 3 5; 2 4 6; 7 8 10]

- Functions : **ones**, **zeros, eye**

- z = zeros(5,1) **;** *%create a 5-by-1 column vector of zeros.*

- *O=ones(1,5); %create a 1-by-5 row vector of ones.*

Matrix and Array Operations

- MATLAB allows you to process all the values in a matrix using a single arithmetic operator or function.

- >> O = O+10;
- >> sin(a);

- To transpose (**flip – interchanging rows and columns**) a matrix, use a single quote (‘):

- Matrix multiplication : a*a

Dot operator (.)

- Given : >> a=[1 2 3;4 5 6; 7 8 9];
- Try : >> a.*a
- To perform element-wise multiplication rather than matrix multiplication, use the .* operator.
- Try : **>> a./3** ; AND **>> a.^2;**

Concatenation (append or join)

- A=[a,a] ; concatenating ‘a’ with ‘a’ horizontally.
- C = **horzcat**(a,a);

- A=[a;a] ; concatenating ‘a’ with ‘a’ vertically.
- C = **vertcat**(a,a);

- The pair of square brackets **[ ]** is the concatenation operator.

Saving and Loading Workspace Variables

- Save variables in your workspace to a MATLAB specific file format called a MAT-file using the save command.

- To save the workspace to a MAT-file named **filename.mat**, use the command:
- >> save **filename**
- >> load **filename**

- Saving the variable **"k"** to a new MAT-file called **justk.mat**:
- >> save **justk k**

Using Built-in Functions and Constants

- >> a = pi (π)

- **https://in.mathworks.com/help/phased/ref/physconst.html**

- >> a = sin(30)
- >> a =sin(pi/6)
- >> a = sqrt(16)
- You can perform calculations within the square brackets.
- x = [abs(-4) 4^2];
- y= [sqrt(10) pi^2]

Creating Evenly-Spaced Vectors

- X = [1 2 3 4] – Vector contains evenly spaced number
- Shortcut method - X = 1:4 – **“:”** operator – **start_value : end_value**
- A = 1:2:10 - **start_value : increment : end_value**
- linspace(**start_value** , **end_value**, **number_of_elements**)
- **If you wanted to create an evenly-spaced vector from 1 to 2π with 100 elements, would you use linspace or :?**

Array Creation Functions

- **x = rand(2)** - output will be a 2-by-2 matrix of random numbers.
- X = ones(1,3) or X = zeros (2,3)
- X = rand(size(X));

Array Indexing – Single element

- Indexing is used to access selected elements of an array
- A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16];

- A(row,col)

- Single subscript traverses down each column in order – **A(4,2) = A(8)**

- What is the output for A(4,5)?

- **You can specify elements outside the current dimensions.**
- **The size of the array increases to accommodate the newcomers.**

Array Indexing – Multiple elements

- To refer to multiple elements of an array, colon operator is used, which allows you to specify a range of the form **start** : **end**.

- Try : A(1**:**3,2)

- Try : A(3,**:**)

- Try : A(2,2:**end**)

- Example : Extract the Third, Sixth, and Eight elements from A.

Indexing Vectors

- A= [16 5 9 4 2 11 7 14];

- Swap the two halves of v to make a new vector:
- A2 = A([5:8 1:4])

- Extracting portions of an array: A(2:**end-1**)

- A(1:2:end)   % Extract all the odd elements

- A(end:-1:1)   % Reverse the order of elements

Indexing Vectors

- A= [16 5 9 4 2 11 7 14];
- A([2 3 4]) = [10 15 20] % Replace some elements of A
- A([2 3]) = 30 % Replace second and third elements by 30

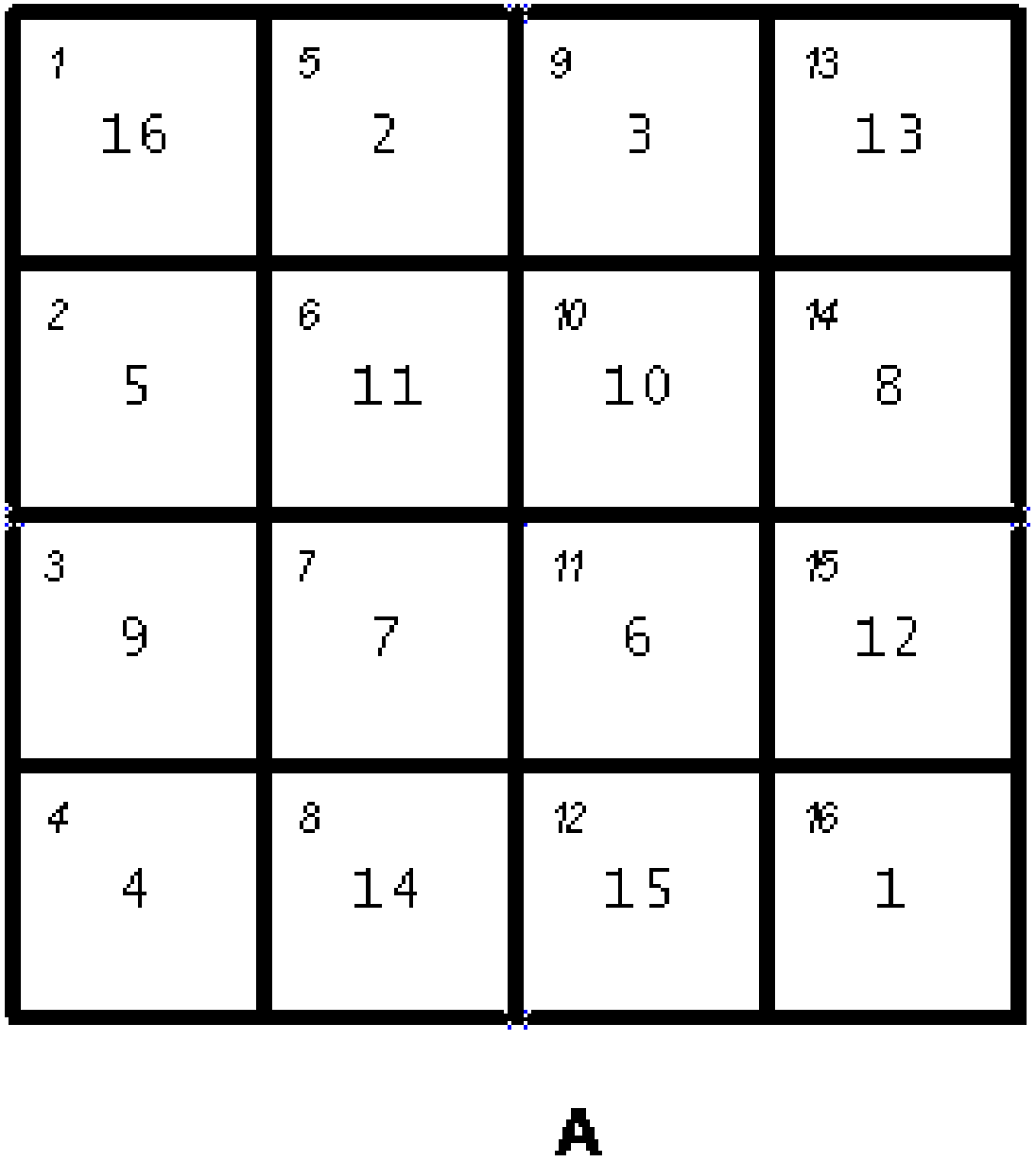
Indexing Matrices with Two Subscripts

- A(row_x, col_y) - Extract the element in row_x, column_y
- A(row_n,:) – Extract nth row
- A(:,end) – Extract last column
- A(2:4,1:2)

Linear Indexing

- When you index into the matrix A using only one subscript,
- MATLAB treats A as if its elements were strung out in a long column vector, by going down the columns consecutively.

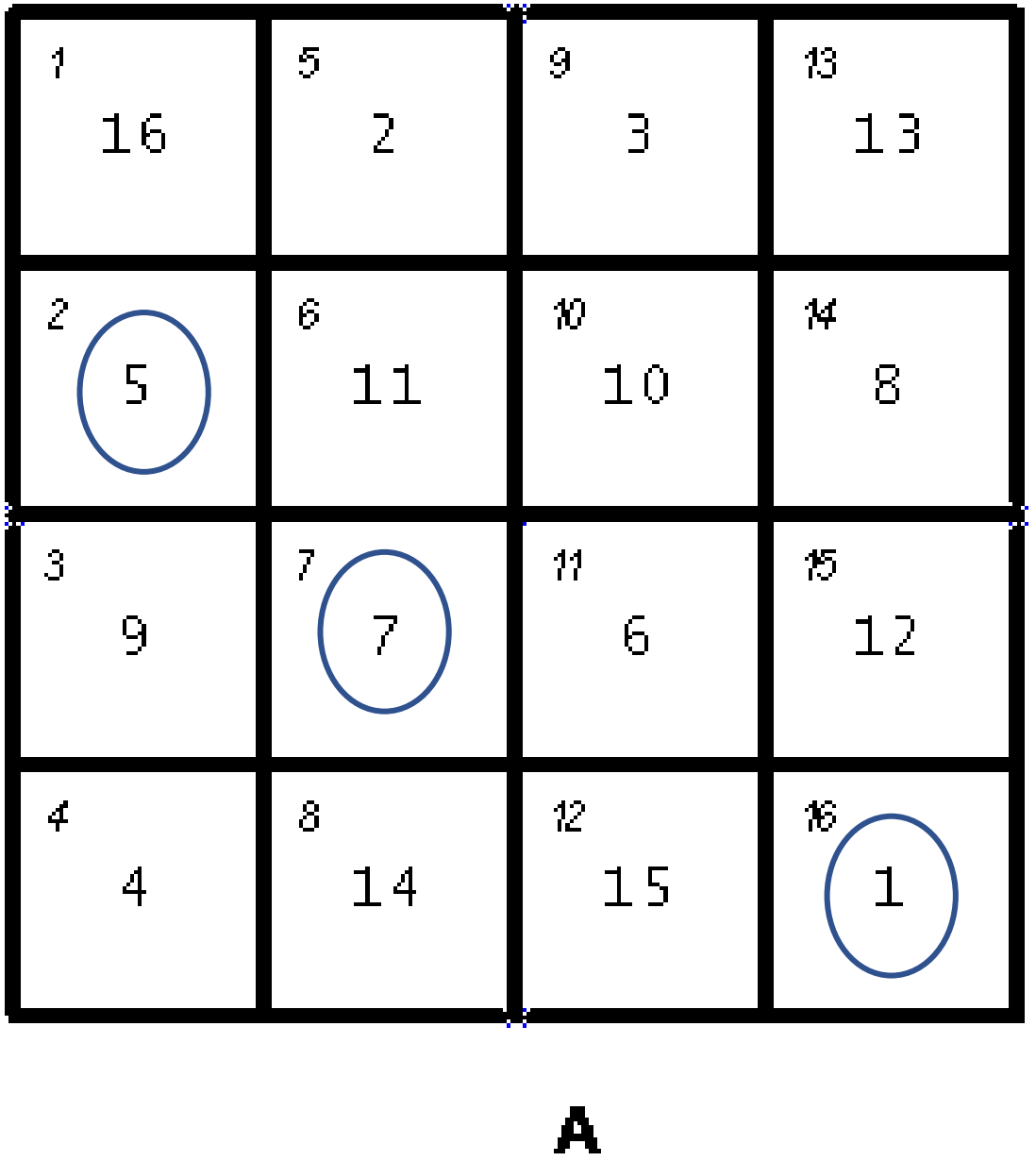
- **A(14) = A(2,4)**
- **A([6 12 15]) = 11 15 12**



Linear Indexing

- **sub2ind** that converts from row and column subscripts to linear indices.

- idx = sub2ind(size(A), [2 3 4], [1 2 4])
- ans =
- 2 7 16
- A(idx)
- ans =
- 5 7 1



A

Logical Indexing

- A(A > 12) extracts all the elements of A that are greater 12 into a logical array (**column vector**).

- Many MATLAB functions that start with **is** return logical arrays and are very useful for logical indexing.

- **B(isnan(B)) = 0** - To replace all NaN elements of the matrix B with zero

- **Strvalue(isspace(Strvalue)) = '_'** - replace all the spaces in a string matrix Strvalue with underscores.

Find function

- Logical indexing is closely related to the **find** function.
- **FIND** returns a vector containing the linear indices of **each nonzero element** in **array** X.
- A(A > 5) is equivalent to A(find(A > 5))
- If X is a vector, then **find** returns a vector with the same orientation as X.
- If X is a multidimensional array, then **find** returns a **column vector** of the **linear indices** of the result.

Find function

- X =
- 16 2 3 13
- 5 11 10 8
- 9 7 6 12
- 4 14 15 1

- k = find(X<10,5) ; Find the first five elements that are less than 10
- X(k) ; View the corresponding elements of X.

- SEE - **ind2sub** - Convert linear indices to subscripts

Changing Values in Arrays

- A(1,4)=0.5
- A(1,end)=0.2

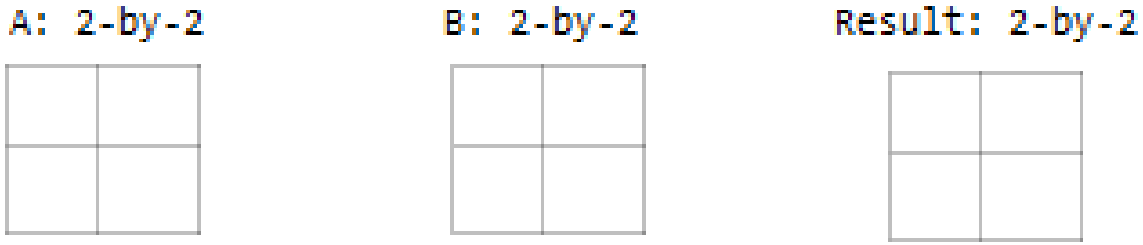
Performing Array Operations on Vectors

- Adding a **scalar value** to all the **elements** of an **array**.
- **Add/Sub** any two arrays of the same size.
- **Multiply** or **divide** all the elements of an array by a scalar.
- Basic statistical functions in MATLAB can be applied to a vector to produce a single output. **Max**, **Min**, **Sum, Sqrt**
- .* operator performs elementwise multiplication and allows to multiply the corresponding elements of two equally sized arrays.
- Ex : z = [3 4] .* [10 20] = ?
- Try : x = [1 2;3 4;5 6; 7 8].*[1;2;3;4]

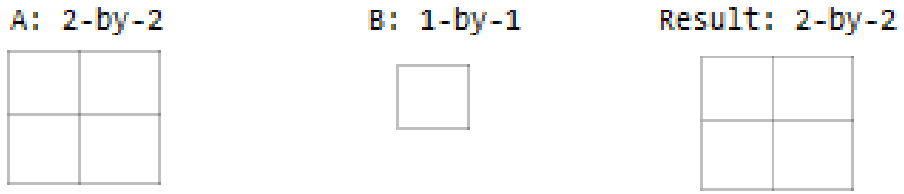
Compatible Array Sizes for Basic Operations

- Two inputs have compatible sizes if, for every dimension, the dimension sizes of the inputs are either the same or one of them is 1.
- In the simplest cases, two array sizes are compatible if they are the same or if one is a scalar.

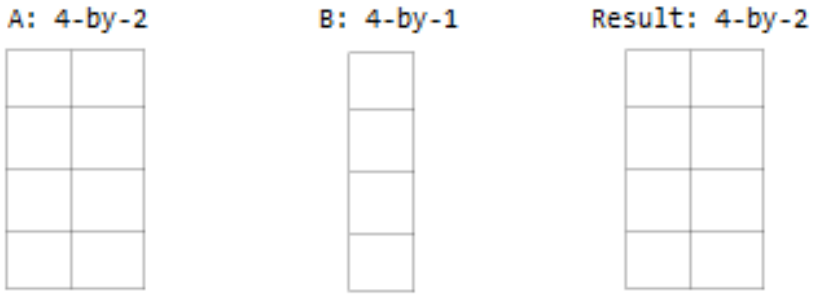
Two inputs which are the same size



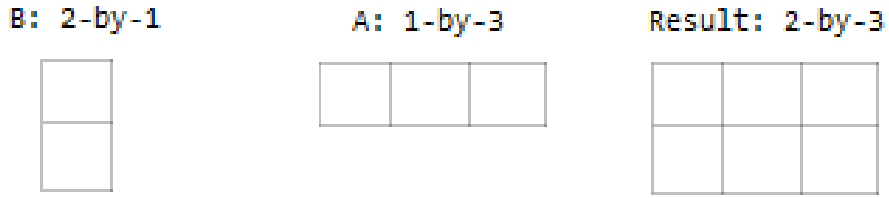
One input is a scalar



Matrix and column vector with the same number of rows.



Column vector, Row vector



Multidimensional Arrays

- Creating Multidimensional Arrays
- >> A = [1 2 3; 4 5 6; 7 8 9];
- Add a second page : **A(:,:,2) = [10 11 12; 13 14 15; 16 17 18]**

- B = cat(**3**,A,[3 2 1; 0 9 8; 5 3 7])
- The **cat** function can be a useful tool for building multidimensional arrays.

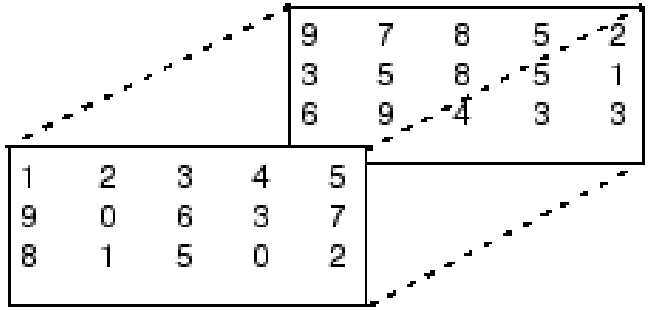
- Expand a multidimensional array >> B(:,:,4) = 0

Accessing Elements

- Try : C = A(**:** ,**[1 3]**, **:**)
  - Use the index vector [1 3] in the second dimension to access only the first and last columns of each page of A.

- Try : D = A(**2:3, :, :**)
  - To find the second and third rows of each page, use the colon operator to create your index vector.

Manipulating Arrays - Reshape



B = **reshape**(A,[6 5])

Use the reshape function to rearrange the elements of the 3-D array into a 6-by-5 matrix.

B = 6×5

1	3	5	7	5
9	6	7	5	5
8	5	2	9	3
2	4	9	8	2
0	3	3	8	1
1	0	6	4	3

reshape operates columnwise, creating the new matrix by taking consecutive elements down each column of A, starting with the first page then moving to the second page.

Manipulating Arrays

- P1 = permute(M,[2 1 3]) - interchange row and column subscripts on each page

- P2 = permute(M,[3 2 1]) - interchange row and page subscripts of M.

- M(:,:,1) = [1 2 3; 4 5 6; 7 8 9];
- M(:,:,2) = [0 5 4; 2 7 6; 9 3 1];

Repeat copies of array

- Create a 3-by-2 matrix whose elements contain the value 10.

- >> A = repmat(10,3,2)

- A = 3×2

- 10 10
- 10 10
- 10 10

Repeat copies of array

- **Example :**
- Try : A = diag([100 200 300]);

- Try : B = repmat(A,2,3);

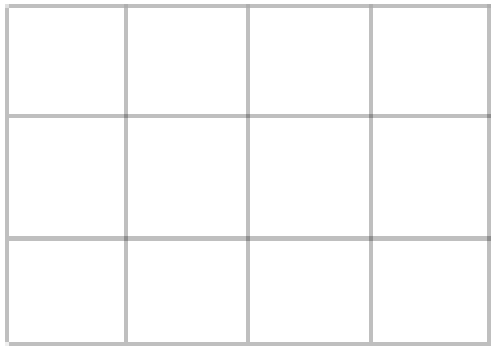
- **Example :**
- Try : A = [1 2; 3 4];

- Try : B = repmat(A,[2 3 2]);

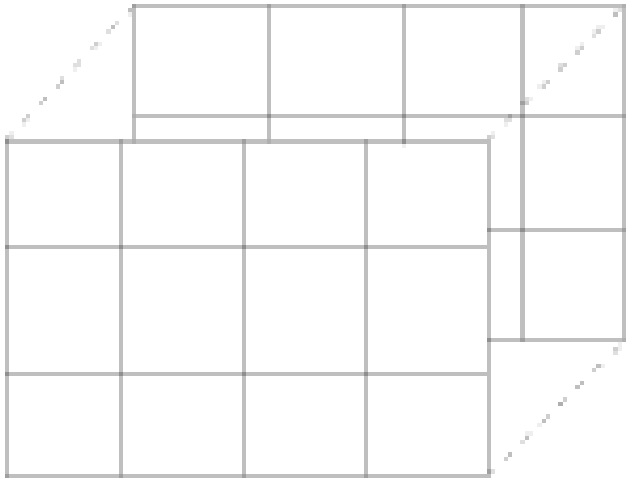
Compatible Array Sizes for Basic Operations

- One input is a matrix, and the other is a 3-D array with the same number of rows and columns.

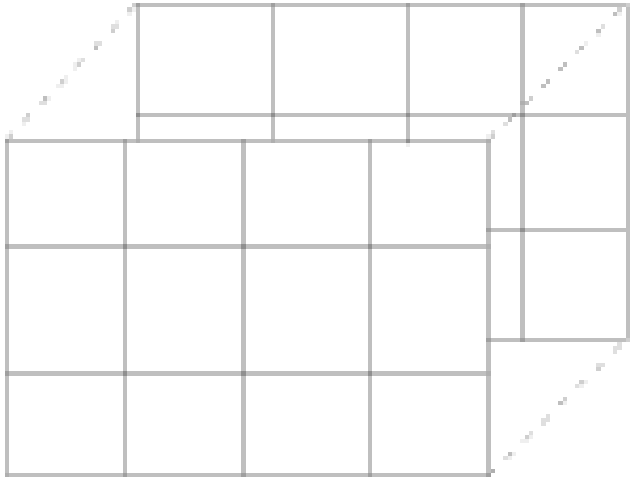
A: 3-by-4



B: 3-by-4-by-2

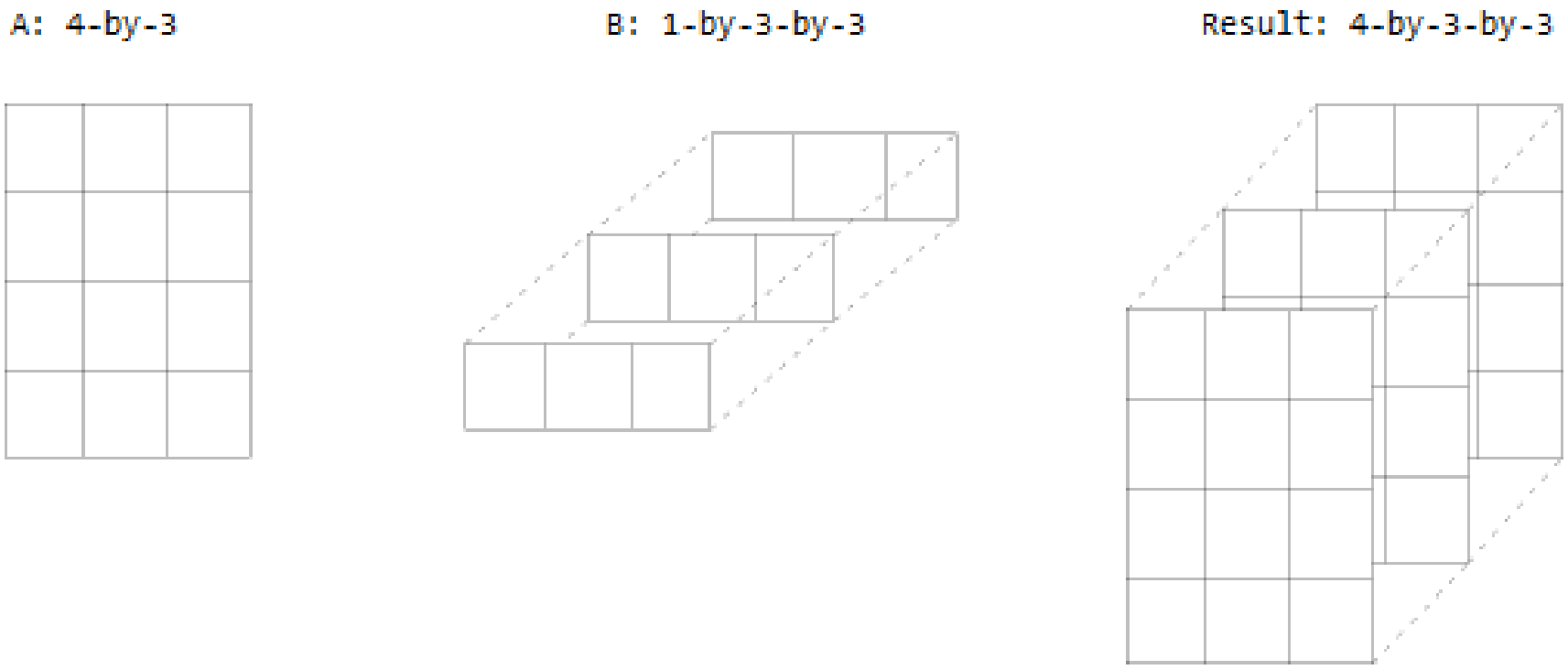


Result: 3-by-4-by-2



Compatible Array Sizes for Basic Operations

- One input is a matrix, and the other is a 3-D array. The dimensions are all either the same or one of them is 1.
-



Cell Array

- A cell array is a data type with **indexed data containers** called **cells**, where each cell can contain any type of data.

- **Cell** arrays commonly contain either **text**, **combinations of text and numbers**, or **numeric arrays** of different sizes.

- C = {'2017-08-16', [56 67 78] }
- C(1,3) = {'Hello world'} - Adding new element
- C(2,:) = {'2017-08-17',[58 69 79],'dd'}; - Adding a new row.

Table Array

- Table array with named variables that can contain different types.
  - column-oriented data in variables
- LastName = {'Sanchez';'Johnson';'Li';'Diaz';'Brown'};
- Age = [38;43;38;40;49];
- Employed = logical([1;0;1;0;1]);
- Height = [71;69;64;67;64];
- Weight = [176;163;131;133;119];
- BloodPressure = [124 93; 109 77; 125 83; 117 75; 122 80];

- **T = table(LastName,Age,Employed,Height,Weight,BloodPressure);**

Table Array

- meanHeight = mean(T.Height); **- Performing calculations**

- T.BMI = (T.Weight*0.453592)./(T.Height*0.0254).^2 **– Adding a new column BMI**

- T.Properties.Description = 'Patient data, including body mass index (BMI) calculated using Height and Weight';

Structure Array

- A structure array is a data type that groups related data using data containers called fields.
  - Each field can contain any type of data.

- Access data in a field using dot notation of the form **structName.fieldName**.

- Rectangle.L = 10;
- Rectangle.W = 5;
- Rectangle.Area = Rectangle.L * Rectangle.Area

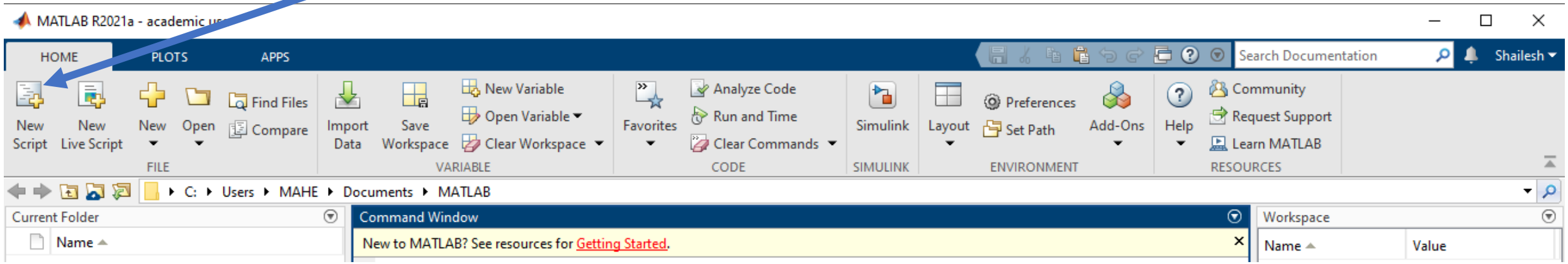
- **s = struct(obj)** creates a scalar structure with field names and values that correspond to properties of obj.

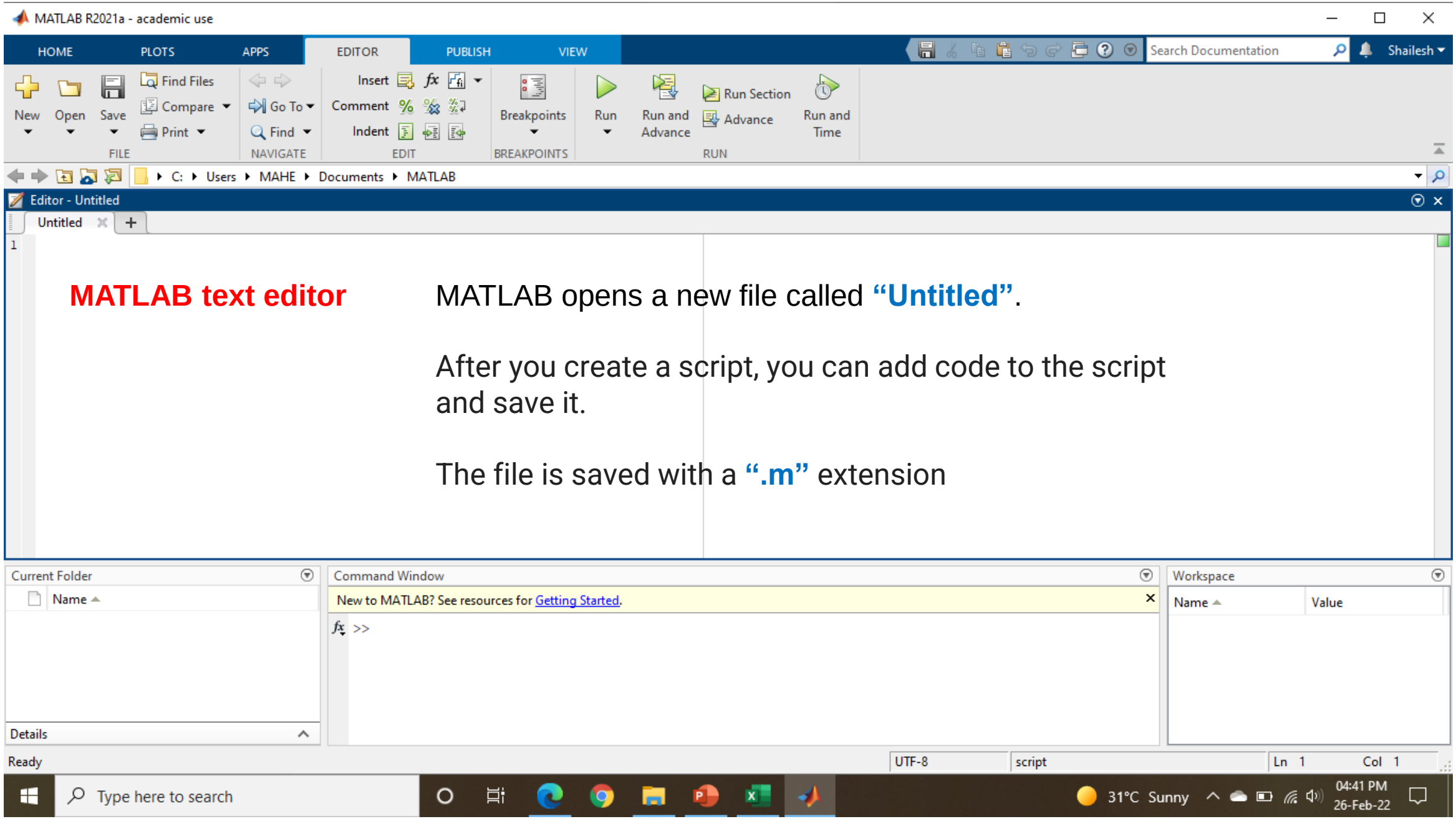
Scripts and Functions

Programming in MATLAB

Scripts (.m files)

- **Scripts** are program (code) files.
- Scripts contain a series of **sequential MATLAB statements and function calls**.
- In command window type >> **edit**
- Or click on "**New Script**" on MATLAB toolbar





MATLAB text editor

MATLAB opens a new file called **"Untitled"**.

After you create a script, you can add code to the script and save it.

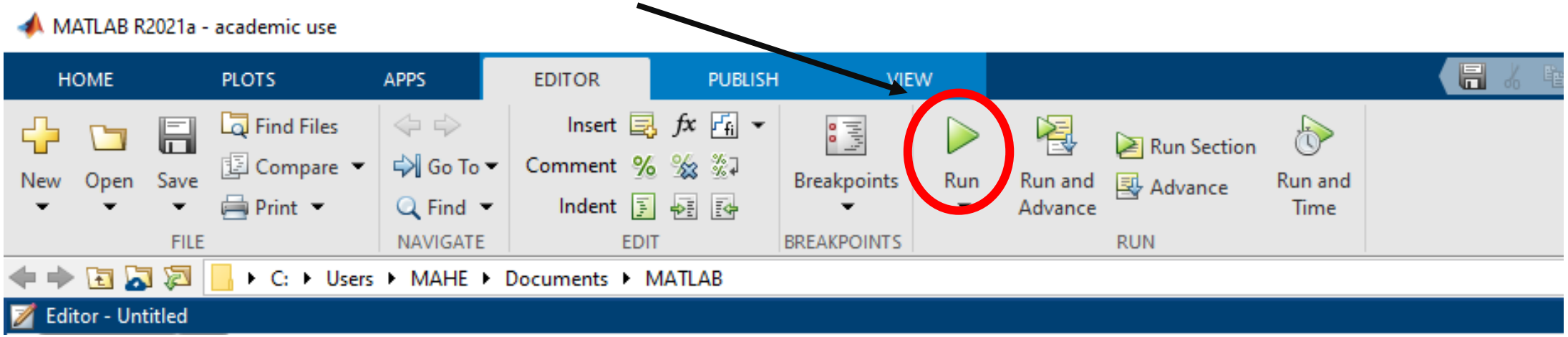
The file is saved with a **".m"** extension

Running the script

- Save your script and run the code using either of these methods:

- Type the script name on the **command line** and press **Enter**.
- For example,
  - to run the **numGenerator.m** script, type >> **numGenerator**

- On the **Editor** tab, click the **Run** button.



Comments in MATLAB

- Comments allow **others** to **understand** code and one can **refresh** their memory when they **return to it later**.

- During code **development** and testing, one can use **comments** to **comment** out **any code** that **does not need to run**.

- To add comments to MATLAB code, use the percent (**%**) symbol.
  - Comment lines can appear anywhere in a code file, and one can append comments to the end of a line of code.

Comments in MATLAB

- To comment out multiple lines of code, use the block comment operators, %{ and %}.
- The %{ and %} operators must appear alone on the lines that immediately precede and follow the block of help text.
- Do not include any other text on these lines.

Example code :

```
a = magic(3);

%{
sum(a)
diag(a)
sum(diag(a))
%}

sum(diag(fliplr(a)))
```

Effective Use of Script Files

- Follow the MA TLAB convention for **naming variables**.
- Avoid giving **script files** the **same name** as a **variable** it computes.
- Do not give a **script file** the **same name** as a MATLAB **command** or **function**.
- Use **exist** command – try >> help **exist**

Debugging Script Files

- **Debugging** a program is the process of **finding** and removing **errors.**
- **Syntax errors** - MATLAB usually detects and displays a message describing the error and its location.
- **Always test the code** with a simple version of the problem, whose **answers** can be checked by **hand calculations**.
- **Display** any **intermediate calculations** by **removing semicolons** at the end of statements.

Programming Style

- **Comments section** – About the input, output variables and user defined functions or any other relevant information.
- **Input section** - About input data and/or the input functions that enable data to be entered.
- **Calculation section –** Functions and algorithms required for calculations in this section.
- **Output section –** Functions necessary to deliver the output in whatever form required.

Input/output commands

- **disp**(A) Displays the contents, but not the name, of the array A.

- **disp**(‘text’) Displays the text string enclosed within single quotes.

- x = **input**(‘text’) Displays the text in quotes, waits for user input from the keyboard, and stores the value in x.

- x = **input**(‘text’,’s’) Displays the text in quotes, waits for user input from the keyboard, and stores the input as a string in x.

Example : Type

```
>> disp(‘The predicted speed is:’)
>> disp(Speed)
```

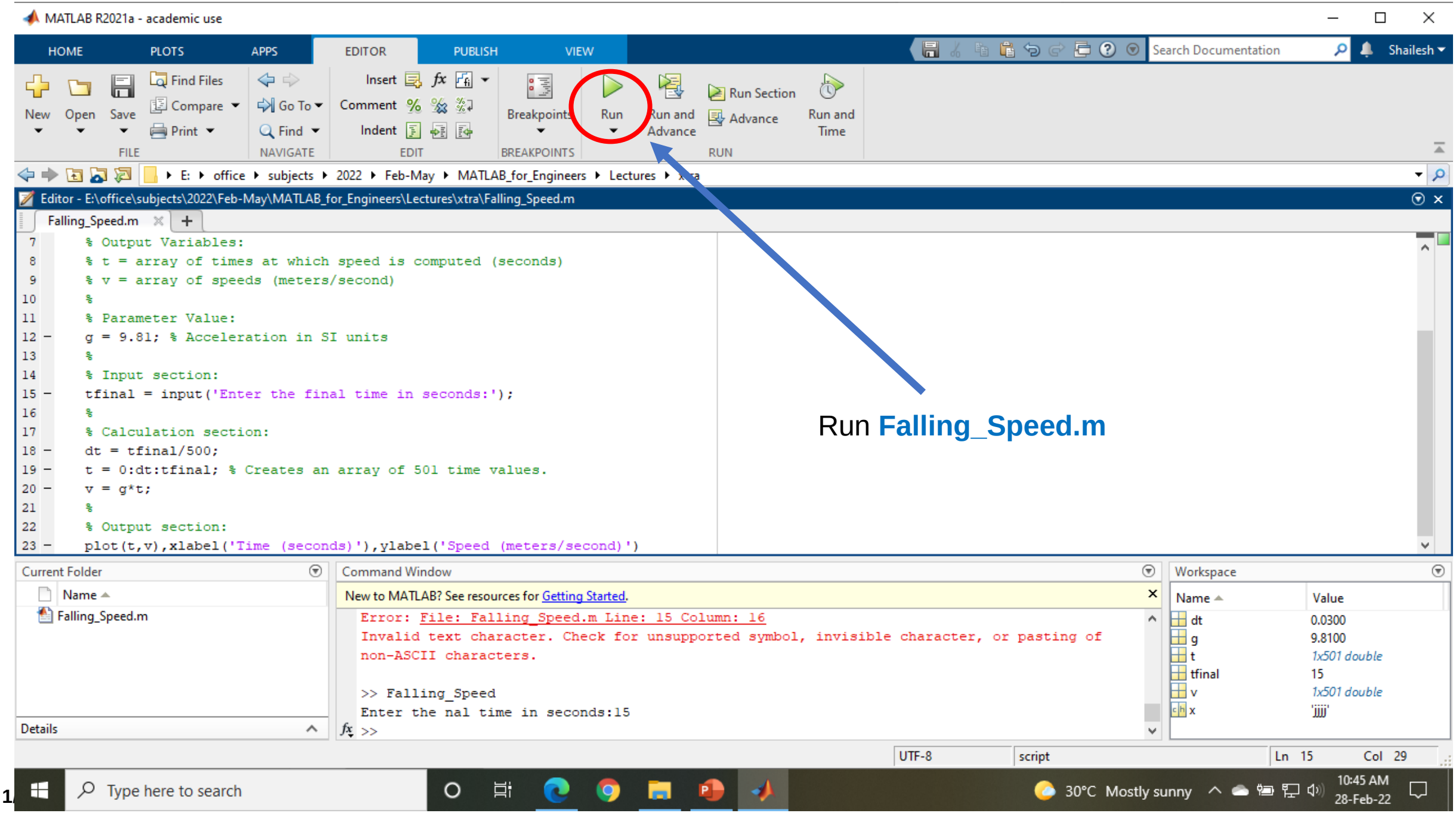
Example of a Script File

- % **Program Falling_Speed.m: plots speed of a falling object.**
- % **Created on** March 1, 2009 by W. Palm III
- %
- % **Input Variable:**
- % tfinal = final time (in seconds)
- %
- % **Output Variables:**
- % t = array of times at which speed is computed (seconds)
- % v = array of speeds (meters/second)
- %
- % **Parameter Value:**
- g = 9.81; % Acceleration in SI units

% **Input section:**
tfinal = input('Enter the final time in seconds:');
%
% **Calculation section:**
dt = tfinal/500;
t = 0:dt:tfinal; % Creates an array of 501 time values.
v = g*t;
%
% **Output section:**
plot(t,v),xlabel('Time (seconds)'),ylabel('Speed (meters/second)')

Save and Run **Falling_Speed.m**

- After creating this file, you save it with the name **Falling_Speed.m**.
- To run it, you type **Falling_Speed** (without the .m) in the Command window at the prompt.
- >> **Falling_Speed**
- You will then be asked to enter a value for **tfinal** .
- After you enter a value and press **Enter**, you will see the plot on the screen.

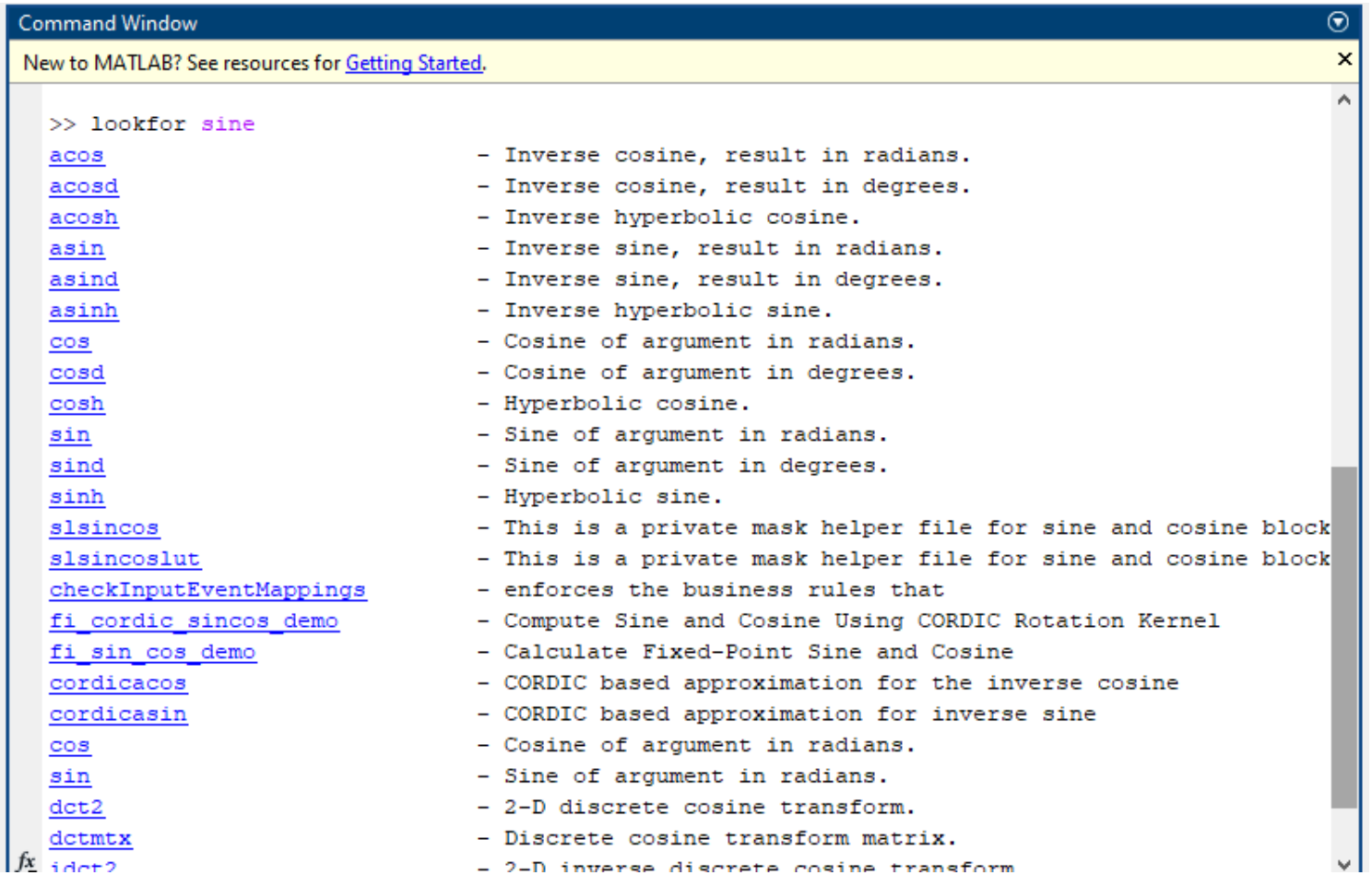

MATLAB R2021a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Shailesh
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Insert
Comment
Indent
EDIT
Breakpoints
BREAKPOINTS
Run
Run and Advance
Run Section
Advance
Run and Time
RUN
E: ▸ office ▸ subjects ▸ 2022 ▸ Feb-May ▸ MATLAB_for_Engineers ▸ Lectures ▸ xtra
Editor - E:\office\subjects\2022\Feb-May\MATLAB_for_Engineers\Lectures\xtra\Falling_Speed.m
Falling_Speed.m
7 % Output Variables:
8 % t = array of times at which speed is computed (seconds)
9 % v = array of speeds (meters/second)
10 %
11 % Parameter Value:
12 g = 9.81; % Acceleration in SI units
13 %
14 % Input section:
15 tfinal = input('Enter the final time in seconds:');
16 %
17 % Calculation section:
18 dt = tfinal/500;
19 t = 0:dt:tfinal; % Creates an array of 501 time values.
20 v = g*t;
21 %
22 % Output section:
23 plot(t,v),xlabel('Time (seconds)'),ylabel('Speed (meters/second)')
Run Falling_Speed.m
Current Folder
Name
Falling_Speed.m
Details
Command Window
New to MATLAB? See resources for Getting Started.
Error: File: Falling_Speed.m Line: 15 Column: 16
Invalid text character. Check for unsupported symbol, invisible character, or pasting of non-ASCII characters.
>> Falling_Speed
Enter the nal time in seconds:15
>>
Workspace
Name
Value
dt 0.0300
g 9.8100
t 1x501 double
tfinal 15
v 1x501 double
x 'jjjjj'
UTF-8
script
Ln 15
Col 29
Type here to search
30°C Mostly sunny
10:45 AM
28-Feb-22

The lookfor Function

- **lookfor** - Search all MATLAB files for keyword.

- Type >> help sine.m
- >> sine.m not found

- Type >> lookfor sine



Problem-Solving Methodologies

- Steps in Engineering Problem Solving –
- 1. **Understand** the purpose of the **problem**.

- 2. **Collect** the known and relevant **information - INPUT**.
  - Realize that some of it might later be found **unnecessary**.

- 3. **Determine** what **information** you must **find - OUTPUT**.

- 4. **Simplify** the **problem** only enough to obtain the required information. State any **assumptions** you make.

- 5. Draw a **sketch** and **label** any necessary **variables**.

Problem-Solving Methodologies

- Steps in Engineering Problem Solving –

- 6. Determine which **fundamental principles (Example : Newton's laws)** are applicable.

- 7. Think generally about your **proposed solution** approach and consider **other approaches** before proceeding with the details.

- 8. Label each step in the solution process.

Problem-Solving Methodologies

- Steps in Engineering Problem Solving –

- 9. If you solve the problem with a program, **hand check the results using a simple version of the problem**. Print the results of intermediate steps.

- 10. Perform a reality check and precision check on the answer.
  - For example : Height cannot be negative or large.

Function files

- **Functions** − functions files are also program files with .m extension.

- Functions can accept inputs and return outputs.

- Internal variables are local to the function.

function **f** = fact(**n**)

End

Save as **fact.m**

Function call

>> X = fact(Y)

Syntax for Function Definition

- **Function name** - Valid function names follow the same rules as **variable names**.
  - They must start with a **letter**, and can **contain letters**, **digits**, or **underscores**.

- function y = myFunction(one,two,three) - Input arguments

- function [one, two, three] = myFunction(x) - Output arguments

- If there is no output, you can omit it.
  - function myFunction(x) or function [] = myFunction(x)

Live script editor

- Watch the demo.

Classes and File operations

Programming in MATLAB

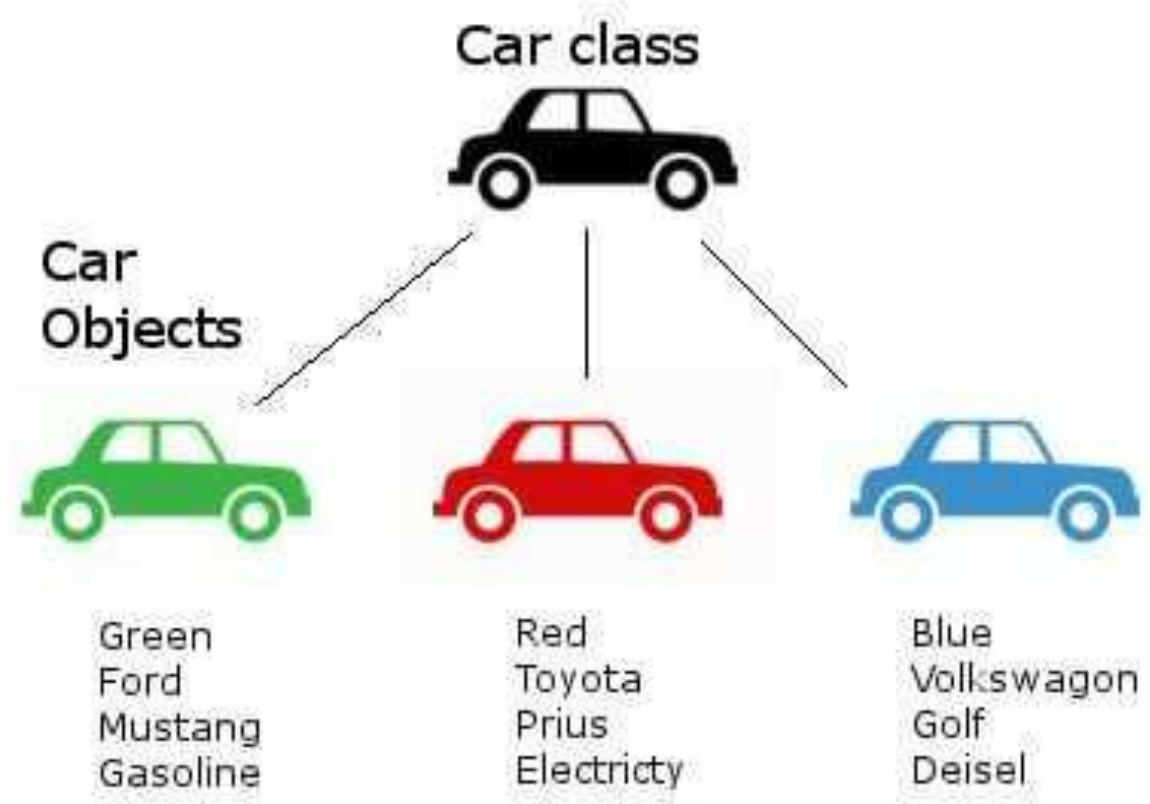
Classes

- Creating new type of **objects** using concepts of object-oriented programming (OOPs).
- Software = Data + Operations on Data.
- **Procedural programs** pass **data** to **functions**, which perform the necessary **operations** on the **data**.
- In **OOPs**, the **data** and the **functions** that operate on them are **bound together** so that **no other part of the code** can **access** this data **except** that **function**.

Classes and Objects

- **Class** is the **blueprint** of an **object**.
  - It is used to declare and create objects.

- A **class** describes a set of **objects** with **common characteristics**. **Objects** are **specific instances** of a **class**.



Creating a Simple Class

```
classdef BasicClass
  properties
    Value {mustBeNumeric}
  end
  methods
    function r = roundOff(obj)
      r = round([obj.Value],2);
    end
    function r = multiplyBy(obj,n)
      r = [obj.Value] * n;
    end
  end
end
```

Value — Property that contains the numeric data stored in an object of the class

Try >> help mustBeNumeric
Try >> lookfor('mustbe')

roundOff — Method that rounds the value of the property to two decimal places

multiplyBy — Method that multiplies the value of the property by the specified number

classdef is a keyword used to define MATLAB classes.

Using a class

- To use the class:
- Save the class definition in a .m file with the same name as the class.

- Create an object of the class.

  a = BasicClass

- Access the properties to assign data.

  a.Value = pi/3;

- Call methods to perform operation on the data. (dot operator)

  a.roundOff(a)
  a.multiplyBy(a,3)

Constructor

- Constructor is a special method to create objects of a class.
- It has the same as that of a class.
- Pass property values as arguments to the constructor

```
methods
    function obj = BasicClass(val)
        if nargin == 1
            obj.Value = val;
        end
    end
End
```

nargin - Number of input arguments to the function

Example : a = BasicClass(pi/3)

BasicClass Code

BasicClass definition after adding the constructor.

```
classdef BasicClass
    properties
        Value {mustBeNumeric}
    end
    methods
        function obj = BasicClass(val)
            if nargin == 1
                obj.Value = val;
            end
        end
        function r = roundOff(obj)
            r = round([obj.Value],2);
        end
        function r = multiplyBy(obj,n)
            r = [obj.Value] * n;
        end
    end
end
```

File operations

Find, view, and change files and folders

File operations

- To list folder contents : type **>> dir** or **>> ls**
- >> list = **ls**(‘*my*’) - List all the files and folders with names that contain my.
- >> list = **ls**(‘*.m’) List all the files and folders with a .m extension.
- >> **pwd** - Identify current folder
- >> **isfile**(fileName) - returns 1 if fileName is a file located on the specified path or in the current folder. Otherwise, isfile returns 0.

Create, Change, and Delete Files and Folders

- **cd** - Change current folder
- **copyfile** - Copy file or folder (returns status logical 1/0)
- **delete** - Delete files or objects
- **mkdir** - Make new folder (returns status logical 1/0)
-
- **movefile** - Move or rename file or folder (returns status logical 1/0)
- **rmdir** - Remove folder (returns status logical 1/0)

File Compression

- Zip - Compress files into zip file

- zippedfiles = **zip**('tmwlogo.zip',{'membrane.m','logo.m’})
  - Compress the files **membrane.m** and **logo.m** into a file named **tmwlogo.zip**.

- **zip**('backup',{'*.m','*.mlx'});
  - Compress all **.m** and **.mlx** files in the current folder to the file **backup.zip**.

Compress a Folder

- Create a folder myfolder containing a subfolder mysubfolder and the files membrane.m and logo.m.

- **mkdir** myfolder;
- **movefile**('membrane.m','myfolder');
- **movefile**('logo.m','myfolder');
- **cd** myfolder;
- **mkdir** mysubfolder;
- **cd** ..

- zippedfiles = **zip**('myfiles.zip','myfolder’);
  - Compress the contents of myfolder, including all subfolders.

Compress Files Into Specified Folder

- zip(zipfilename, filenames, rootfolder)
- zip(**‘../backup.zip'**,**{‘notes.doc’,’tutorial.ppt'}**,**’d:/Subjects’**);
- The files **notes.doc** and **tutorial.ppt** located in the folder **d:/Subjects**.
- Compress these files into **backup.zip**, **one level up from the current folder**.

Extract contents of zip file

- filenames **= unzip**(zipfilename)
  - extracts the archived contents of zipfilename into the current folder.
  - unzip can extract files from your **local system** or from an **Internet URL**

- filenames = **unzip**(zipfilename, outputfolder)
- extracts zipfilename into outputfolder. If outputfolder does not exist, MATLAB creates it.

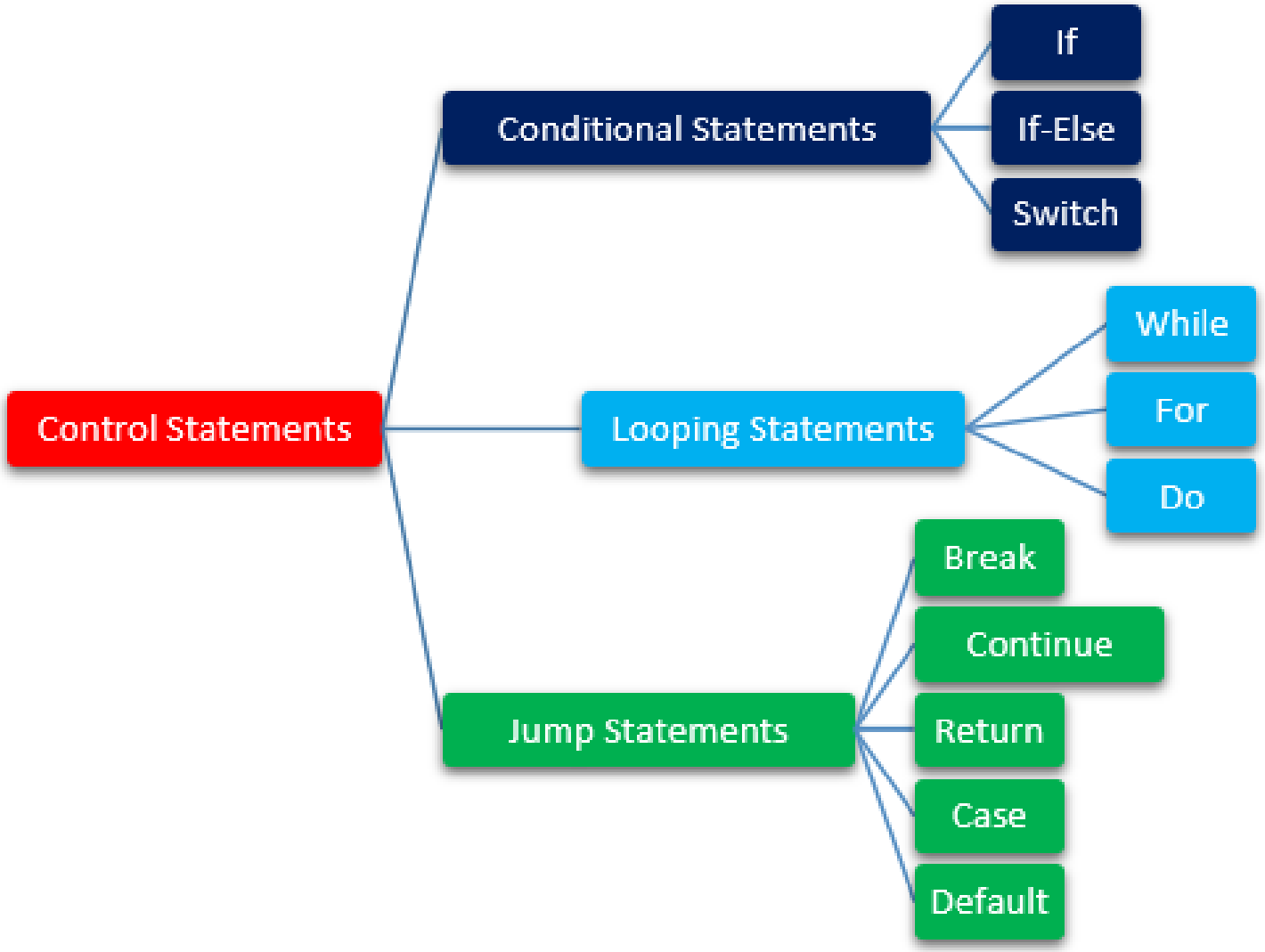
- Download and extract a zip file from a URL to a local folder.
- url = **'http://example.com/example_file.zip'**;
- **unzip**(url, **'example_folder'**);

Loops and Conditional Statements

Control flow and branching using keywords,

such as **if**, **for**, and **while**

Loops and Conditional Statements



Conditional statements with the proper **comparison** and **boolean operators** allow the creation of **alternate execution paths** in the code.

Loops allow **repeated execution** of the **same** set of **statements** on all the objects within a sequence.

Jump statements allow you to **exit a loop**, start the next iteration of a loop, or explicitly **transfer program control** to a specified location in your program.

FOR Loop

for loop to repeat specified number of times

```
for index = values
   statements
end
```

Decrement Values

```
for v = 1.0:-0.2:0.0
   disp(v)
end
```

Execute Statements for Specified Values

```
for v = [1 5 8 17]
   disp(v)
end
```

Repeat Statements for Each Matrix Column

```
for I = eye(4,3)
    disp('Current unit vector:')
    disp(I)
end
```

Assign Matrix Values

```
s = 10;
H = zeros(s);

for c = 1:s
    for r = 1:s
        H(r,c) = 1/(r+c-1);
    end
end
```

Selectively Display Values in Loop : continue

```
for n = 1:50
    if mod(n,7)
        continue
    end
    disp(['Divisible by 7: ' num2str(n)])
end
```

WHILE Loop

while loop to repeat when condition is true

```
while expression
    statements
end
```

Repeat Statements Until Expression Is False

```
n = 10;
f = n;
while n > 1
    n = n-1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Exit Loop Before Expression Is False

```
limit = 0.8;
s = 0;

while 1
    tmp = rand;
    if tmp > limit
        break
    end
    s = s + tmp;
end
```

Conditional statements

Conditional statements enable you to select at run time which block of code to execute.

Some examples :

```
% Generate a random number
a = rand(100, 1);

% If it is even, divide by 2
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end
```

```
a = rand(100, 1);

if a < 30
    disp('small')
elseif a < 80
    disp('medium')
else
    disp('large')
end
```

```
[dayNum, dayString] = weekday(date, 'long', 'en_US');

switch dayString
   case 'Monday'
      disp('Start of the work week')
   case 'Tuesday'
      disp('Day 2')
   case 'Wednesday'
      disp('Day 3')
   case 'Thursday'
      disp('Day 4')
   case 'Friday'
      disp('Last day of the work week')
   otherwise
      disp('Weekend!')
end
```

return

return the control to the invoking program before it reaches the end of the script or function.

Return Control to Keyboard

```
function idx = findSqrRootIndex(target,arrayToSearch)
idx = NaN;
if target < 0
   return
end
for idx = 1:length(arrayToSearch)
    if arrayToSearch(idx) == sqrt(target)
        return
    end
end
```

```
>>A = [3 7 28 14 42 9 0];
>>b = 81;
>> findSqrRootIndex(b,A)
```

Return Control to Invoking Function

```
function returnControlExample(target)
    arrayToSearch = [3 7 28 14 42 9 0];
    idx = findSqrRootIndex(target,arrayToSearch);

    if isnan(idx)
        disp('Square root not found.')
    else
        disp(['Square root found at index ' num2str(idx)])
    end
end
```

```
>> returnControlExample(49)
>> Square root found at index 2
```

Time complexity

Iterative and recursive algorithms

How to measure running time of an algorithm?

- Experimental study –
  - **Implement** the algorithm in a **programming language**
  - **Run** it with **different input sets**
  - Use **system time** (clock() function, time(), data()) to **measure** actual running **time**.

- Drawbacks –
  - **Implementation** of algorithm in a **preferred language** – time required
  - Only **finite input sets can be verified** – Not all input sizes are considered
  - For **comparing** two **algorithms same hardware** and **software** is required

Algorithm analysis

- Use **high-level description** of the algorithm **instead** of **testing its implementations.**
- Consider **all possible inputs**
- **Analysis** algorithm running time **irrespective** of **software and hardware requirements**.

Pseudo-code

- **Pseudocode** is an informal high-level description of the operating principle of a computer program or other algorithm.
- It uses the **structural conventions** of a normal programming language, but is intended for **human reading** rather than machine reading.
- Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines.
- No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program.

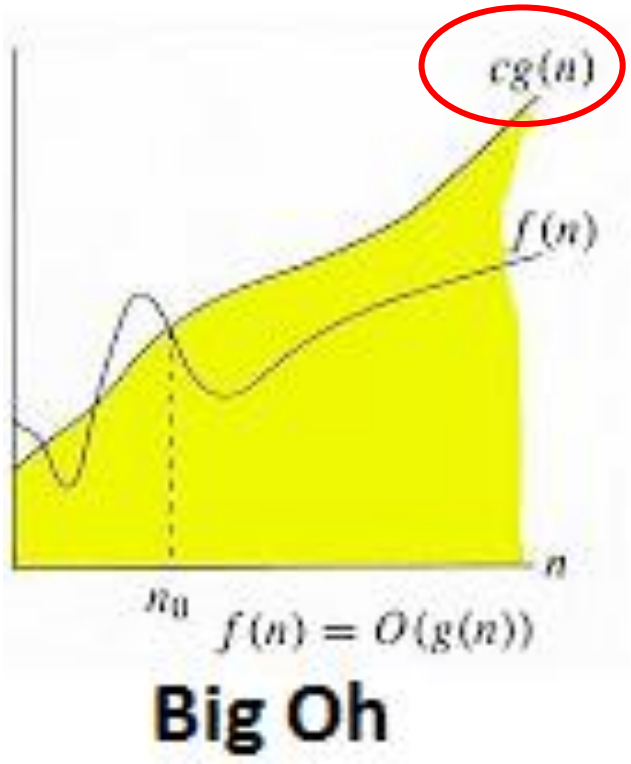
Pseudo-code

- Decision structures – **If .. Else .. End**
- While loop – **While …. End**
- For loop – **For … End**
- Array indexing – **A[i] .. A[i, j]**
- Methods – **methodname(Arguments)**

Time complexity – Big Oh notation

- Total **time** required by the **program** to run till its **completion**.
- It is estimated by **counting** the **number** of **elementary** steps **performed** by any **algorithm(code)** to **finish execution**.
- **Algorithm's** (**code**) **performance** varies with different types of input data.
- Usually, the **worst-case time complexity** of an algorithm is of interest.
  - **Maximum time** taken for any input size.

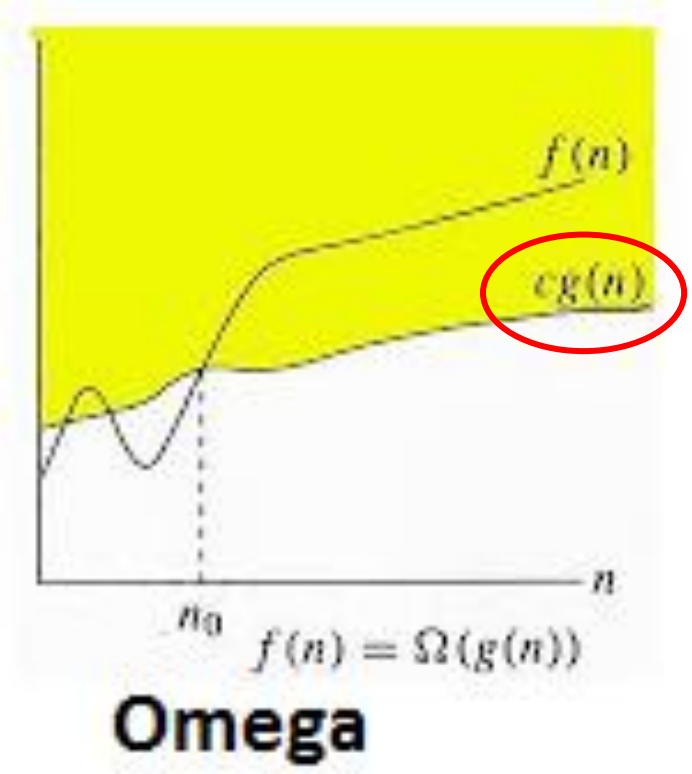
WORST CASE



Big Oh

Worst case : Input which takes long time or algorithm is slower

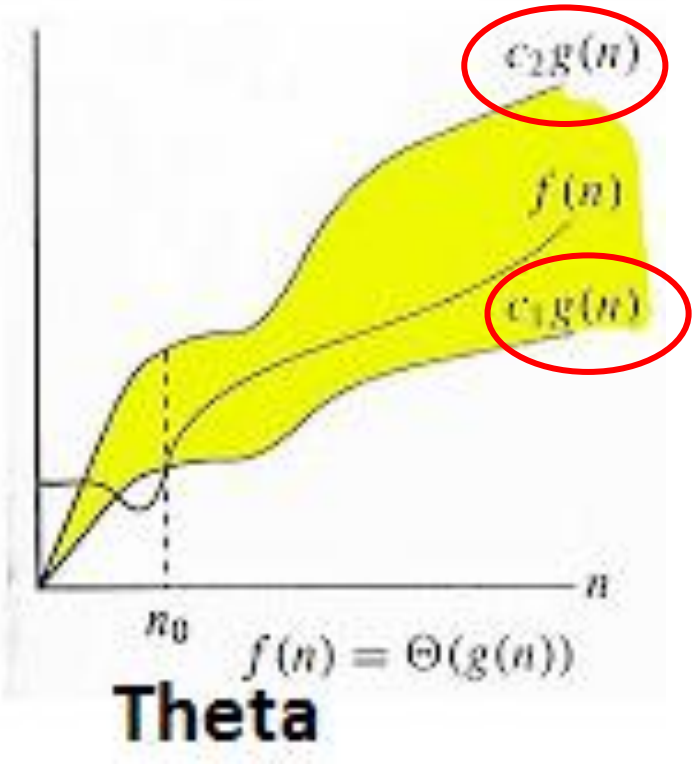
BEST CASE



Omega

Best case : Input for which algorithm takes lowest time or algorithm is faster

AVERAGE CASE



Theta

Average case : Predicts running time of algorithm for a random input

Order of growth

- The rate at which the running time increases as a function of input is called "**Order of growth**"

	constant	*logarithmic*	*linear*	*N-log-N*	*quadratic*	*cubic*	*exponential*
n	$O(1)$	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84×10^{19}

Question

- Write an algorithm to find a factorial of a number.
- Express its running time in terms of input size.

Iterative and recursive algorithms

Iterative

- Factorial
- For i=1:n
  - Fact = Fact * i;
- End
- Return Fact

Recursive

- **Factorial** (n)
- If n==0
  - Return 1
- Else
  - Return n***Factorial**(n-1)
- End

Iterative and recursive algorithms

Iterative

- keep repeating until a task is “done”

Recursive

- Solve a large problem by breaking it up into smaller and smaller pieces until you can solve it; combine the results.

Which is Better? **No clear answer, but there are known trade-offs.**

Iterative and recursive algorithms

- Which approach to choose? Depends on the problem

- Algorithms with Abstract Data Types (ex: Trees) can be easily implemented **recursively**.
-
- “Mathematicians” often prefer recursive approach.
  - Solutions often shorter
  - Good recursive solutions may be more difficult to design and test.
-
- “Programmers”, often prefer iterative solutions.
  - Easy to implement
  - Control stays local to loop

Master theorem for subtract and conquer recurrences

- If the recurrence is of the form $T(n) = aT(n-b) + O(n^K)$
- $T(n) = O(n^K)$ if $a<1$
- $T(n) = O(n^{K+1})$ if $a=1$
- $T(n) = O(n^K a^{n/b})$, if $a>1$

Recursive linear search

- Search(A,i,x)
- If A[i] == x
  - Return i
- Else
  - Return Search(A,i+1,x)
- End

- Time complexity – O(n)

Solution :
$T(n) = O(1) + T(n-1)$
$= O(n^0) + T(n-1)$

a=1 K=0

$T(n) = O(n^{K+1})$ if a=1
$=O(n)$

Question

- What is the time complexity of the following code?
- Function(n)
- If n<= 1
  - Return
- End
- For i=1:3
  - Function(n-1)
- End

Solution :

$T(n) = O(1) + 3T(n-1)$

$= O(n^0) + 3T(n-1)$

a=3 K=0 b=1

$T(n) = O(n^K a^{n/b})$, if a>1

$=O(n^0 3^n) = O(3^n)$

Guidelines for algorithm analysis

- Loops – O(n)
- Nested loops – Total running time is product of sizes of all the loops
- Consecutive statements – Add the time complexities of each statement
- If-then-else : Worst-case running time of either ‘then’ part or the ‘else’ part (whichever is the larger)

Logarithmic complexity

- An algorithm is O(log n) if it takes a constant time to divide the problem size by a fraction.
- Example :
- For i=1:n
  - i=i*2
- End
- Let us assume loop ends after ‘k’ times that is $2^k=n$
- $n = 2^k$ therefore k = logn

Master Theorem for Divide and Conquer

- If the recurrence is of the form
- $T(n) = a\,T(n/b) + O(n^K)$, where $a \geq 1$, $b > 1$, $K \geq 0$ then

- If $a < b^K$, $T(n) = O(n^K)$

- If $a = b^K$, $T(n) = O(n^K \log n)$

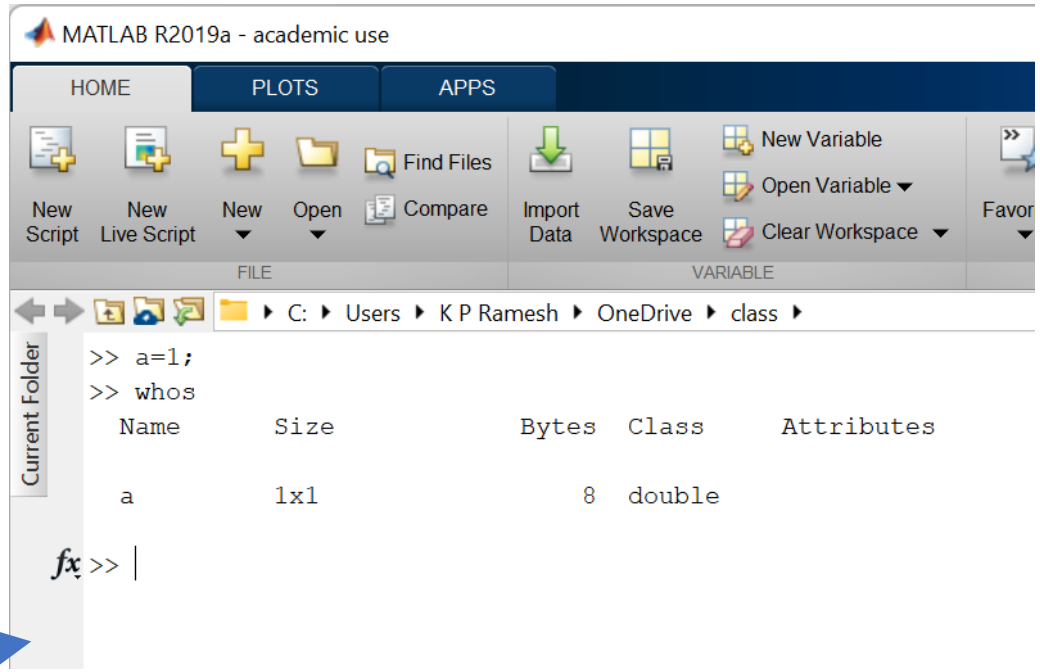
- If $a > b^K$, $T(n) = O(n^{(\log_b a)})$

Data types and operators

Int, float, char, Arithmetic , relational and logical operators

MATLAB Data types

- Data type - what **type** of **value** a **variable** has and what type of mathematical, relational or logical **operations** can be **applied** to it.
- **Matrix** or **Array** is the **primary data type** in MATLAB.
- In MATLAB, all numeric variables are stored as **double-precision floating-point (64 bits)**.
- Type >> a=1;
- >> whos

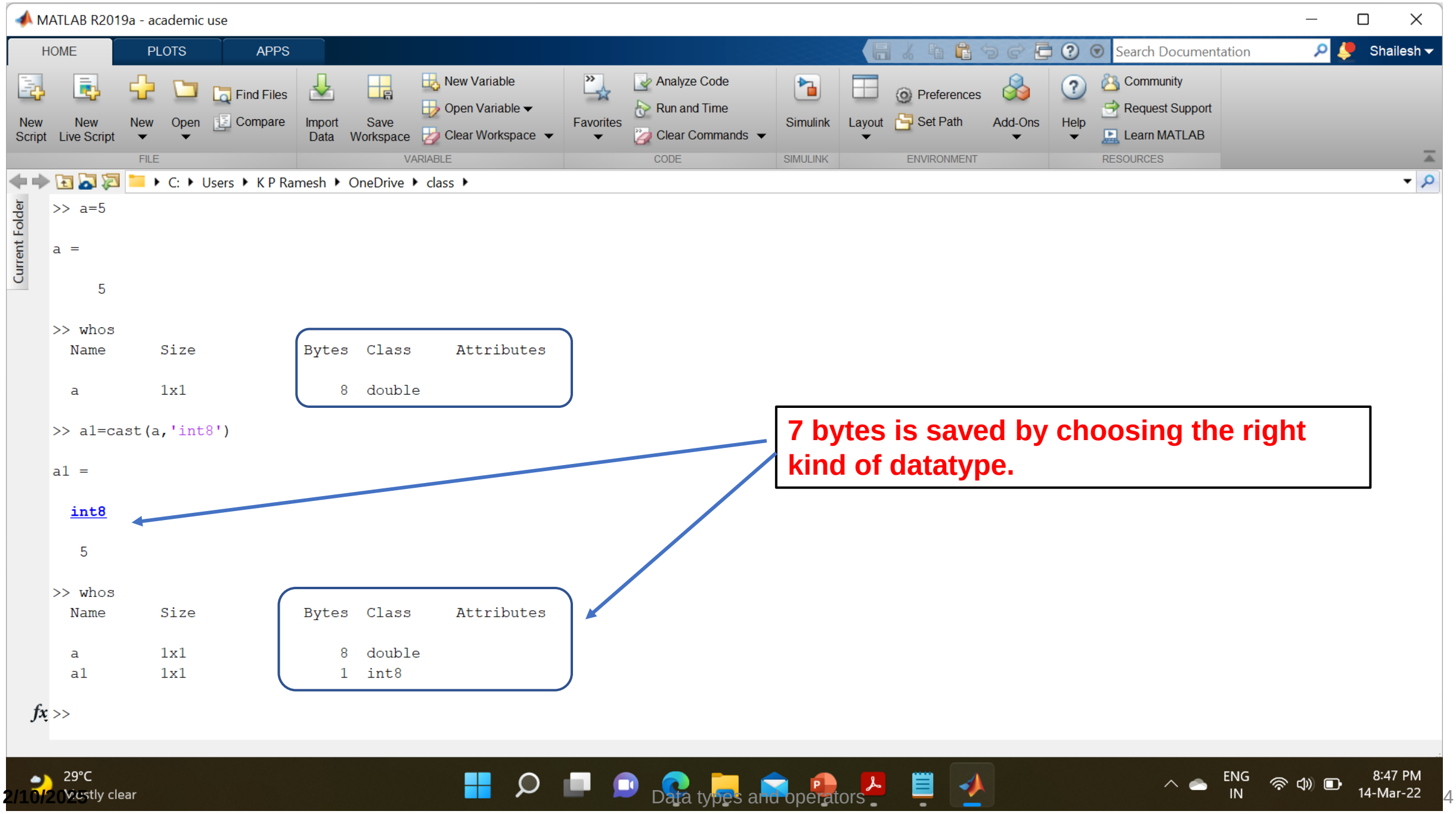


Numeric data types

- **int8** - 8-bit signed integer
- **uint8** - 8-bit unsigned integer
- **int16** - 16-bit signed integer
- **uint16** - 16-bit unsigned integer
- **int32** - 32-bit signed integer
- **uint32** - 32-bit unsigned integer
- **int64** - 64-bit signed integer
- **uint64** - 64-bit unsigned integer
- **single** - single precision numerical data
- **double** - double precision numerical data
- **logical** - 1 or 0

The basic strategy for selecting the best data type is to **select** the **smallest data type** that **matches** the **kind of data** you have and that allows for **all the feasible values of your data**.

```
type >>intmin('int8')
type >>intmax('int8')

type >>intmin('uint8')
type >>intmax('uint8')
```


MATLAB R2019a - academic use
HOME
PLOTS
APPS
Search Documentation
Shailesh
New Script
New Live Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Favorites
Analyze Code
Run and Time
Clear Commands
Simulink
Layout
Preferences
Set Path
Add-Ons
Help
Community
Request Support
Learn MATLAB
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
C: ▸ Users ▸ K P Ramesh ▸ OneDrive ▸ class ▸
Current Folder
>> a=5
a =
5
>> whos
Name Size Bytes Class Attributes
a 1x1 8 double
>> a1=cast(a,'int8')
a1 =
int8
5
>> whos
Name Size Bytes Class Attributes
a 1x1 8 double
a1 1x1 1 int8
>>
7 bytes is saved by choosing the right kind of datatype.
29°C
Mostly clear
ENG
IN
8:47 PM
14-Mar-22

Characters and strings

- String scalar is created by enclosing a piece of text in double quotes.
- >> **Str=“Hello world!”;**
- >> **Str1=["Hello", "World"]** - string array, by concatenated string scalars using square brackets

- Character vector is created using single quotation marks.
- >> **C = 'Hello, world’**
- >> **B =char('MIT','Manipal’)**
- >> **C1 = char(Str);** - string scalar to char array

Concatenation / Joining strings

- >> **strcat**(‘MIT’, ‘Manipal’) - **'MITManipal’**
  - Concatenate strings horizontally

- >> str = ["Carlos","Sada";
- "Ella","Olsen";
- "Diana","Lee"]

SPLIT :
newStr = **split**(str) - divides str at whitespace character
newStr = **split**(str,delimiter) -divides each element of str at the delimiters

- >> newStr = **join**(str,"--");
  - combines the text in str and places the **elements of delimiter** between the elements of str instead of a space character.

Convert Between Numeric and Strings

- Convert numbers to character array
- >>s = **num2str**(pi);

- Convert character array or string to numeric array
- >> X = **str2num**('100 200 300 400’);

Find

- str = ["Mary Ann Jones","Paul Jay Burns","John Paul Smith"];
- pat = "Paul";
- TF = **contains**(str,pat); % returns a logical array

- str = "paired with red shoes";
- A = **count**(str,"red");

- str = 'Find the starting indices of substrings in a character vector';
- k = **strfind**(str,'in')

Replace

- newStr = **replace**(str,old,new) - Find and replace one or more substrings

- newStr = **strrep**(str,old,new) - Find and replace substrings

Extract

- Extract substrings from strings
- newStr = **extract**(str,**pat**) - returns any substrings in str that match the **pattern** specified by **pat**. (Example – digitsPattern, lettersPattern)

- newStr = **extract**(str,**pos**) - returns the character in str at the position specified by **pos**.

extractAfter	Extract substrings after specified positions
extractBefore	Extract substrings before specified positions
extractBetween	Extract substrings between start and end points

Dates and Time

- Arrays of date and time values that can be displayed in different formats.
- **clock** - Current date and time as date vector
- **cputime-** CPU time used by MATLAB expressed in seconds.
- **date** - Current date as character vector
- t = **datetime** - scalar datetime array corresponding to the current date and time.
- t = **datetime**('now','TimeZone','Asia/Seoul','Format','d-MMM-y HH:mm:ss Z')

MATLAB operators

Arithmetic Operators, Relational Operators, Logical Operators

Arithmetic operators

- Addition +
- Subtraction –
- Element-wise multiplication .*
- Matrix multiplication *
- Element-wise power .^

Symbol	Operation	MATLAB form
^	exponentiation: a^b	`a^b`
*	multiplication: ab	`a*b`
/	right division: $a/b = \frac{a}{b}$	`a/b`
\	left division: $a\backslash b = \frac{b}{a}$	`a\b`
+	addition: $a + b$	`a+b`
−	subtraction: $a - b$	`a-b`

^	Exponential Operator	6
*	Multiplication	7
/	Division	7
+	Addition	8
-	Subtraction	8

Order of precedence

Precedence	Operation
First	Parentheses, evaluated starting with the innermost pair.
Second	Exponentiation, evaluated from left to right.
Third	Multiplication and division with equal precedence, evaluated from left to right.
Fourth	Addition and subtraction with equal precedence, evaluated from left to right.

Relational and logical Operators

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

Logical Operators

&&	logical AND
||	logical OR
~	logical NOT

expr1 && expr2
expr1 || expr2

Exercise

- What is the output of the following expressions? Give reasons.

1. >> 8 + 3*5
2. >> (8 + 3)*5
3. >> 4^2 - 12 - 8/4*2
4. >> 4^2 - 12 - 8/(4*2)
5. >> 3*4^2 + 5
6. >> (3*4)^2 + 5
7. >> 27^(1/3) + 32^(0.2)
8. >> 27^(1/3) + 32^0.2
9. >> 27^1/3 + 32^0.2

ANSWERs

- Output of the following expressions.

1. >> 8 + **3*5** = 8+**15** = **23**
2. >> **(8 + 3)***5 = **11** * 5 = 55
3. >> **4^2** - 12 - **8/4***2 = 16 – 12 – 2*2 = 0
4. >> **4^2** - 12 - 8/**(4*2)** = 16 – 12 – 8/8 = 3
5. >> 3***4^2** + 5 = 3*16+5 = 53
6. >> **(3*4)**^2 + 5 = 149
7. >> 27^**(1/3)** + 32^**(0.2)** = 27^**0.333** + 2 = 5
8. >> 27^(1/3) + 32^0.2 = 5
9. >> **27^1**/3 + 32^0.2 = 9+2 = 11

Elementary Mathematical functions

MATLAB has several built-in functions, such as square root, sine, cosine, exponential, etc., functions.

Elementary math functions.

Type “help elfun” (elementary math functions) in the Command window

round(x)	Rounds x to the nearest integer
floor(x)	Rounds x down to nearest integer
ceil(x)	Rounds x up to nearest integer
rem(y,x)	Remainder after dividing y by x (eg: remainder of 17/3 is 2)
sign(x)	Returns -1 if x; returns 0 if x=0; returns 1 if x>0.
rand or **rand**(1)	Generates a random number between 0 and 1
exp(x)	Exponential function e^x
log(x)	Natural logarithm function, y=lnx (where $e^y=x$)
sqrt(x)	Square root function,
abs(x)	Absolute value function,
sin(x)	sine function sinx
cos(x)	cos function cosx
tan(x)	tan function tanx

Data analysis and statistics functions

- [val idx]=**max**(X) - returns **maximum** value and its **index** from a vector.
- If X is a matrix, a row vector containing the maximum element from each column is returned.

- >> x = [1, 5, 3; 2, 4, 6]; y = [10,2,4; 1, 8, 7];
- >> **max**(x,y) - Each element in the resulting matrix contains the maximum value from the corresponding positions in x and y.

- Similarly for **min()** function.

Data analysis and statistics functions

- **mean**(x)- Computes the mean value (or average value) of a vector x
- **median**(x) -Finds the median of the elements of a vector x
- **mode**(x)- Finds the value that occurs most often in an array
- **std**(x) Computes the standard deviation of the values in a vector x
- **var**(x) Calculates the variance of the data in x

Sorting functions

- **sort**(x) -Sorts the elements of a vector x into ascending order
- **sort**(x,**'descend'**)- Sorts the elements in descending order (column wise for matrix)
- **sortrows**(x) -Sorts the rows in a matrix in ascending order based on the values in the first column, and keeps each row intact
- **sortrows**(x,n) - Sorts the rows in a matrix on the basis of the values in column n

Convert Complex Number from Rectangular Form to Polar (Trigonometric) Form

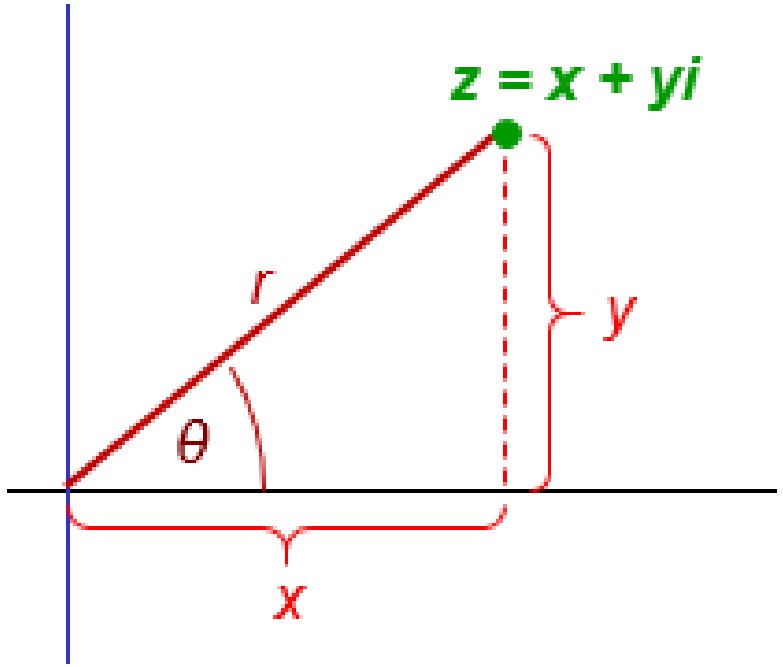
$z = x + yi$ (rectangular form)

$$r = |z| = \sqrt{x^2 + y^2}$$

$$x = r\cos\theta$$

$$y = r\sin\theta$$

$z = r(\cos\theta + i\sin\theta)$ (polar form)



Complex numbers

Function	Description	Example
i,j	Imaginary unit. As the basic imaginary unit SQRT(-1), i and j are used to enter complex numbers. For example, the expressions 3+2i, 3+2*i, 3+2j, 3+2*j and 3+2*sqrt(-1) all have the same value.	>>z=2+4i >>z=2+4j
abs	abs(x) is the absolute value of the elements of x. When x is complex, abs(x) is the complex modulus (magnitude) of the elements of X.	>>z=2+4i >>abs(z)
angle	Phase angle. angle(z) returns the phase angles, in radians	>>z=2+4i >>angle(z)
imag	Complex imaginary part. imag(z) is the imaginary part of z.	>>z=2+4i >>b=imag(z)
real	Complex real part. real(z) is the real part of z.	>>z=2+4i >>a=real(z)
conj	Complex conjugate. conj(x) is the complex conjugate of x.	>>z=2+4i >>z_con=conj(z)
complex	Construct complex result from real and imaginary parts. c = complex(a,b) returns the complex result A + Bi	>>a=2; >>b=3; >>z=complex(a,b)

Discrete mathematics

- **factor**(x) - Finds the prime factors of x.
- **gcd**(x,y) Finds the greatest common denominator of x and y.
- **lcm**(x,y) Finds the least common multiple of x and y.
- **rats**(x) Represents x as a fraction (1.5 as ½)

Discrete mathematics

- **factorial**(x) Finds the value of x factorial (x!).
-
- **nchoosek**(n,k) Finds the number of possible combinations of k items from a group of n items.
- **primes**(x) Finds all the prime numbers less than x.
- **isprime**(x) Checks to see if x is a prime number. If it is, the function returns 1; if not, it returns 0.

Data Import and Export

Data access from Text files, spreadsheets, images;

File I/O functions – read, write, delete

https://in.mathworks.com/help/matlab/import_export/supported-file-formats-for-import-and-export.html

Text files

- Read and write numeric and non-numeric data in delimited and formatted text files like **.csv** and **.txt** files.

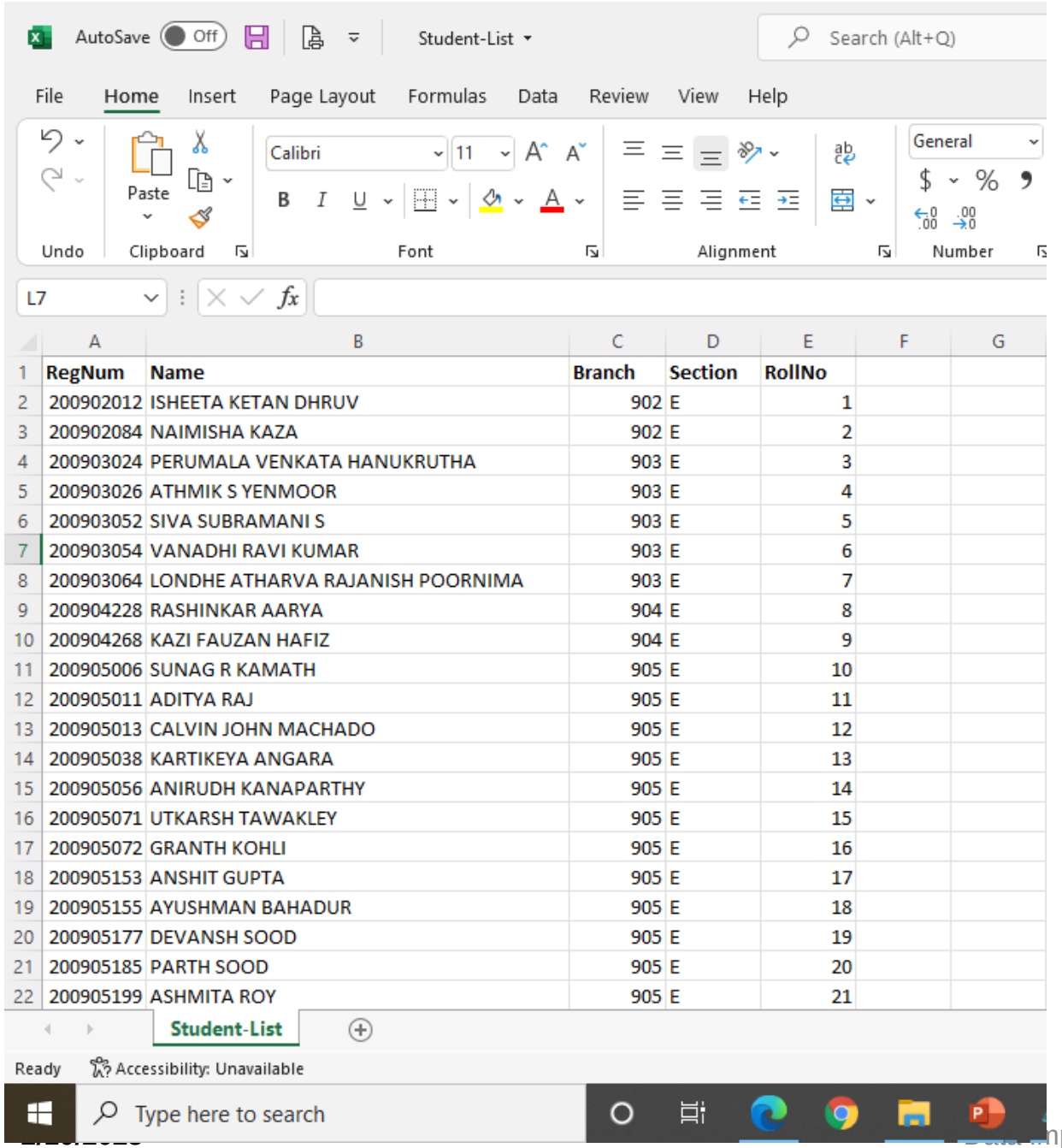
- **Text** files often contain a **mix** of **numeric** and **text** data as well as **variable** and **row** names.

- You can represent this data in MATLAB as **tables**, **timetables**, **matrices**, **cell arrays**, or **string arrays**.

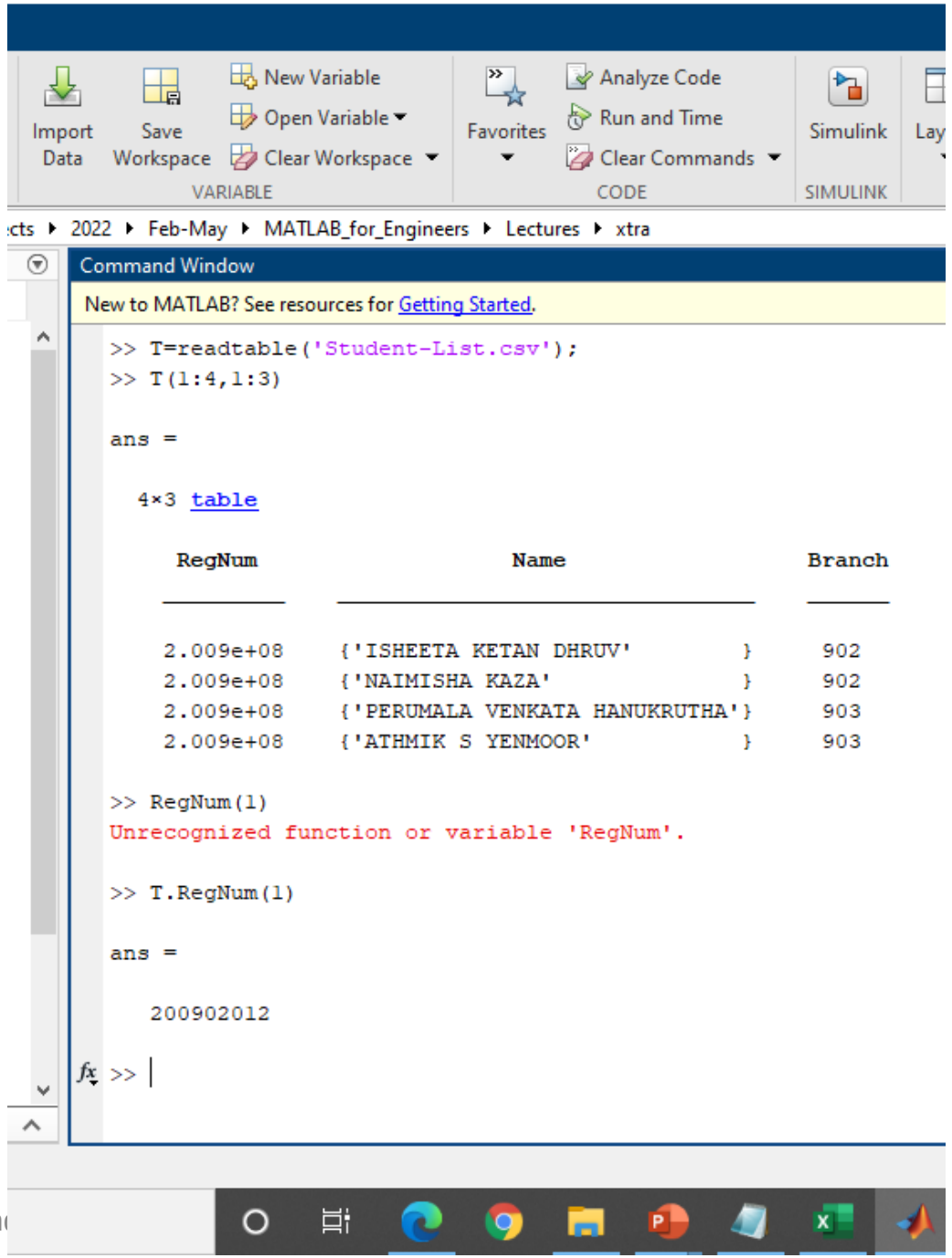
Import Data as Tables

- If text file has tabular data, it can be imported as a table.
- A table consists of column-oriented variables containing rows of data of the same type.
- Each variable in a table can hold a different data type and size, however, each variable must have the same number of rows.

- >> T = **readtable**(‘**Student-List**.**csv**’);
- >> T(1:5,1:5) **% Display the first five rows and columns of the table**.



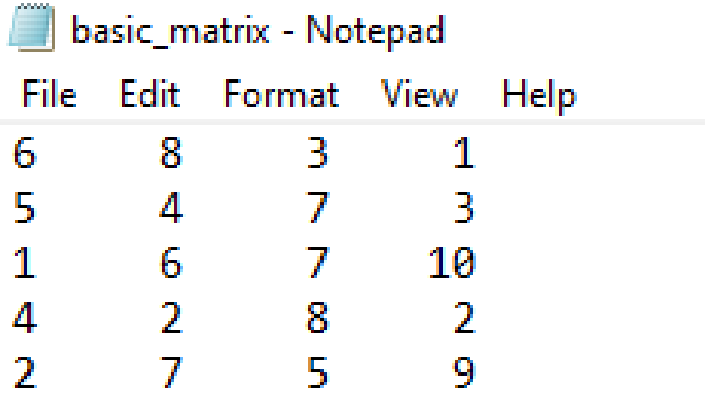
	A	B	C	D	E
1	RegNum	Name	Branch	Section	RollNo
2	200902012	ISHEETA KETAN DHRUV	902	E	1
3	200902084	NAIMISHA KAZA	902	E	2
4	200903024	PERUMALA VENKATA HANUKRUTHA	903	E	3
5	200903026	ATHMIK S YENMOOR	903	E	4
6	200903052	SIVA SUBRAMANI S	903	E	5
7	200903054	VANADHI RAVI KUMAR	903	E	6
8	200903064	LONDHE ATHARVA RAJANISH POORNIMA	903	E	7
9	200904228	RASHINKAR AARYA	904	E	8
10	200904268	KAZI FAUZAN HAFIZ	904	E	9
11	200905006	SUNAG R KAMATH	905	E	10
12	200905011	ADITYA RAJ	905	E	11
13	200905013	CALVIN JOHN MACHADO	905	E	12
14	200905038	KARTIKEYA ANGARA	905	E	13
15	200905056	ANIRUDH KANAPARTHY	905	E	14
16	200905071	UTKARSH TAWAKLEY	905	E	15
17	200905072	GRANTH KOHLI	905	E	16
18	200905153	ANSHIT GUPTA	905	E	17
19	200905155	AYUSHMAN BAHADUR	905	E	18
20	200905177	DEVANSH SOOD	905	E	19
21	200905185	PARTH SOOD	905	E	20
22	200905199	ASHMITA ROY	905	E	21

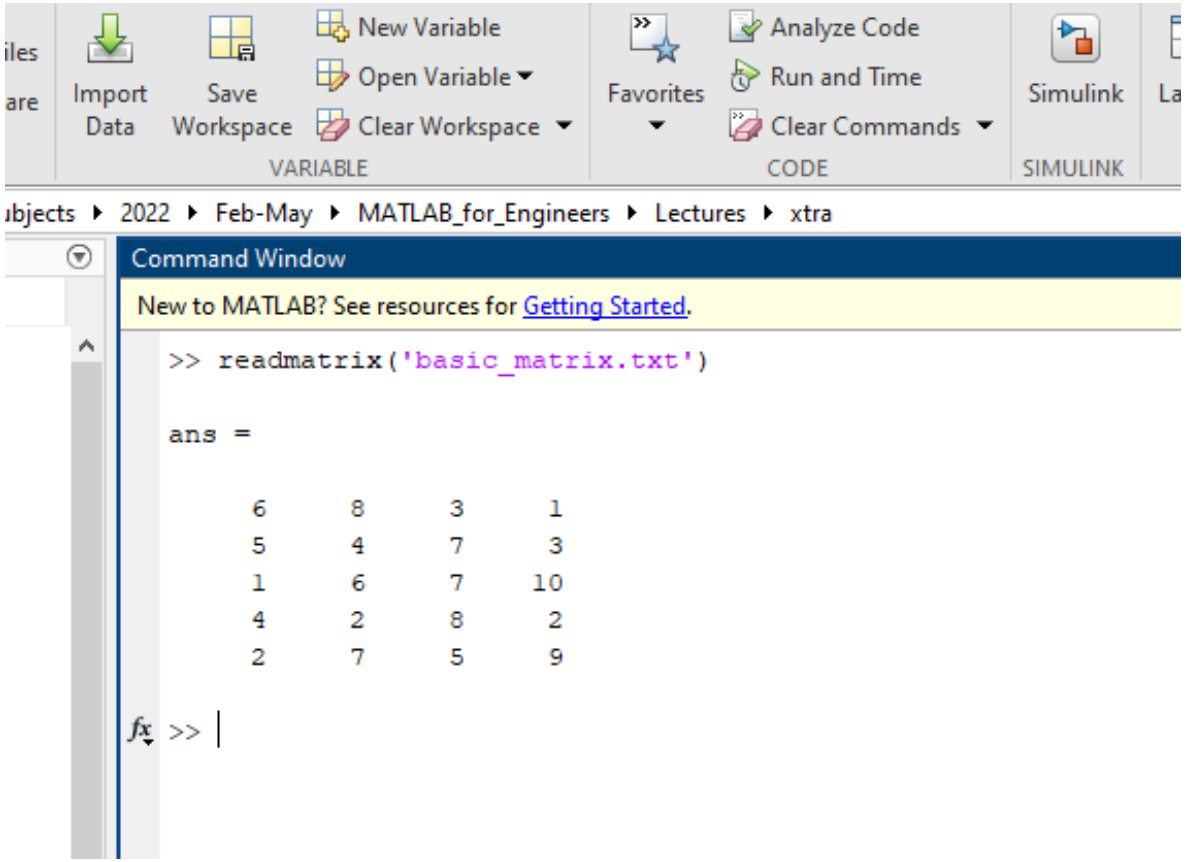


Import Data as Matrices

- If text file contains uniform data (all of the same type), it can be imported as a matrix. Importing data into a matrix allows to work with a minimally formatted array.

- >> M = readmatrix('basic_matrix.txt')





Import Data as Cell Arrays

- A cell array is a data type with **indexed data containers** called **cells**, where **each cell** can **contain any type of data**.

student-list - Notepad

File Edit Format View Help

```
200902012       ISHEETA KETAN DHRUV     902     E       1
200902084       NAIMISHA KAZA   902     E       2
200903024       PERUMALA VENKATA HANUKRUTHA     903     E       3
200903026       ATHMIK S YENMOOR        903     E       4
200903052       SIVA SUBRAMANI S        903     E       5
200903054       VANADHI RAVI KUMAR      903     E       6
200903064       LONDHE ATHARVA RAJANISH POORNIMA        903     E       7
200904228       RASHINKAR AARYA 904     E       8
200904268       KAZI FAUZAN HAFIZ       904     E       9
200905006       SUNAG R KAMATH  905     E       10
200905011       ADITYA RAJ      905     E       11
200905013       CALVIN JOHN MACHADO     905     E       12
200905038       KARTIKEYA ANGARA        905     E       13
200905056       ANIRUDH KANAPARTHY      905     E       14
200905071       UTKARSH TAWAKLEY        905     E       15
200905072       GRANTH KOHLI    905     E       16
200905153       ANSHIT GUPTA    905     E       17
200905155       AYUSHMAN BAHADUR        905     E       18
200905177       DEVANSH SOOD    905     E       19
200905185       PARTH SOOD      905     E       20
200905199       ASHMITA ROY     905     E       21
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> C = readcell('student-list.txt')

C =

  21×5 cell array

    {[200902012]}    {'ISHEETA KETAN D…'}    {[902]}    {'E'}    {[ 1]}
    {[200902084]}    {'NAIMISHA KAZA'    }    {[902]}    {'E'}    {[ 2]}
    {[200903024]}    {'PERUMALA VENKAT…'}    {[903]}    {'E'}    {[ 3]}
    {[200903026]}    {'ATHMIK S YENMOOR' }    {[903]}    {'E'}    {[ 4]}
    {[200903052]}    {'SIVA SUBRAMANI S' }    {[903]}    {'E'}    {[ 5]}
    {[200903054]}    {'VANADHI RAVI KU…'}    {[903]}    {'E'}    {[ 6]}
    {[200903064]}    {'LONDHE ATHARVA …'}    {[903]}    {'E'}    {[ 7]}
    {[200904228]}    {'RASHINKAR AARYA'  }    {[904]}    {'E'}    {[ 8]}
    {[200904268]}    {'KAZI FAUZAN HAFIZ'}    {[904]}    {'E'}    {[ 9]}
    {[200905006]}    {'SUNAG R KAMATH'   }    {[905]}    {'E'}    {[10]}
    {[200905011]}    {'ADITYA RAJ'       }    {[905]}    {'E'}    {[11]}
    {[200905013]}    {'CALVIN JOHN MAC…'}    {[905]}    {'E'}    {[12]}
    {[200905038]}    {'KARTIKEYA ANGARA' }    {[905]}    {'E'}    {[13]}
    {[200905056]}    {'ANIRUDH KANAPAR…'}    {[905]}    {'E'}    {[14]}
    {[200905071]}    {'UTKARSH TAWAKLEY' }    {[905]}    {'E'}    {[15]}
    {[200905072]}    {'GRANTH KOHLI'     }    {[905]}    {'E'}    {[16]}
    {[200905153]}    {'ANSHIT GUPTA'     }    {[905]}    {'E'}    {[17]}
    {[200905155]}    {'AYUSHMAN BAHADUR' }    {[905]}    {'E'}    {[18]}
    {[200905177]}    {'DEVANSH SOOD'     }    {[905]}    {'E'}    {[19]}
    {[200905185]}    {'PARTH SOOD'       }    {[905]}    {'E'}    {[20]}
    {[200905199]}    {'ASHMITA ROY'      }    {[905]}    {'E'}    {[21]}
```

textscan

- Read formatted data from text file or string

- **fileID** = **fopen**(‘student-list.txt’,'r');
- C = textscan(**fileID**,’%s %s %s %s %s','Delimiter',’ ');
- fclose(fileID);

- Format Specifiers :
- https://in.mathworks.com/help/matlab/ref/textscan.html

MATLAB R2021a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation | Shailesh

New Open Save Find Files Compare Print — FILE
Go To Find — NAVIGATE
Insert Comment Indent — EDIT
Breakpoints — BREAKPOINTS
Run Run and Advance Run Section Advance Run and Time — RUN

E: ▸ office ▸ subjects ▸ 2022 ▸ Feb-May ▸ MATLAB_for_Engineers ▸ Lectures ▸ xtra

Editor - E:\office\subjects\2022\Feb-May\MATLAB_for_Engineers\Lectures\xtra\read4mtxtfile.m

read4mtxtfile.m

```
1 -    clear all;clc;
2 -    fileID = fopen('student-list.txt','r');
3 -    C = textscan(fileID,'%s %s %s %s %s','Delimiter','');
4 -    fclose(fileID);
5
```

Current Folder

Name

- tutorial_01.mlx
- Tut_03_while.m
- Tut_03_binary_search.m
- Tut_03_arr_rev_itr.m
- Tut_03_03.m
- Tut_03_02.m
- Tut_03_01.m
- Tut_02_06.m
- Tut_02_03.mlx
- tut_02_03.m
- tut_02_02.m
- tut_02_01.m
- test_live.mlx
- test.m
- T02-Live_script.mlx
- Student-List.csv
- student-list.txt
- recArrayRev.m

Details

Command Window

New to MATLAB? See resources for Getting Started.

```
>> celldisp(C)

C{1}{1} =

200902012   ISHEETA KETAN DHRUV 902 E   1


C{1}{2} =

200902084   NAIMISHA KAZA   902 E   2


C{1}{3} =

200903024   PERUMALA VENKATA HANUKRUTHA 903 E   3


C{1}{4} =
```

Workspace

Name	Value
ans	0
C	1x5 cell
fileID	3

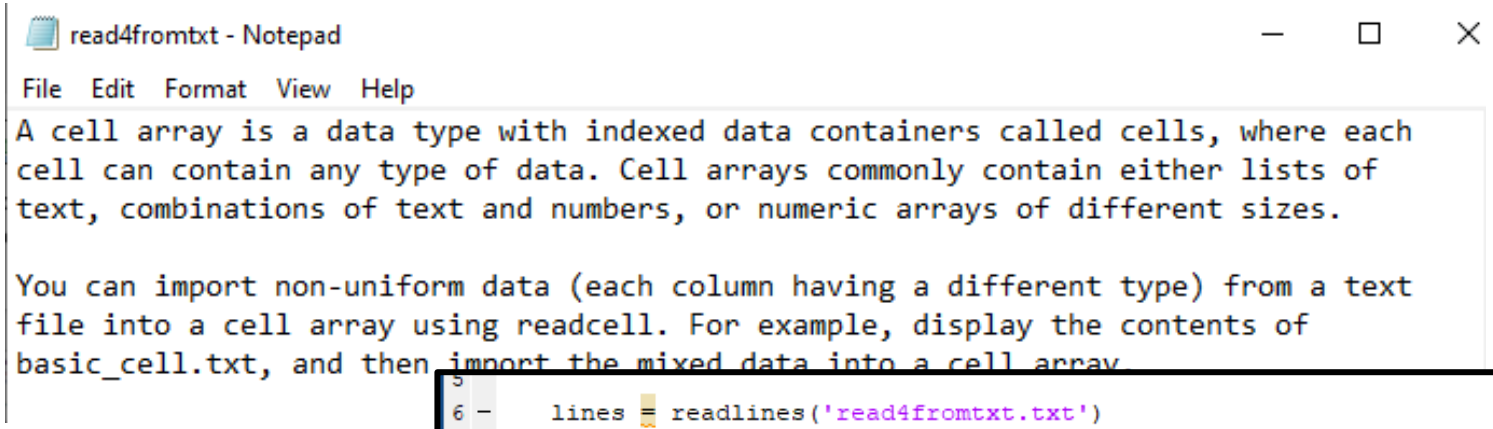
Type here to search

28°C Mostly cloudy

09:11 AM
26-Mar-22

Import Data as String Arrays

- If text file contains lines of plain text, it can be represented as plain text in MATLAB as a string array.




```
6 -    lines = readlines('read4fromtxt.txt')
```

Current Folder

Name

- tutorial_01.mlx
- Tut_03_while.m
- Tut_03_binary_search.m
- Tut_03_arr_rev_itr.m
- Tut_03_03.m
- Tut_03_02.m
- Tut_03_01.m
- Tut_02_06.m
- Tut_02_03.mlx
- tut_02_03.m
- tut_02_02.m
- tut_02_01.m

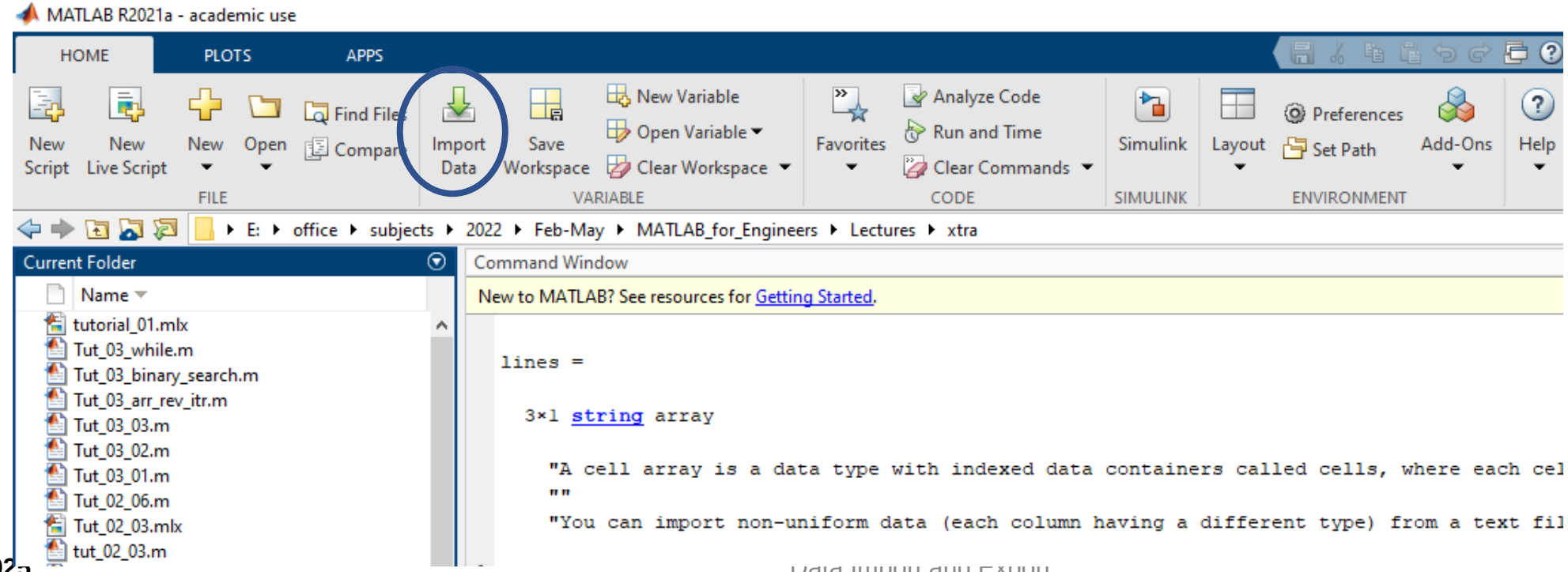
Command Window

New to MATLAB? See resources for Getting Started.

```
lines =

  3×1 string array

    "A cell array is a data type with indexed data containers called cells, where each cell ca
    ""
    "You can import non-uniform data (each column having a different type) from a text file ir

fx >>
```

Import Tool (Homework / Self study)

- The Import Tool lets you preview and import data from spreadsheet files, delimited text files, and fixed-width text files.
- You can interactively select the data to import and reuse the script or function that the tool generates to import other similar files.



Importing Spreadsheets (Excel files)

- Import Spreadsheet Data Using **readtable**
  - T = **readtable**('patients.xls’);
  - T = **readtable**('patients.xls','Range','A1:E5’)
  - Specify the range in Excel notation as 'A1:E5' to read the first five rows and columns of the spreadsheet.

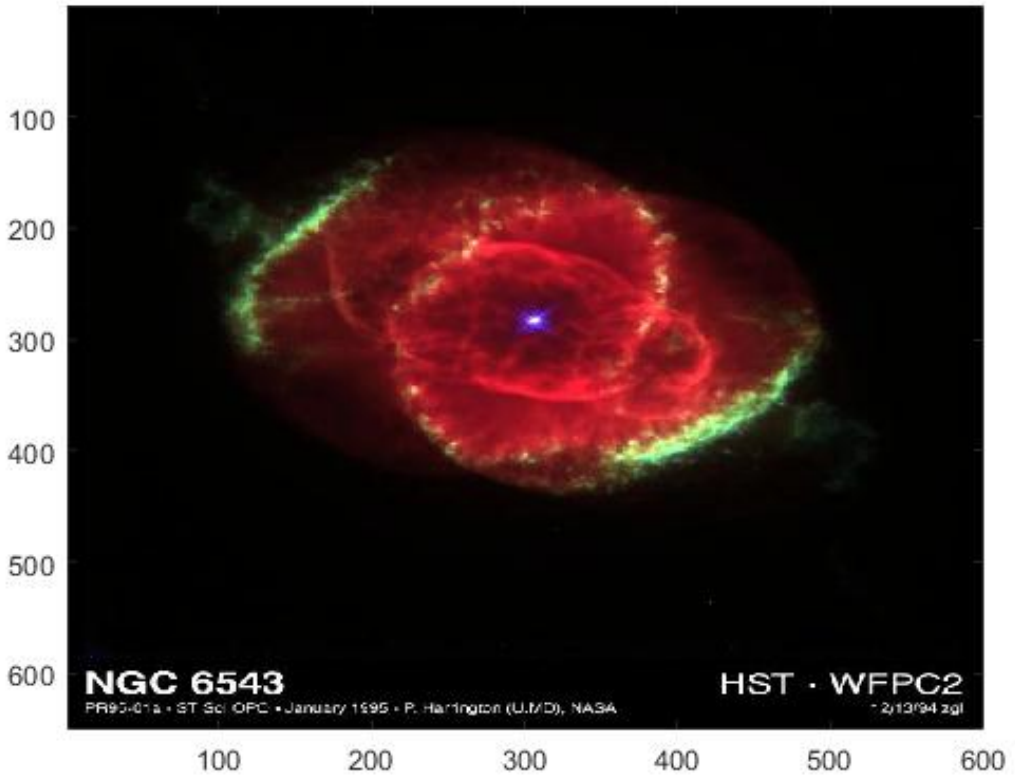
- Read Spreadsheet Data into Matrix
  - M = **readmatrix**('basic_matrix.xls')
  - M = **readmatrix**('basic_matrix.xls','Sheet','Sheet1','Range','B1:D3')

Importing Images

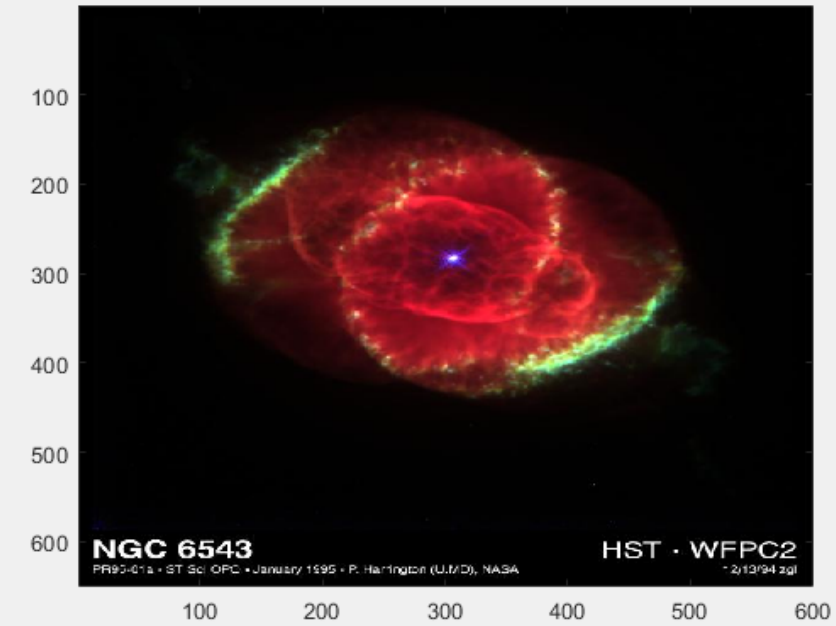
- To import data into the MATLAB workspace from a graphics file, use the **imread** function.

- >> A = **imread**('ngc6543a.jpg');
- >> image(A); % Display image file

- >> info = **imfinfo**(filename) –
- Information about graphics file



```
>> A = imread('ngc6543a.jpg');
>> image(A)
>> imfinfo('ngc6543a.jpg')

ans =

  struct with fields:

          Filename: 'E:\MATLAB\toolbox\matlab\demos
      FileModDate: '02-Oct-1996 01:49:44'
         FileSize: 27387
           Format: 'jpg'
    FormatVersion: ''
            Width: 600
           Height: 650
         BitDepth: 24
        ColorType: 'truecolor'
  FormatSignature: ''
  NumberOfSamples: 3
     CodingMethod: 'Huffman'
    CodingProcess: 'Sequential'
          Comment: {'CREATOR: XV Version 3.00b  Rev: 6/15/94  Quality = 75, Smoothing = 0

>> image(A)
>>
```



Low-Level File I/O

- **fscanf** - Read data from text file

Create a sample text file that contains floating-point numbers.

```
x = 100*rand(8,1);
fileID = fopen('nums1.txt','w');
fprintf(fileID,'%4.4f\n',x);
fclose(fileID);
```

```
>> writeandread
>> type 'nums1.txt'

81.4724
90.5792
12.6987
91.3376
63.2359
9.7540
27.8498
54.6882
>>
```

Reading from the txt file

Define the format of the data to read.
Use '%f' to specify floating-point numbers.

```
fileID = fopen('nums1.txt','r');
formatSpec = '%f';
A = fscanf(fileID,formatSpec);
fclose(fileID);
```

Export functions

Writing as TXT, XLS and IMAGEs

writematrix

- **writematrix**(A,**filename**) - writes to a file with the name and extension specified by **filename**.

- writematrix determines the file format based on the specified extension.
- The extension must be one of the following:
- **.txt, .dat, or .csv** for delimited text files
- **.xls, .xlsm, or .xlsx** for Excel spreadsheet files
- **.xlsb** for Excel spreadsheet files supported on systems with Excel for Windows

writematrix

- M = magic(5);

```
M = 5×5

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

Write the matrix to a comma delimited text file and display the file contents. The **writematrix** function outputs a text file named **M.txt**.

```
>>writematrix(M)
>>type 'M.txt'
```

To write the same matrix to a text file with a different delimiter character, use the 'Delimiter' name-value pair.

```
>>writematrix(M, 'M_tab.txt', 'Delimiter','tab')
>>type 'M_tab.txt'
```

Write the matrix to a spreadsheet file.

```
>>writematrix(M,'M.xls')
```

Read and display the matrix from M.xls.

```
>>readmatrix('M.xls')
```

Write the matrix to M.xls, to the second worksheet in the file, starting at the third row.

```
>>writematrix(M,'M.xls','Sheet',2,'Range','A3:E8')
```

Read and display the matrix.

```
>>readmatrix('M.xls','Sheet',2,'Range','A3:E8')
```

Append Data to Spreadsheet

- Append an array of data below existing data in a spreadsheet.

```
>> M1 = magic(5)  % First Matrix
>> writematrix(M1,'M.xls');
>> M2 = [5 10 15 20 25; 30 35 40 45 50]  % second matrix
>> writematrix(M2,'M.xls','WriteMode','append');
>> readmatrix('M.xls')
```

Append Matrix Data to Text File

- Append an array of data below existing data in a text file.

```
>> writematrix(M1,’M.txt’)
>> writematrix(M2,’M.txt','WriteMode','append’)
>> readmatrix('fibonacci.txt')
```

writetable

- Write table to file
- >> T = table(['M';'F';'M'],[45 45;41 32;40 34],{'NY';'CA';'MA'},[true; false; false])

```
T=3×4 table
    Var1      Var2      Var3      Var4
    ____    ________    ______    _____

     M        45        45      {'NY'}    true
     F        41        32      {'CA'}    false
     M        40        34      {'MA'}    false
```

Write the table to a **comma delimited text** file and display the file contents.

```
>>writetable(T)

writetable outputs a text file named T.txt.
>>type 'T.txt'
```

Write the table to a **space-delimited text** file named myData.txt and display the file contents.

```
>>writetable(T,'myData.txt','Delimiter',' ')
>>type 'myData.txt'
```

writetable

- Write the table to a **comma-separated text file** named **myData.csv** and view the file contents. Use the **'QuoteStrings**' name-value pair argument to **ensure** that the **commas** in the **third column are not treated as delimiters**.

- >> **writetable**(T,'myData.csv','Delimiter',',','QuoteStrings',true)
- >> **type** 'myData.csv'

writetable

- Write Table to Specific Sheet and Range in Spreadsheet

- Write the table to a spreadsheet named myData.xls. Include the data on the first sheet in the 5-by-5 region with corners at B2 and F6. You can change the worksheet to write to by specifying the index corresponding to the worksheet.

- >> **writetable**(T,'myData.xls','Sheet',1,'Range','B2:F6')

writelines

- Write text to file

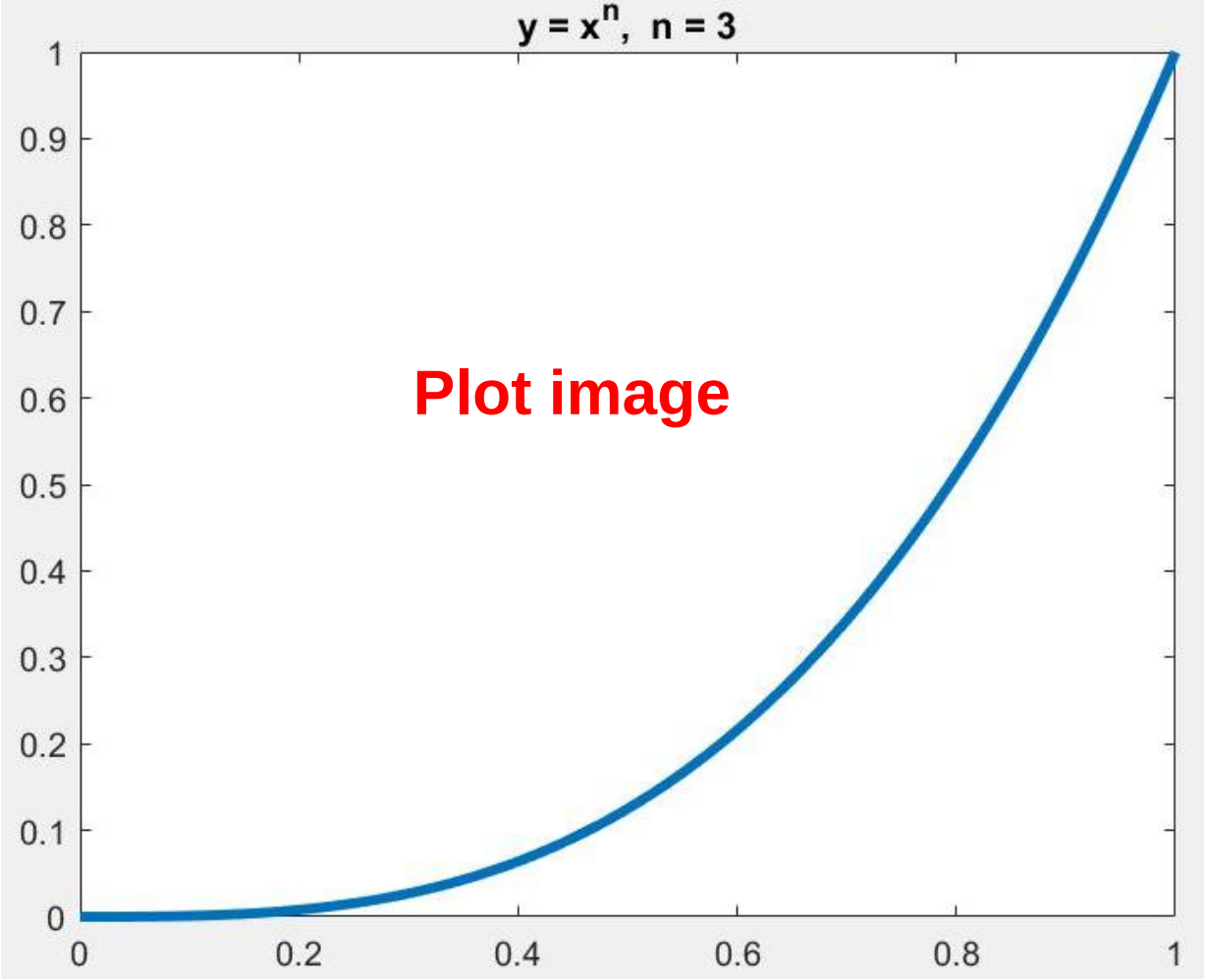
- Write the text "Example String" to a new file within the current directory.
- >>**writelines**("Example String","temp.txt")
- Display the contents of the new file.
- >>**type** temp.txt

- Append a string to an existing file.
- **lines** = "New Content 456";
- **filename** = "temp.txt";
- **writelines**(**lines**,filename,WriteMode="append")

imwrite

- Write image to graphics file

```
x = 0:0.01:1;
n = 3;
y = x.^n;
plot(x,y,'LineWidth',3);
title(['y = x^n,  n = ' num2str(n) ]);

H = getframe(gcf);

% save the image:
imwrite(H.cdata, 'testimage.jpg');
```



Delete file

- Delete files from CURRENT directory
- >>**delete** filename1 … Filenamen
- To delete all files in the current folder with a **.mat** extension.
- **>> delete *.mat**

Graphics – 2D and 3D plots

Plot continuous, discrete, surface, and volume data

Create a 2-D Line Plot

- Create a two-dimensional line plot using the plot function.
- Eg: plot the value of the sine function from 0 to 2π.

Plot

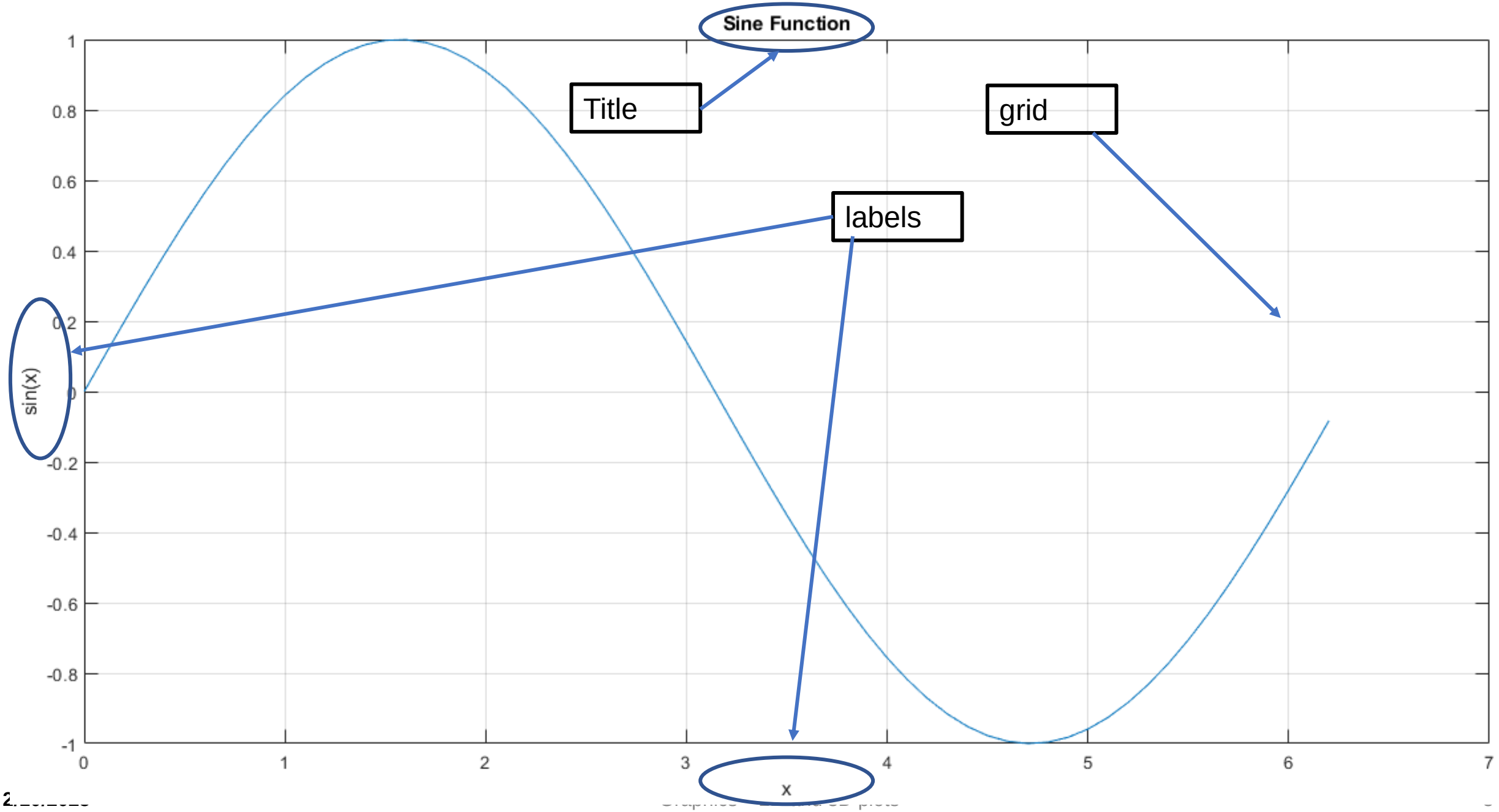
```
>> x = 0:0.1:2*pi;
 >> y = sin(x);
>> plot(x,y)
```

Label the axes and add a title

```
>>xlabel('x')                    %add label
>>ylabel('sin(x)')
>>title('Sine Function')         %add title
```

Display the grid lines for a sine plot

```
>>grid on
```


Sine Function
Title
grid
labels
sin(x)
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Multiple plots - using hold command

- By default, MATLAB **clears** the **figure before** each **plotting command**.
- Use the **figure** command to open a new figure window.
- Plot **multiple lines** using the **hold** on command.
- Until you use hold off or close the window, all plots appear in the current figure window.

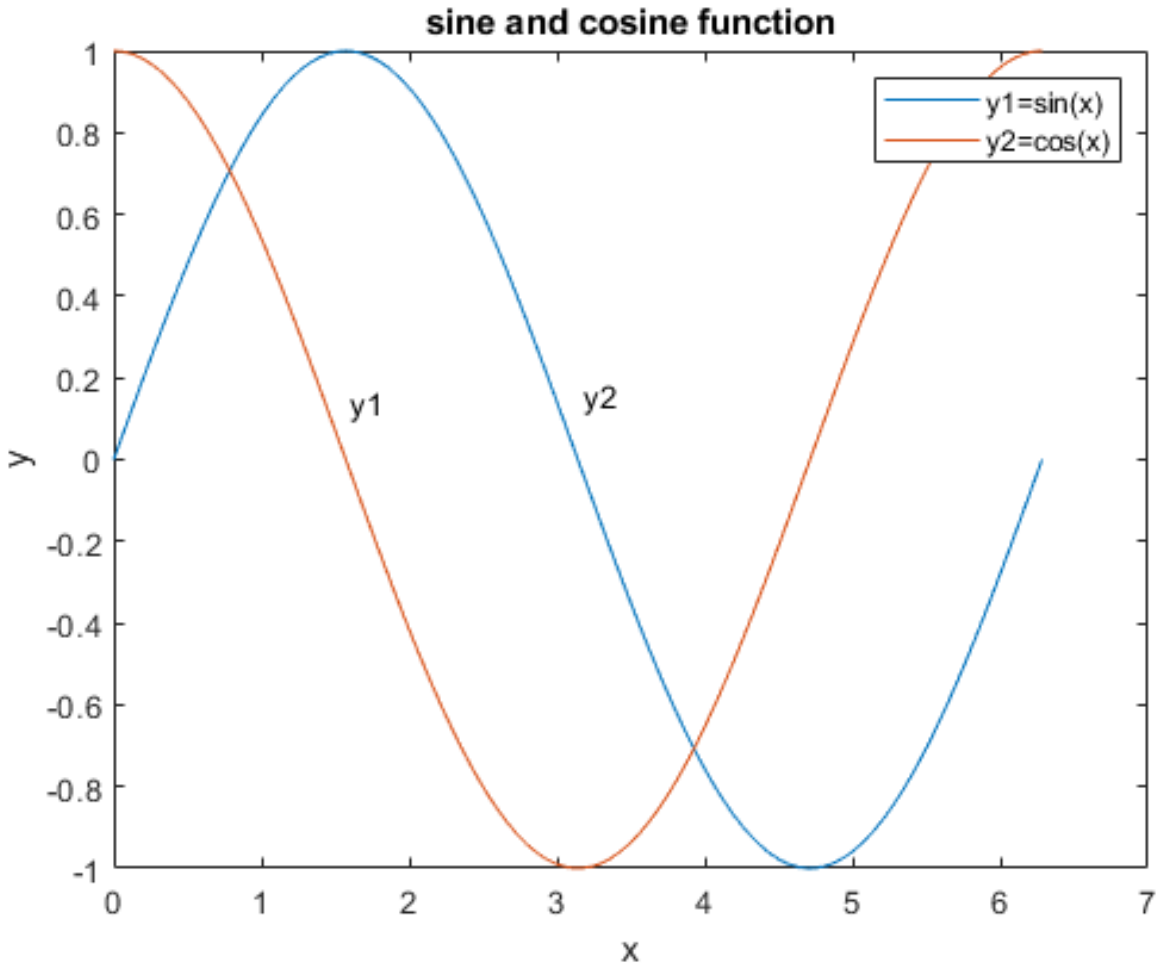
```
figure
x = linspace(0,2*pi,100);
y1 = sin(x);
plot(x,y)
hold on

y2 = cos(x);
plot(x,y2)
hold off

legend('y1= sin(x)','y2= cos(x)');
```

Add Legend

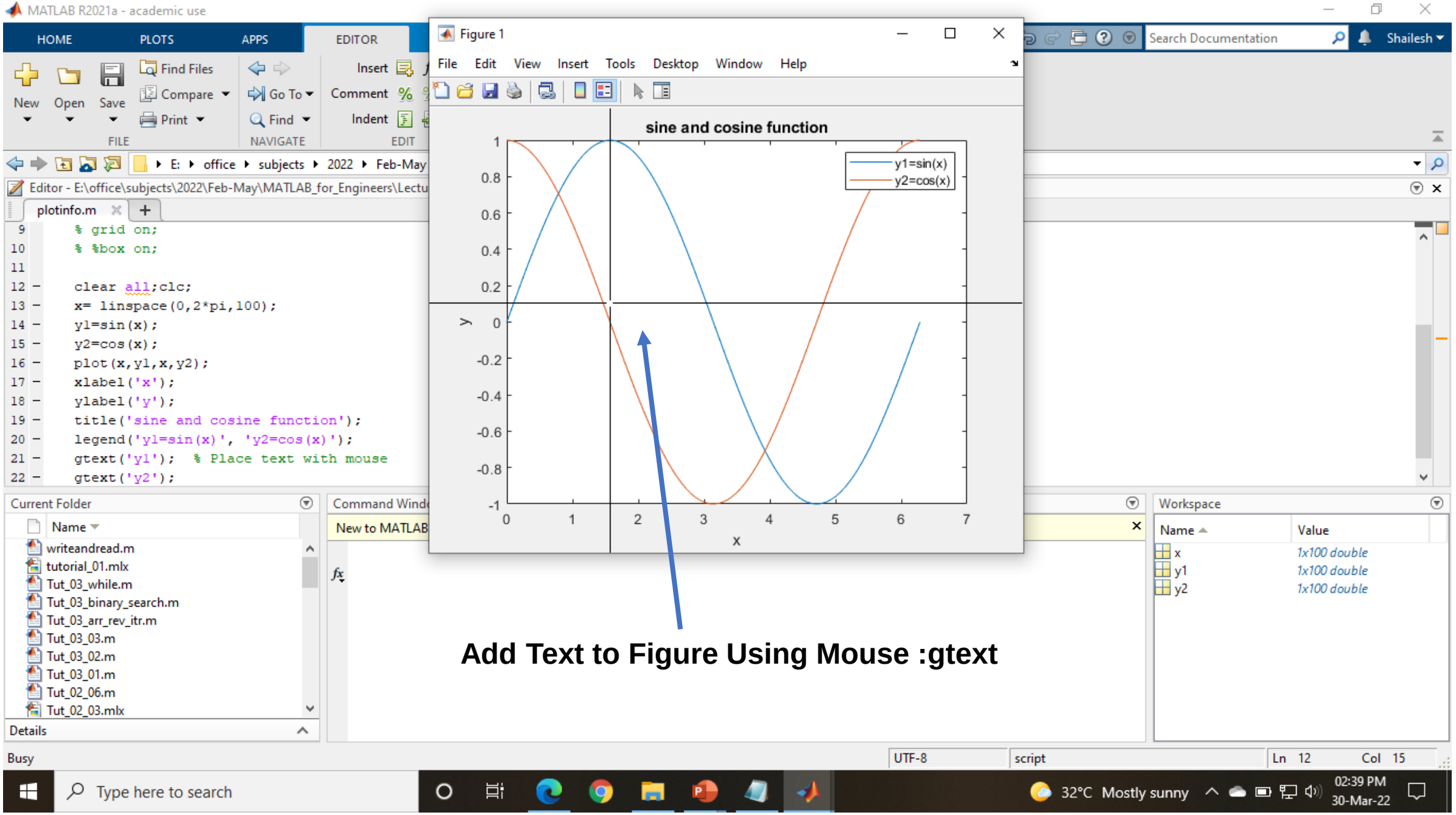
Add a legend to the graph that identifies each data set using the legend function

Use of plot command for Plotting Multiple Lines

```
>> x= linspace(0,2*pi,100)
>> y1=sin(x)
>> y2=cos(x)
>> plot(x,y1,x,y2)
>> xlabel('x')
>> ylabel('y')
>> title('sine and cosine function')
>> legend('y1=sin(x)', 'y2=cos(x)')
>> gtext('y1')          % Place text
with mouse
>> gtext('y2')
```



Add Text to Figure Using Mouse :gtext

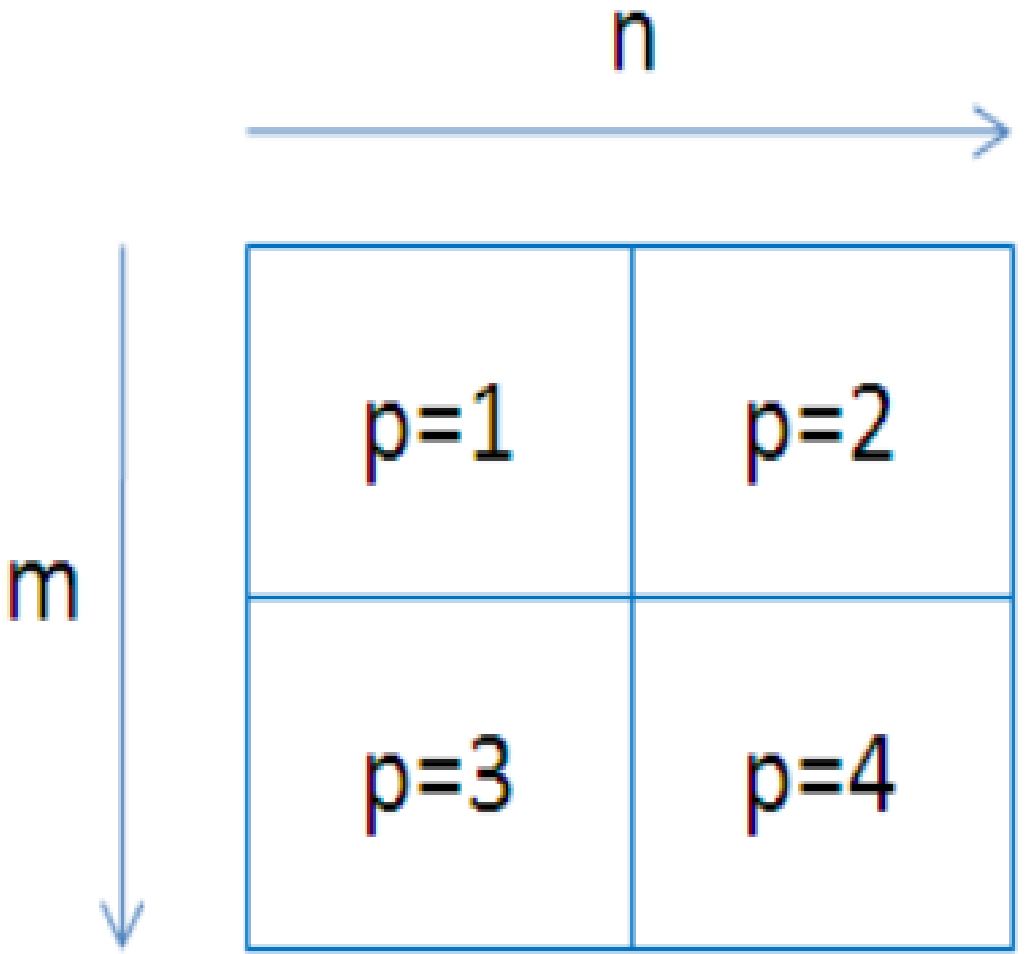


Add Text to Figure Using Mouse :gtext

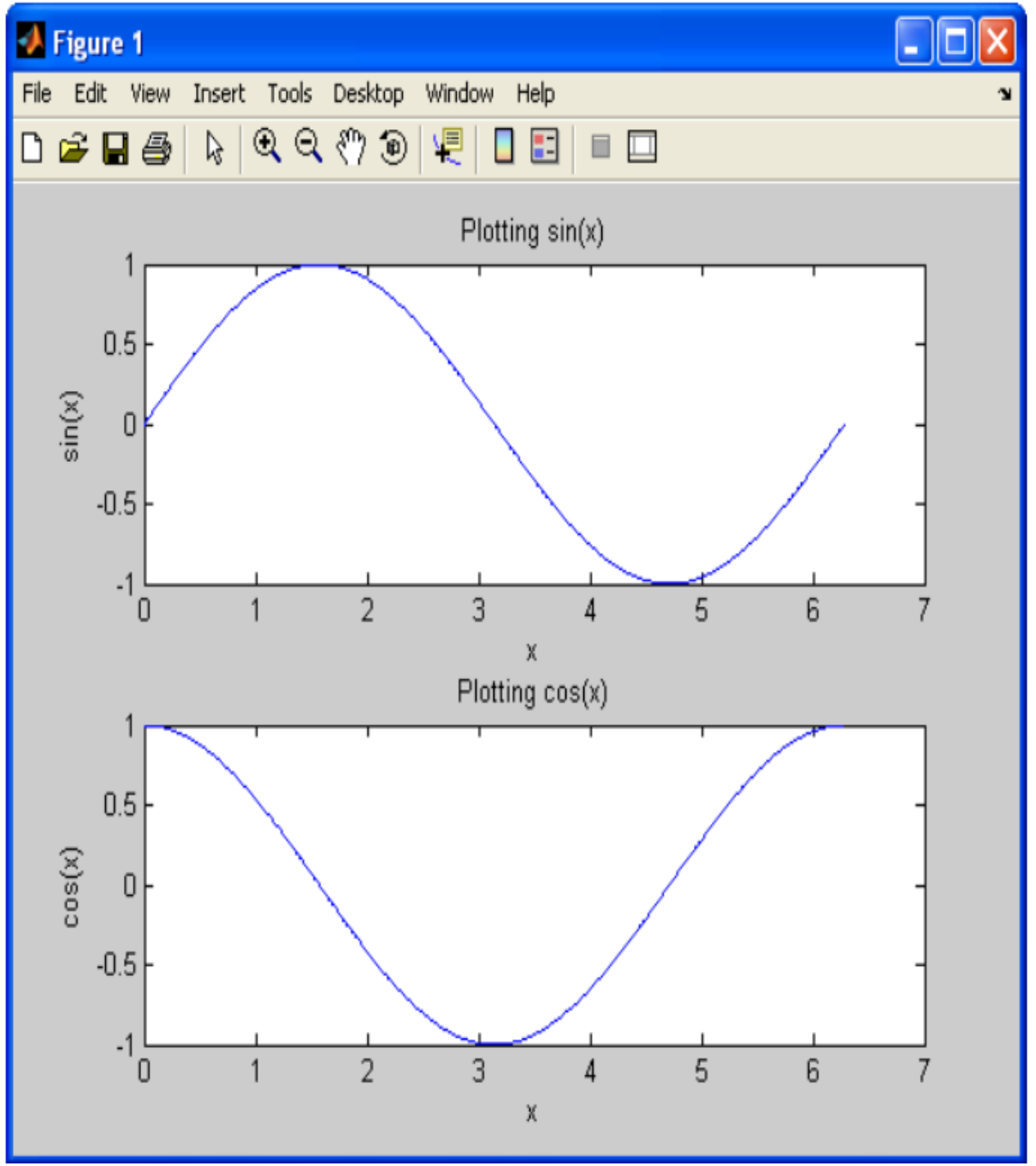
Sub-plots

- Displaying **Multiple** Plots in **one** Figure – **Sub-Plots**

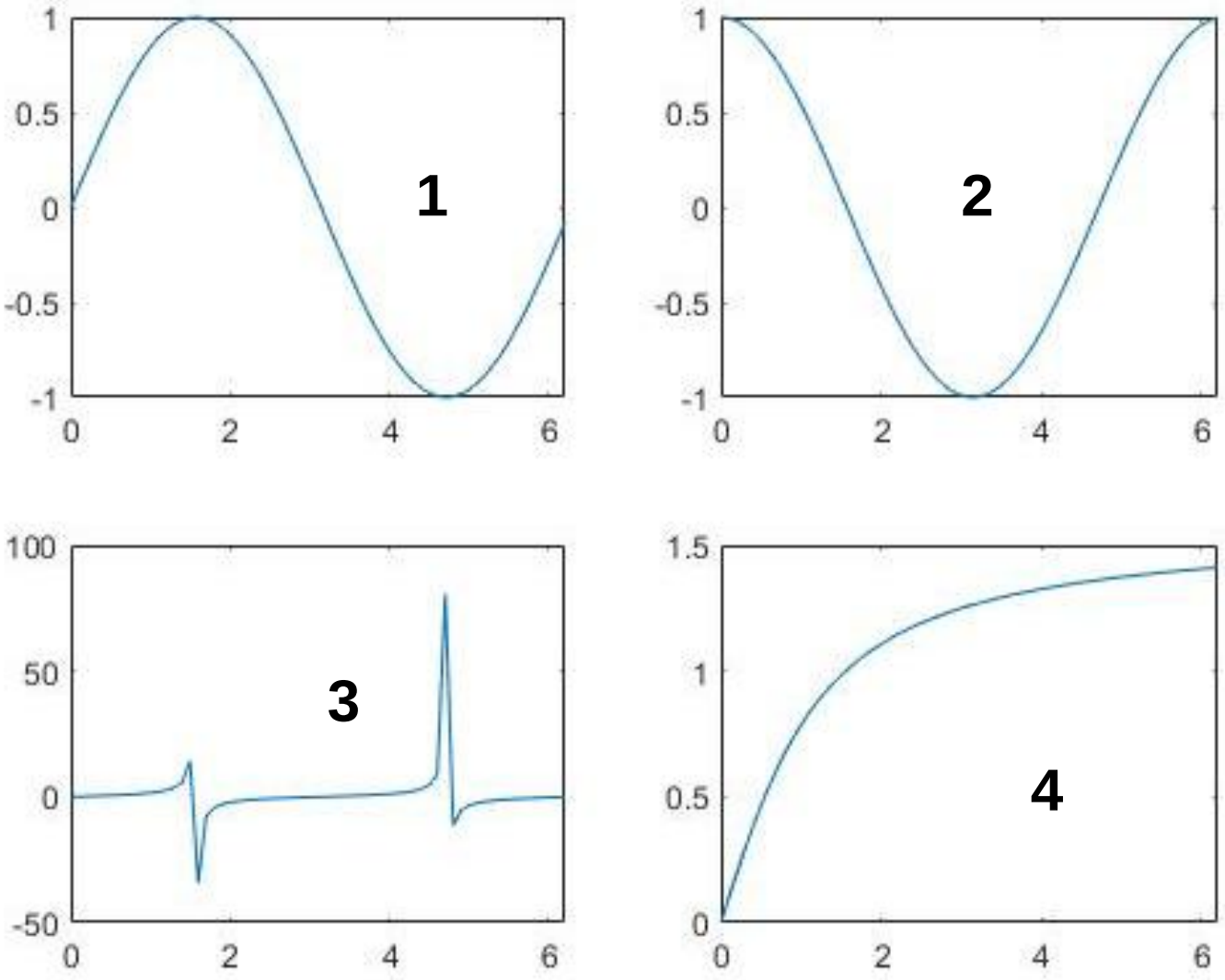
- **subplot**(m,n,p)



```
% Define x-values
x=0:0.01:2*pi;

% subplot 1
subplot(2,1,1) ;
plot(x, sin(x)) ;
title('Plotting sin(x)') ;
xlabel('x') ;
ylabel('sin(x)') ;

% Subplot 2
subplot(2,1,2);
plot(x, cos(x));
title('Plotting cos(x)') ;
xlabel('x');
ylabel('cos(x)') ;
```



```
x=0:0.1:2*pi;

y=sin(x);        y2=cos(x);
y3=tan(x);     y4=atan(x);

subplot(2,2,1) ; plot(x,y);
subplot(2,2,2); plot(x,y2);
subplot(2,2,3); plot(x,y3);
subplot(2,2,4); plot(x,y4);
```



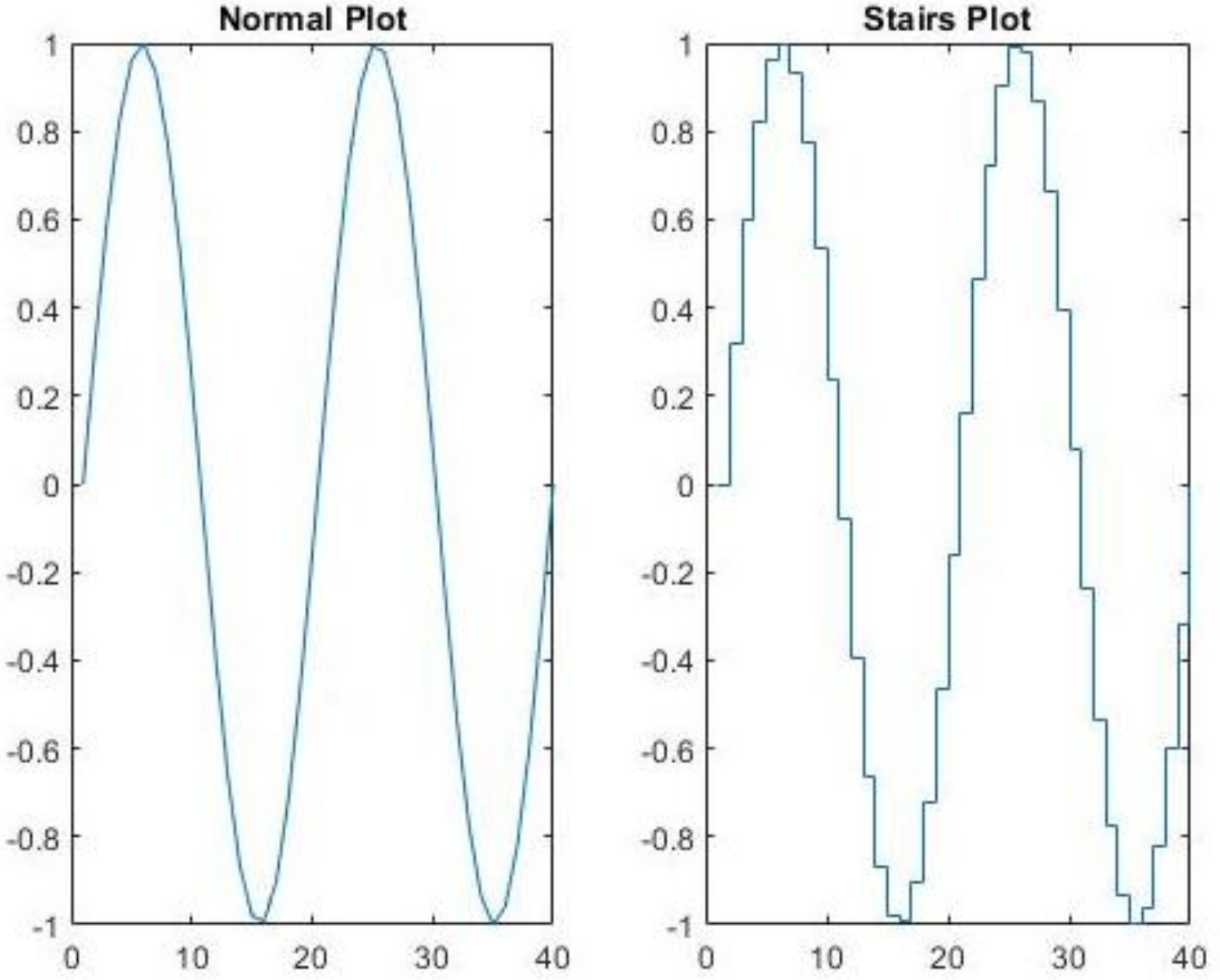
Special 2-D plots

plot	2-D line plot
plot3	3-D point or line plot
stairs	Stairstep graph
errorbar	Line plot with error bars
area	Filled area 2-D plot
stackedplot	Stacked plot of several variables with common x-axis

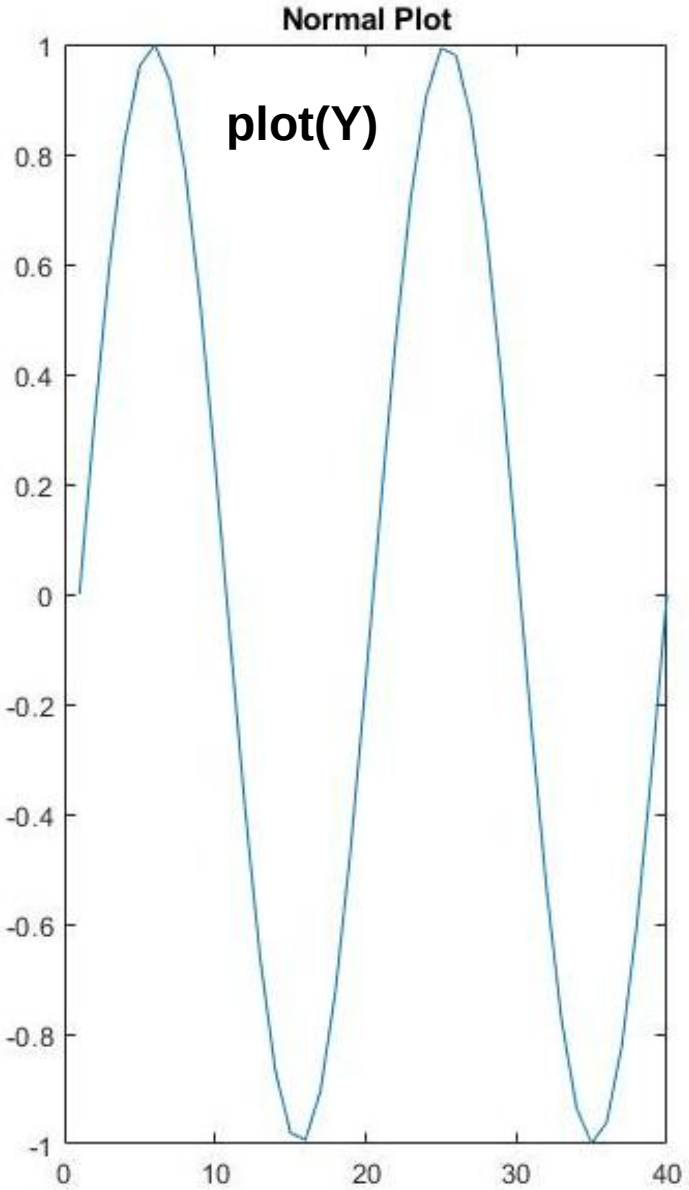
stairs

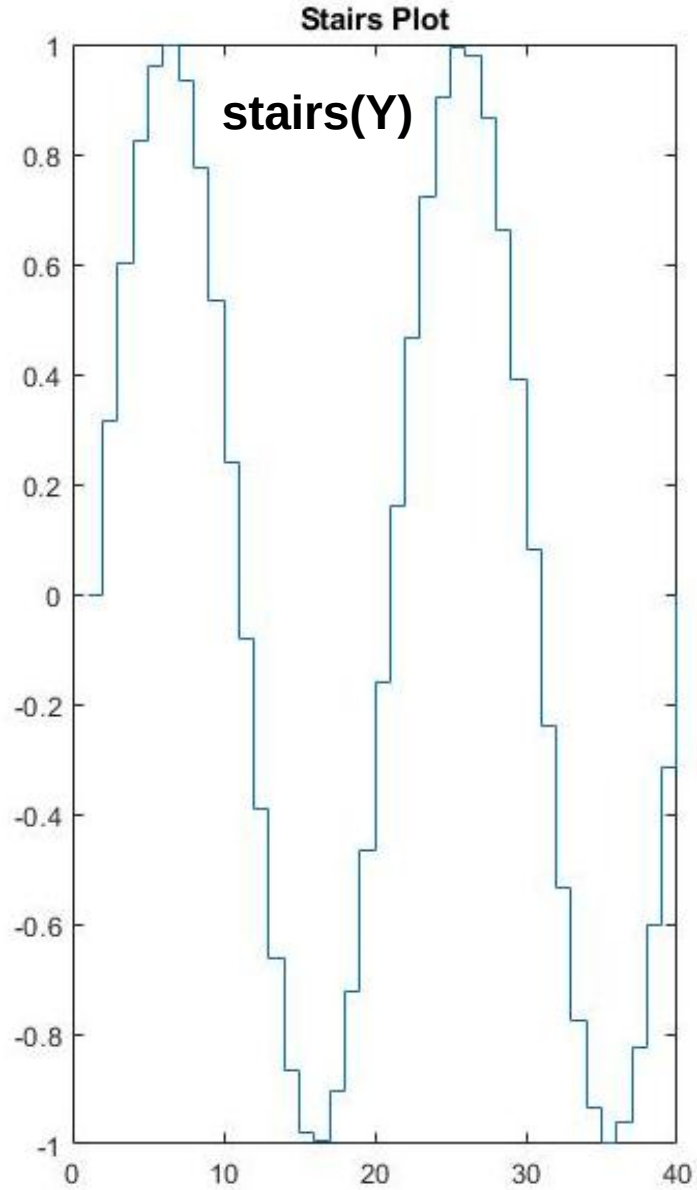
- Stairstep graph

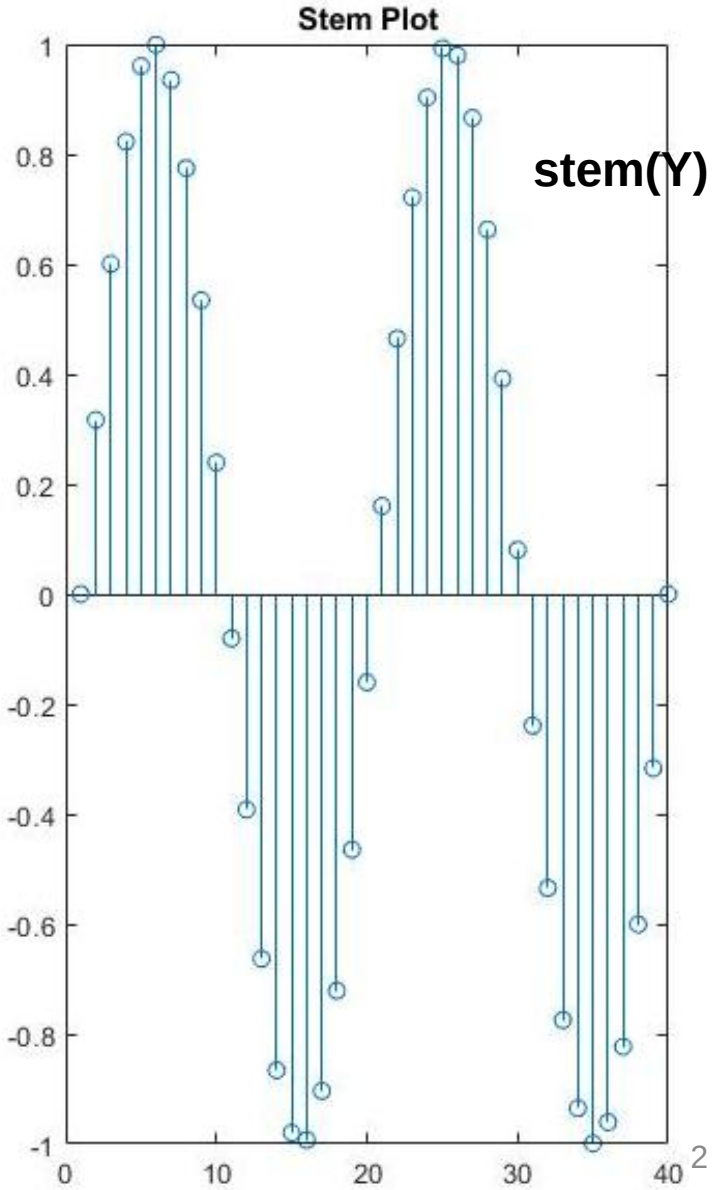
```
X = linspace(0,4*pi,40);
Y = sin(X);

figure

subplot(1,2,1);
plot(Y);title('Normal Plot’);

subplot(1,2,2);
stairs(Y);title('Stairs Plot');
```



Plot a stem graph







Log plots

loglog	Log-log scale plot
semilogx	Semilog plot (x-axis has log scale)
semilogy	Semilog plot (y-axis has log scale)

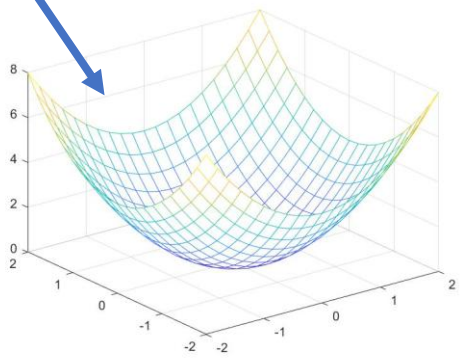
3-D plots

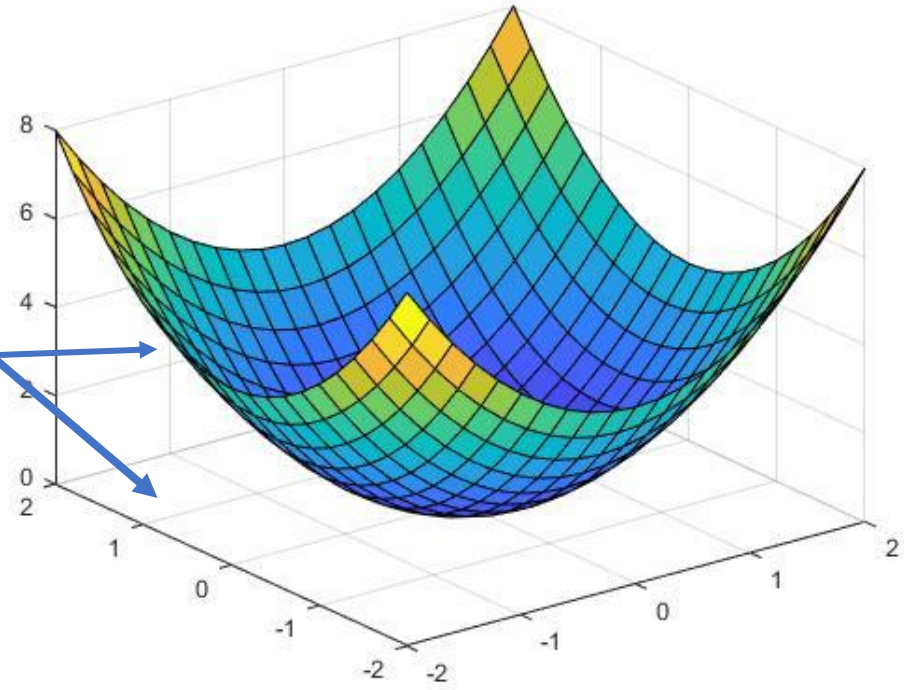
plot3	To plot simple 3D plot
bar3	To plot 3-D vertical bar graph
bar3h	To plot 3-D horizontal bar graph
pie3	To plot a 3-D pie plot
stem3	3-D stem plot
meshgrid	To give array for mesh grid
mesh	To plot wireframe mesh plot
surf	Surface plot
contour	To give contour of the given matrix
contour3	To give contour over the plane

Three-dimensional plots typically display a surface defined by a function in two variables, $z = f(x,y)$

To evaluate *z*, first create a set of (*x,y*) points over the domain of the function using **meshgrid**

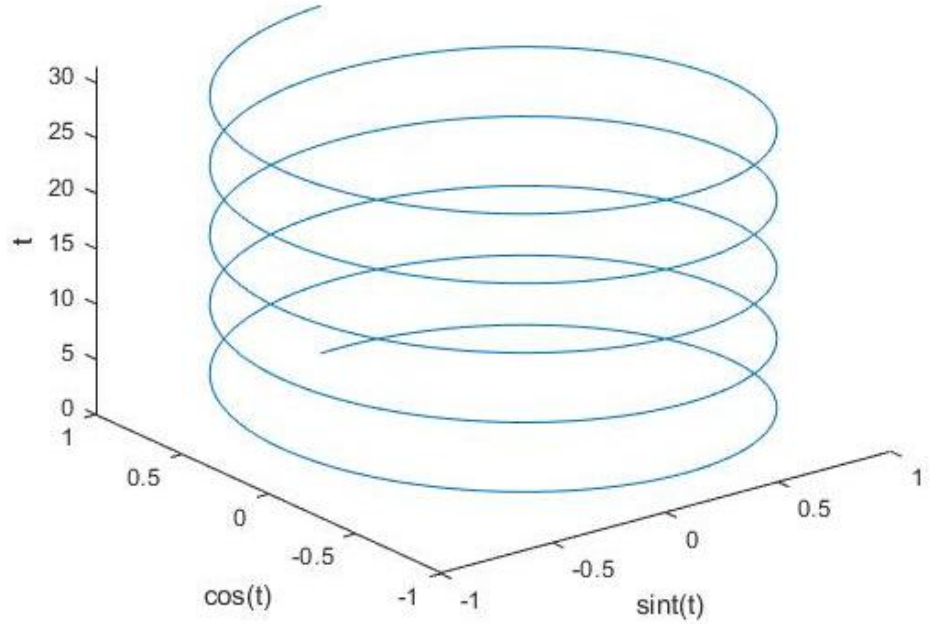
```
[x,y]= meshgrid(-2:0.2:2);
z=x.^2+y.^2;
figure
mesh(x,y,z);
figure
surf(x,y,z);
```





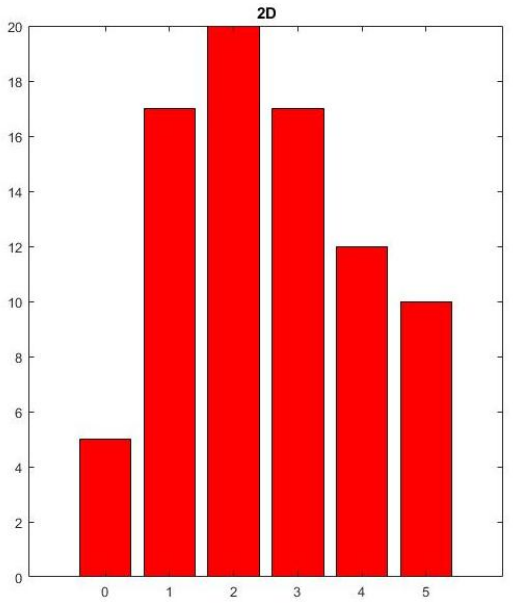
Plot 3-D Helix

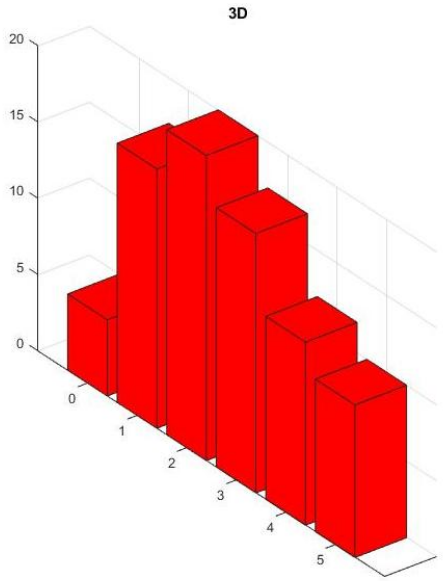
```
t = 0:pi/50:10*pi;
st = sin(t);
ct = cos(t);
plot3(st,ct,t);
xlabel('sint(t)');
ylabel('cos(t)');
zlabel('t');
```



2-D/ 3-D bar graph of data

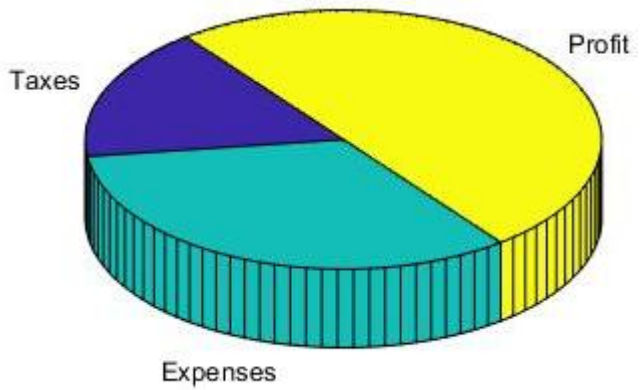
```
x= [0 1 2 3 4 5];
y=[5 17 20 17 12 10];
subplot(1,2,1);
bar(x, y, 'r'); title('2D');
subplot(1,2,1);
bar(x, y, 'r'); title('3D');
```





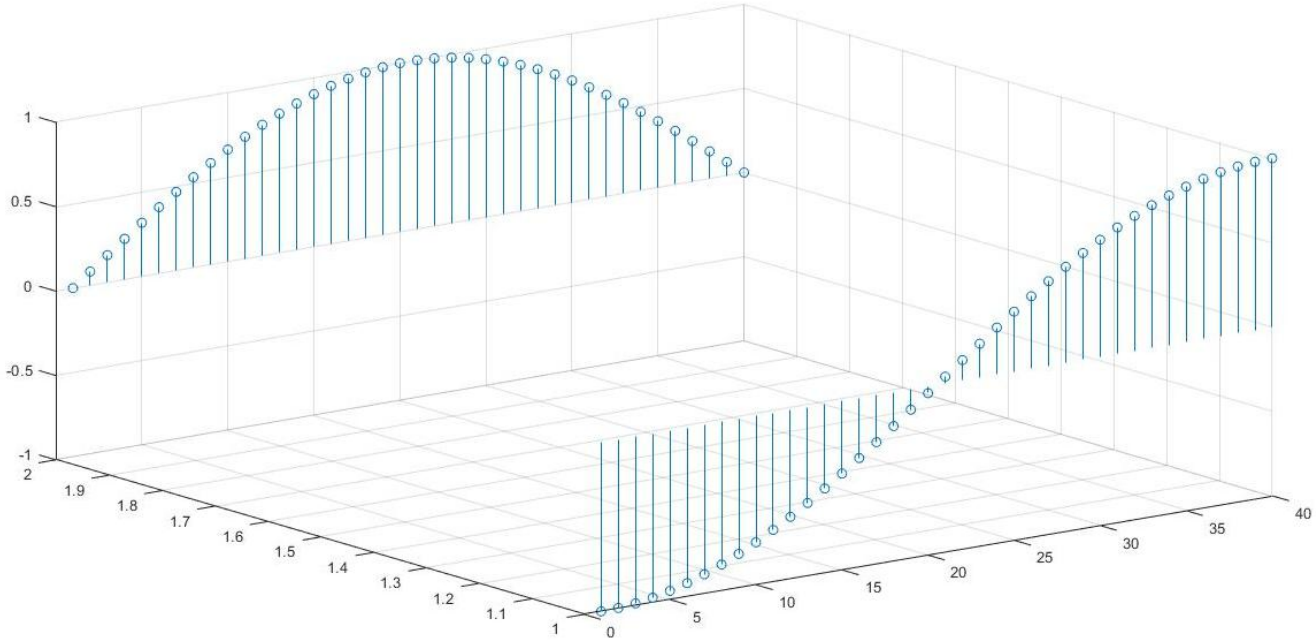
Create a 3-D pie chart and specify the text labels

```
x = 1:3;
labels = {'Taxes', 'Expenses', 'Profit'};
figure
pie3(x, labels);
```



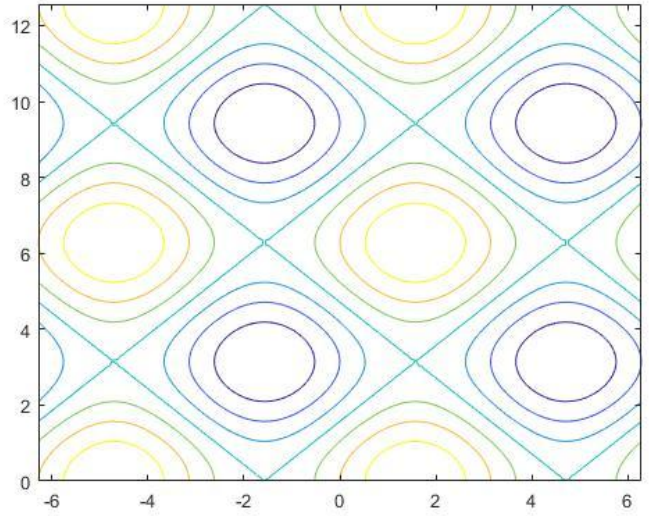
3-D stem plot of sine and cosine values between −π/2 and π/2 with a matrix input

```
figure
X = linspace(-pi/2,pi/2,40);
Z = [sin(X); cos(X)];
 stem3(Z)
```



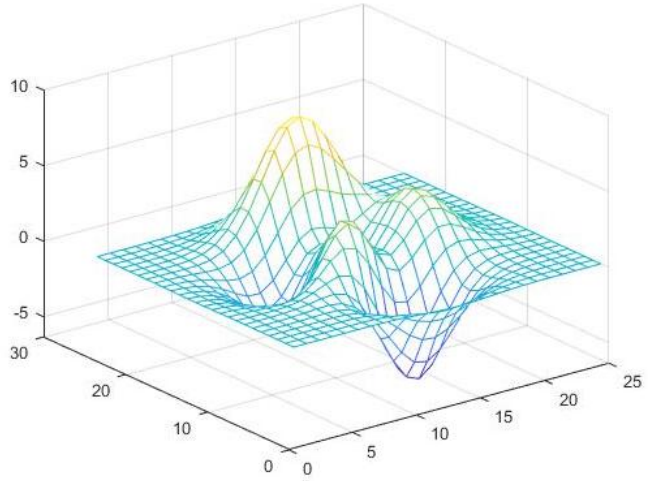
Contours of a Function

```
x = linspace(-2*pi,2*pi);
 y = linspace(0,4*pi);
[X,Y] = meshgrid(x,y);
 Z = sin(X)+cos(Y);
contour(X,Y,Z)
```



Mesh Plot - The mesh function creates a wireframe mesh. By default, the color of the mesh is proportional to the surface height

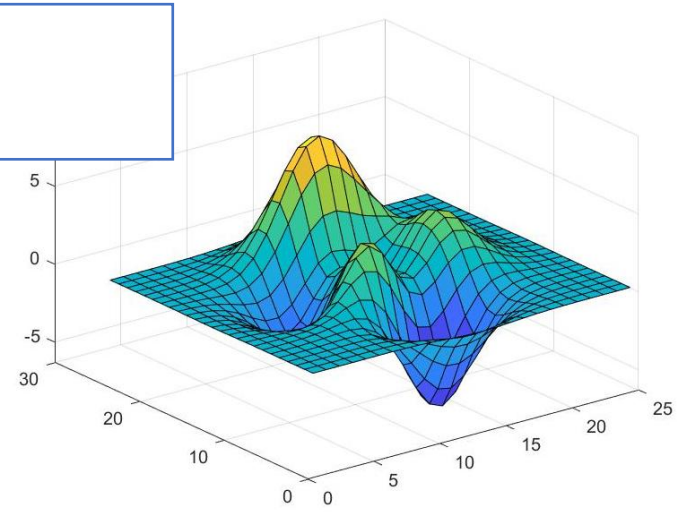
```
z = peaks(25);
 figure
mesh(z)
```



Surface Plot

The **surf** function is used to create a **3-D surface** plot.

```
figure
surf(z)
```



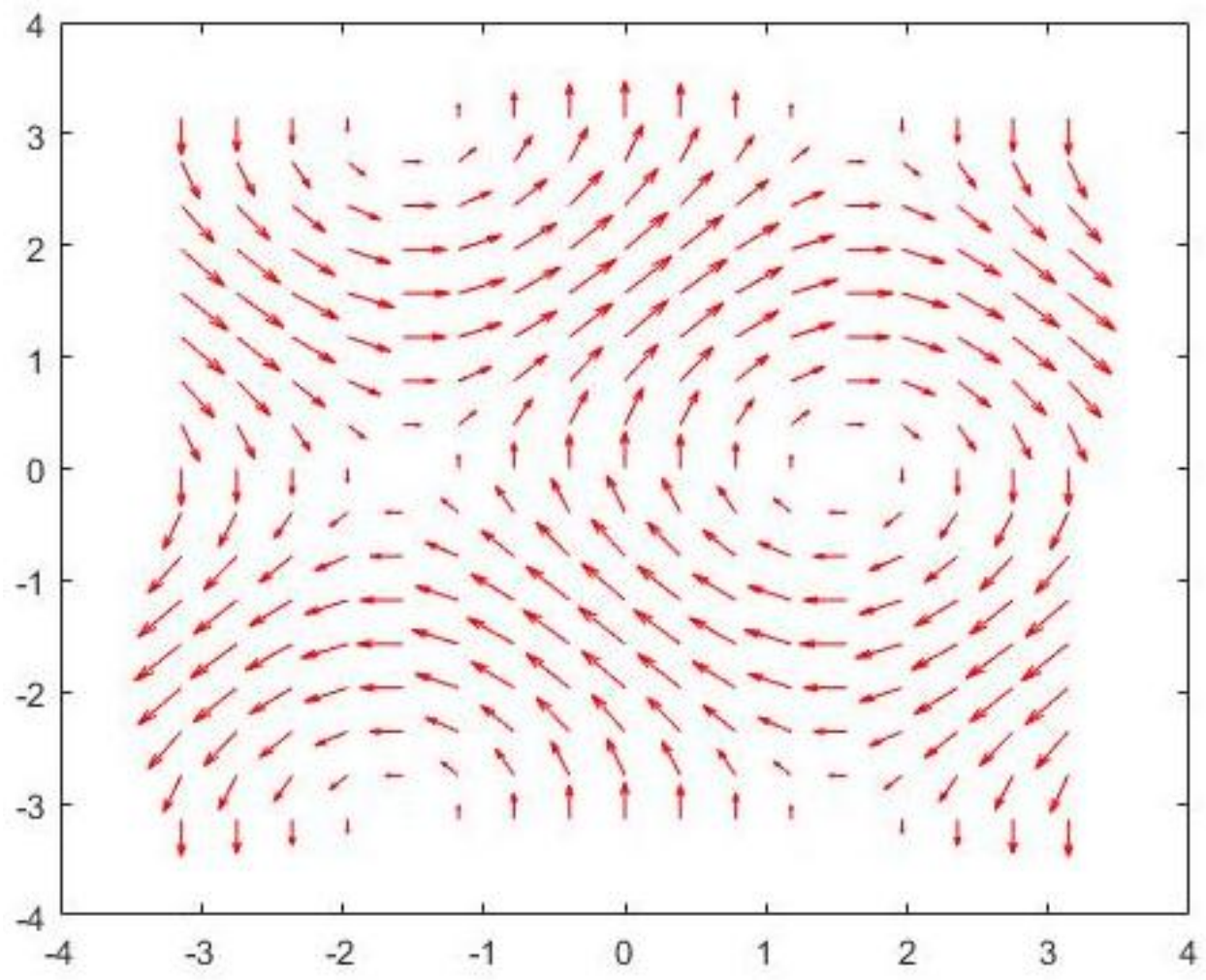
Quiver Plot

- The quiver function plots 2-D vectors as arrows

```
[X,Y] = meshgrid(-pi:pi/8:pi,-pi:pi/8:pi);
U = sin(Y);
V = cos(X);
quiver(X,Y,U,V,'r');
```

(x_pos, y_pos, x_dir, y_dir, color)

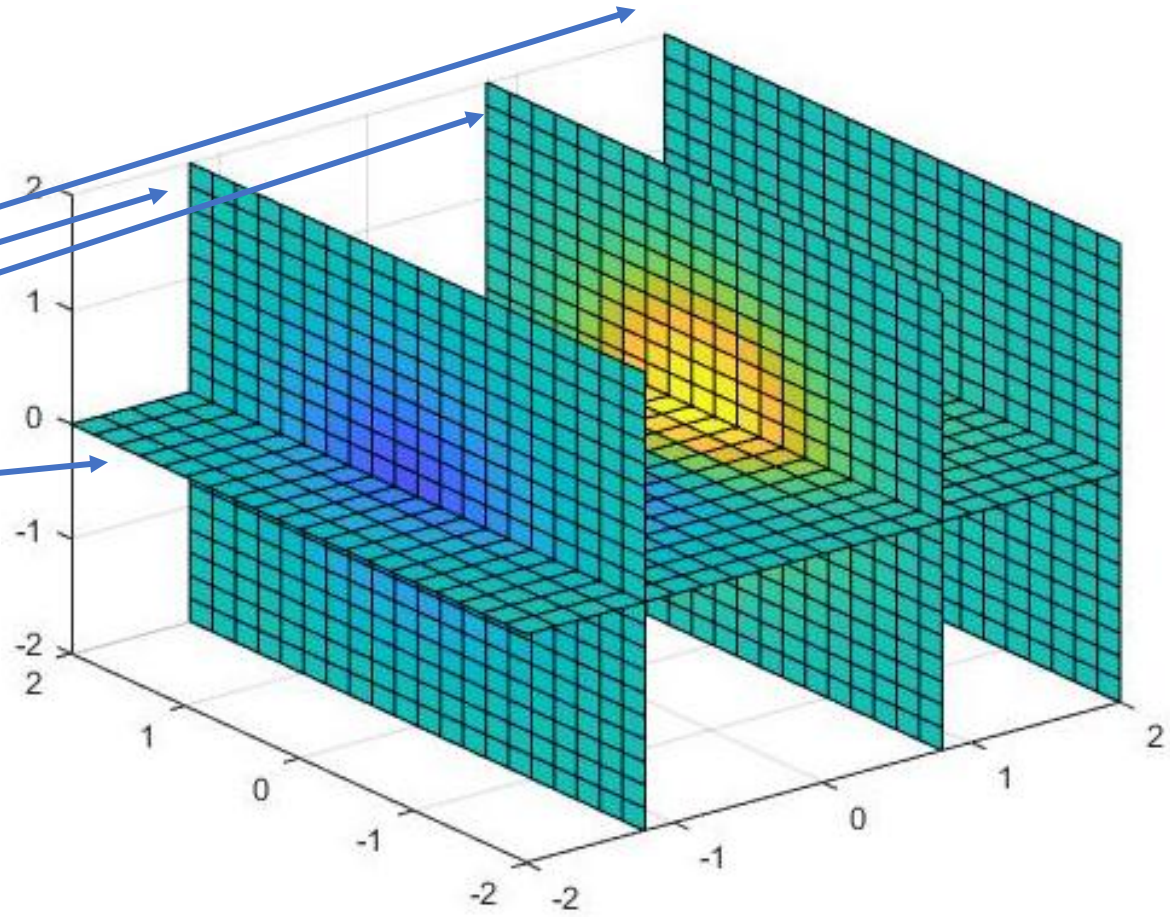
x_pos and **y_pos** are the starting positions of the arrow while **x_dir** and **y_dir** are the directions of the arrow.



Slices through 3-D Volumes

- The slice function displays data at planes that slice through volumetric data

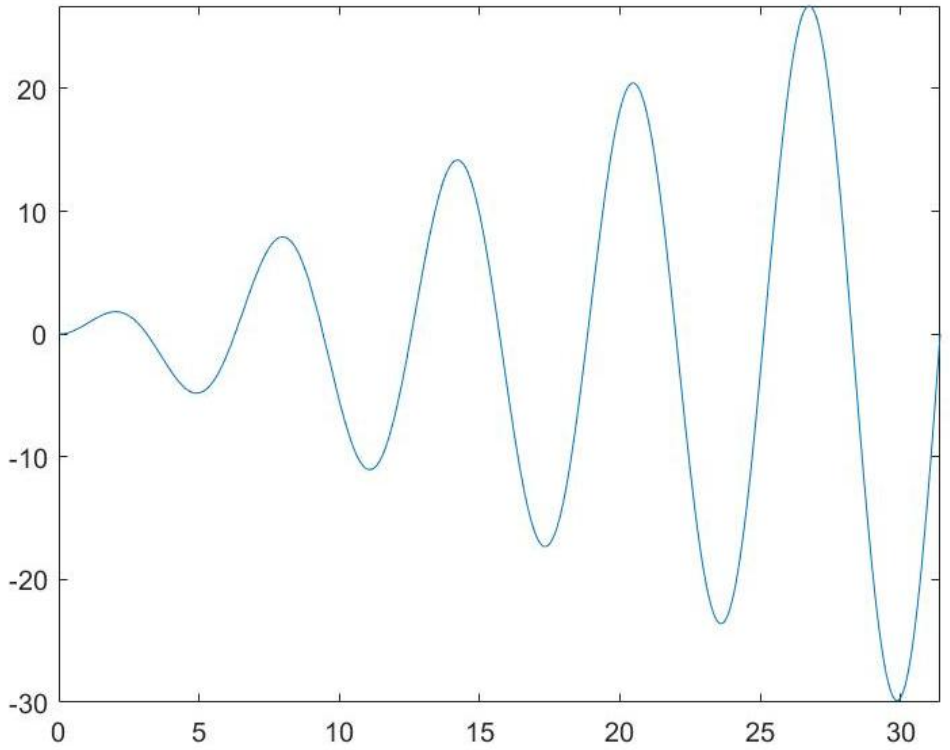
```
[X,Y,Z] = meshgrid(-2:.2:2);
V = X.*exp(-X.^2-Y.^2-Z.^2);

xslice = [-1.2,0.8,2];
yslice = [];
zslice = 0;
slice(X,Y,Z,V,xslice,yslice,zslice)
```



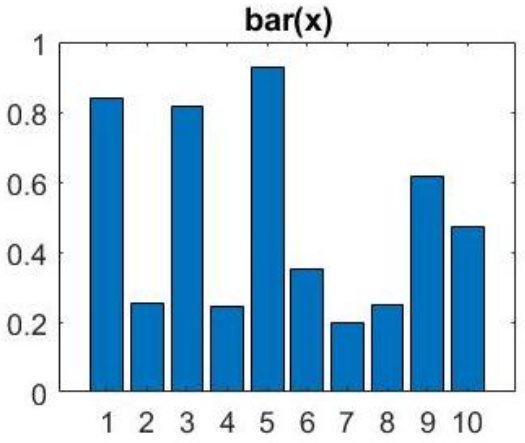

```
xslice  % location of y-z planes
yslice  % location of x-z plane
zslice % location of x-y planes
```

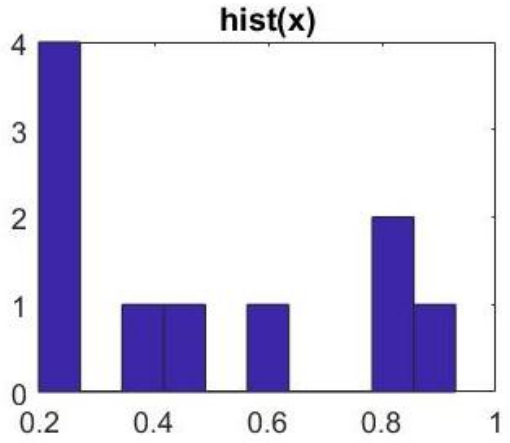
Plot expression or function - **fplot**

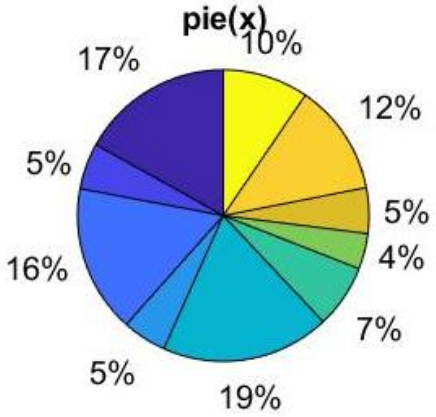
- Plot f(t)= tsint , 0 ≤ t ≤ T
- >> fplot(**@(x) x.*sin(x)**, [0 10*pi])
- fplot(f(x), **xinterval**) plots the curve
- defined by the function f(x)
- over the interval [xmin xmax] for x.

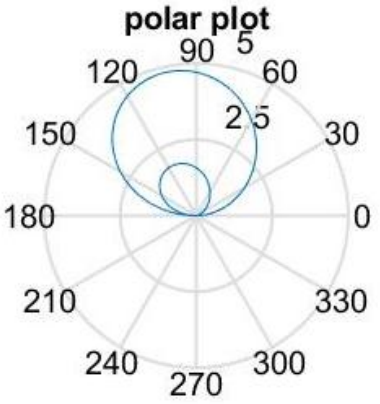


```
x=rand(10,1);

subplot(2,2,1);
bar(x);
title('bar(x)');

subplot(2,2,2);
hist(x);%group of data points organised within specified ranges
title('hist(x)');

subplot(2,2,3);
pie(x);
title('pie(x)');

subplot(2,2,4);
t= linspace(0, 2*pi, 200);
polar(t, t.*sin(t)); % polar plots magnitude and phase
title('polar plot');
```









Formatting and Annotation

Add labels, adjust colors, define axis limits, change line styles, add markers

Labels

title	Add title
subtitle	Add subtitle to plot
sgtitle	Add title to subplot grid
xlabel	Label x-axis
ylabel	Label y-axis
zlabel	Label z-axis
legend	Add legend to axes
bubblelegend	Create legend for bubble chart

Annotations

comment added to a text or diagram.

text	Add text descriptions to data points
gtext	Add text to figure using mouse
xline	Vertical line with constant x-value
yline	Horizontal line with constant y-value
annotation	Create annotations
datatip	Create data tip
line	Create primitive line
rectangle	Create rectangle with sharp or curved corners
texlabel	Format text with TeX characters
ginput	Identify axes coordinates

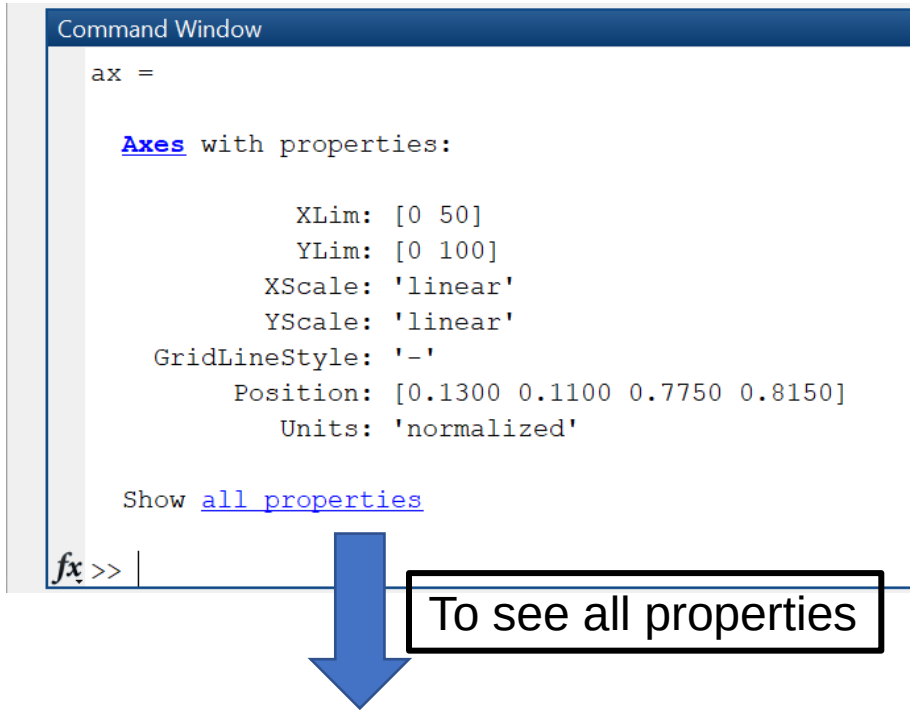
Axis control

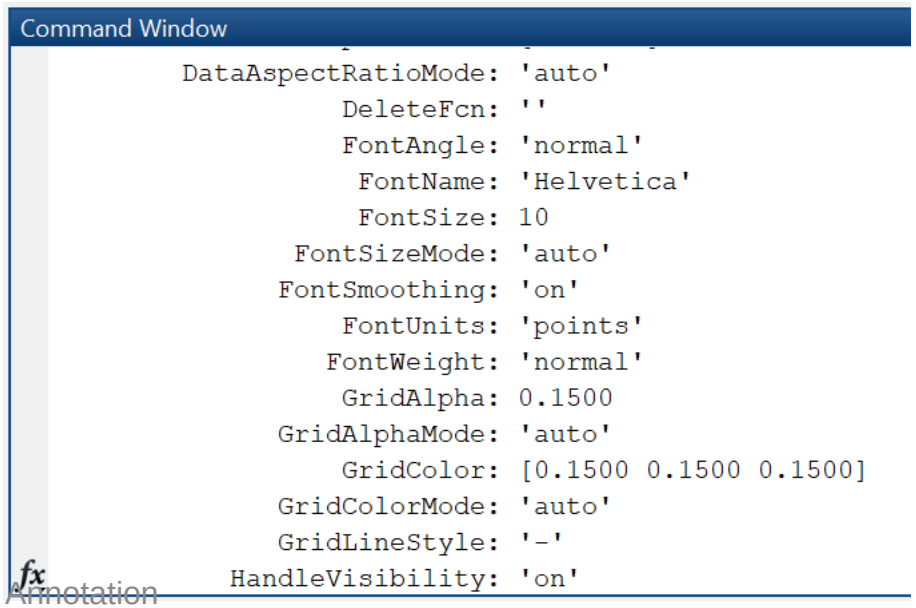
- Defines axis limits for plots

- >>axis([**xmin xmax ymin ymax**])
- % set the x axis from xmin to xmax
- % set the y axis from ymin to ymax

Change Font Size

- Axes **objects** have properties that you can use to **customize** the **appearance** of the axes
- Access the **current Axes object** using the **gca** function
- Then use dot notation to set the **FontSize** property
- >>**ax** = gca;
- >>**ax**.FontSize = 13;



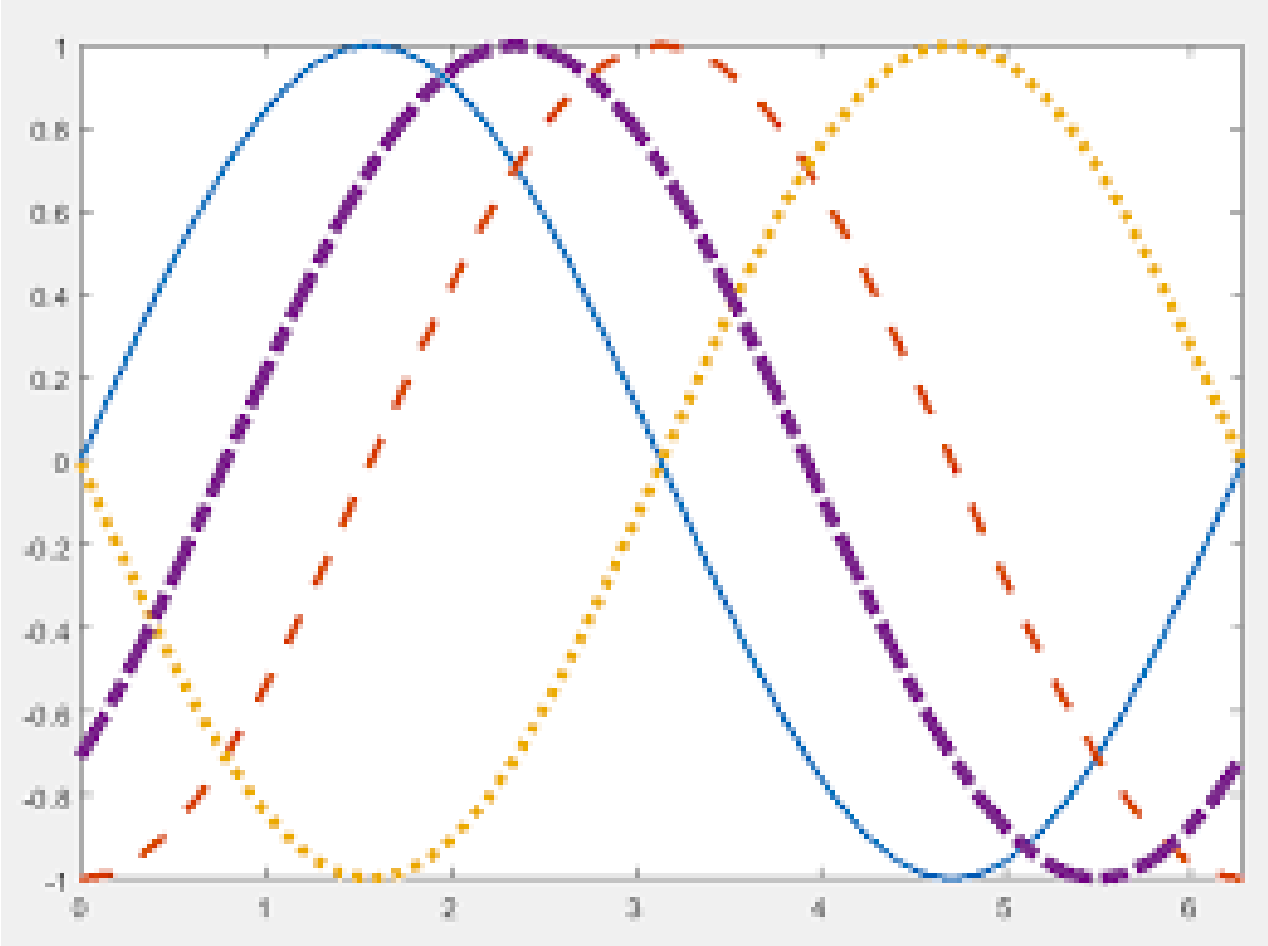


Change line appearance in plots

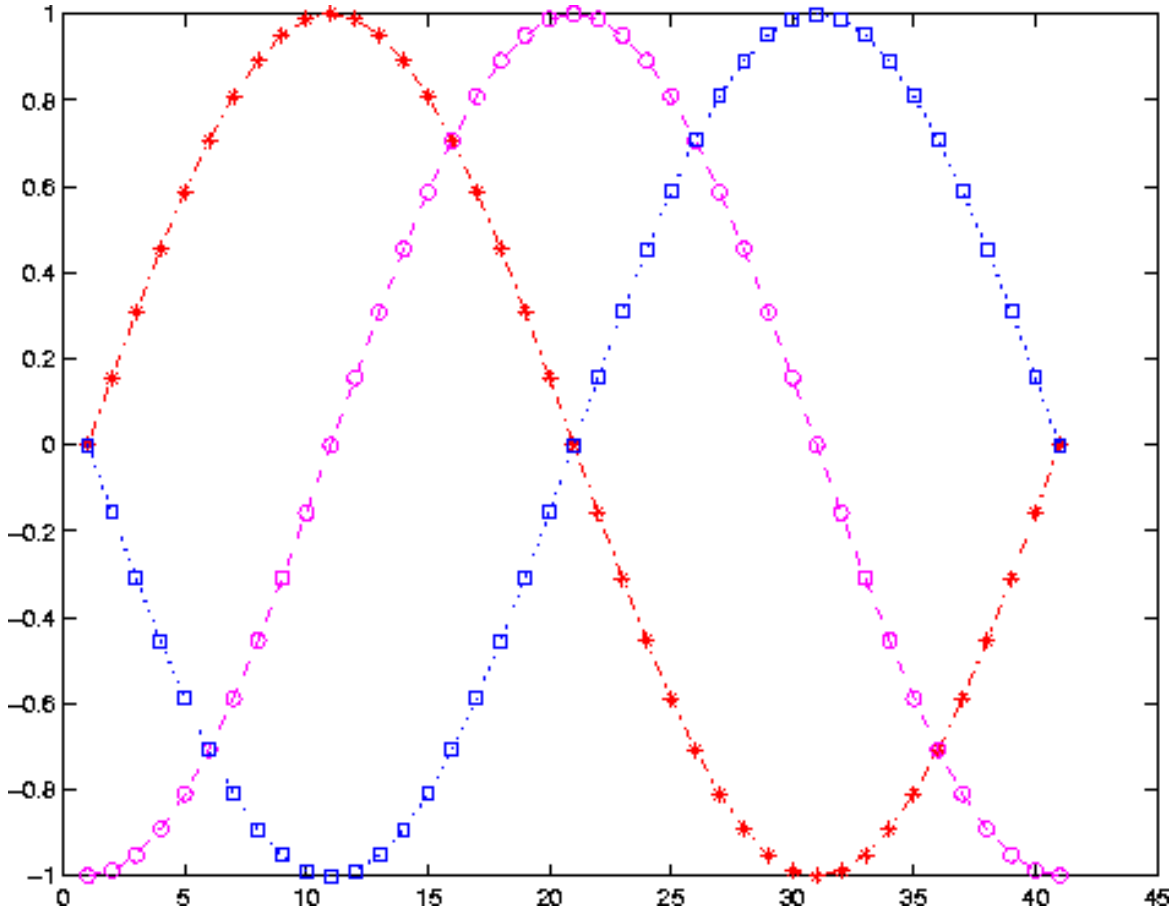
- Color and style of line can be changed by including an optional line specification when calling the plot function.
- Markers can be added in a similar way.

- Eg:
- ':' plots a dotted line.
- 'g:' plots a green, dotted line.
- 'g:*' plots a green, dotted line with star markers.
- '*' plots star markers with no line.

- The symbols can appear in any order.



Different Line Styles



Different Marker Styles

Specify Line Style, Color, and Marker

Color		Marker		Style	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

MarkerIndices

- Indices of data points at which to display markers, specified as a vector of positive integers
- If you do not specify the indices, then MATLAB displays a marker at every data point

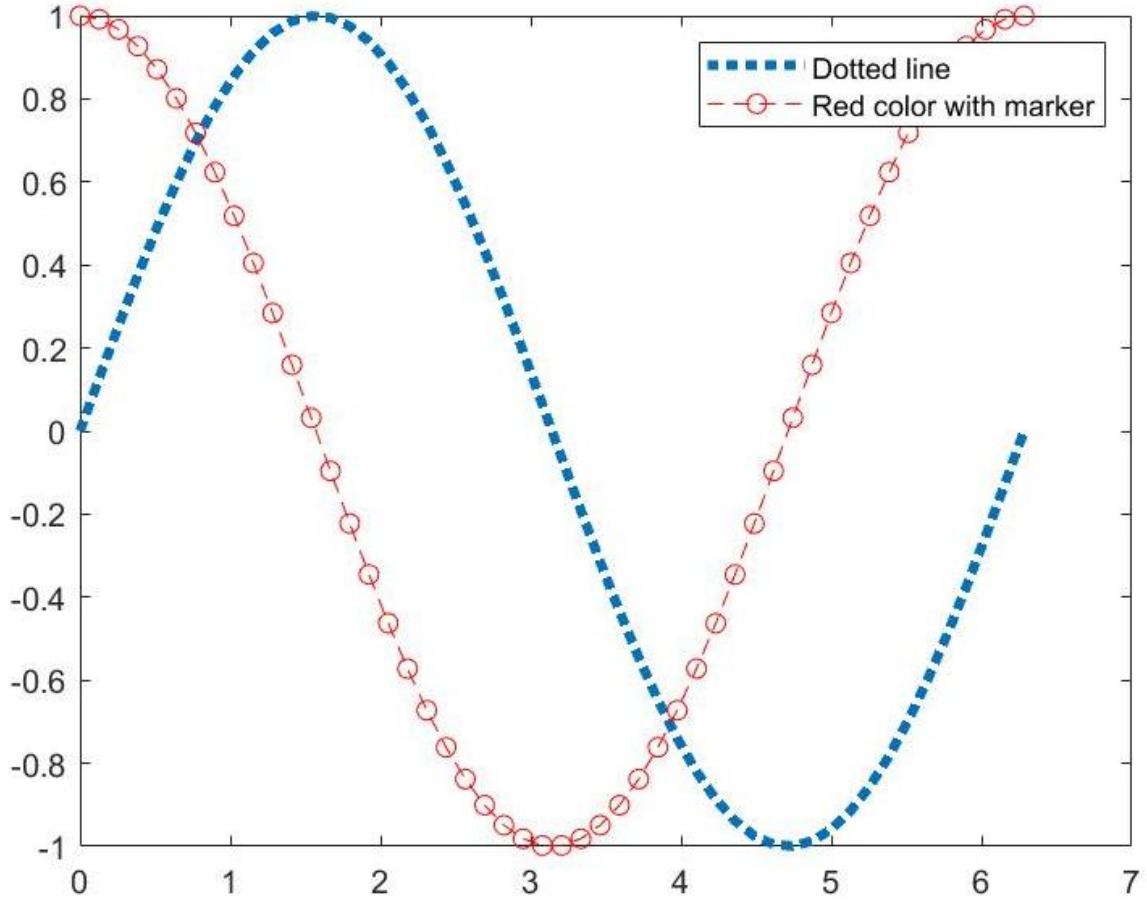
- >>plot(x, y, '-o', **'MarkerIndices**', [1 5 10])

- %displays a circle marker at the first, fifth, and tenth data points.

Example

- Plot a dotted line
- Add a second plot that uses a dashed, red line with circle markers

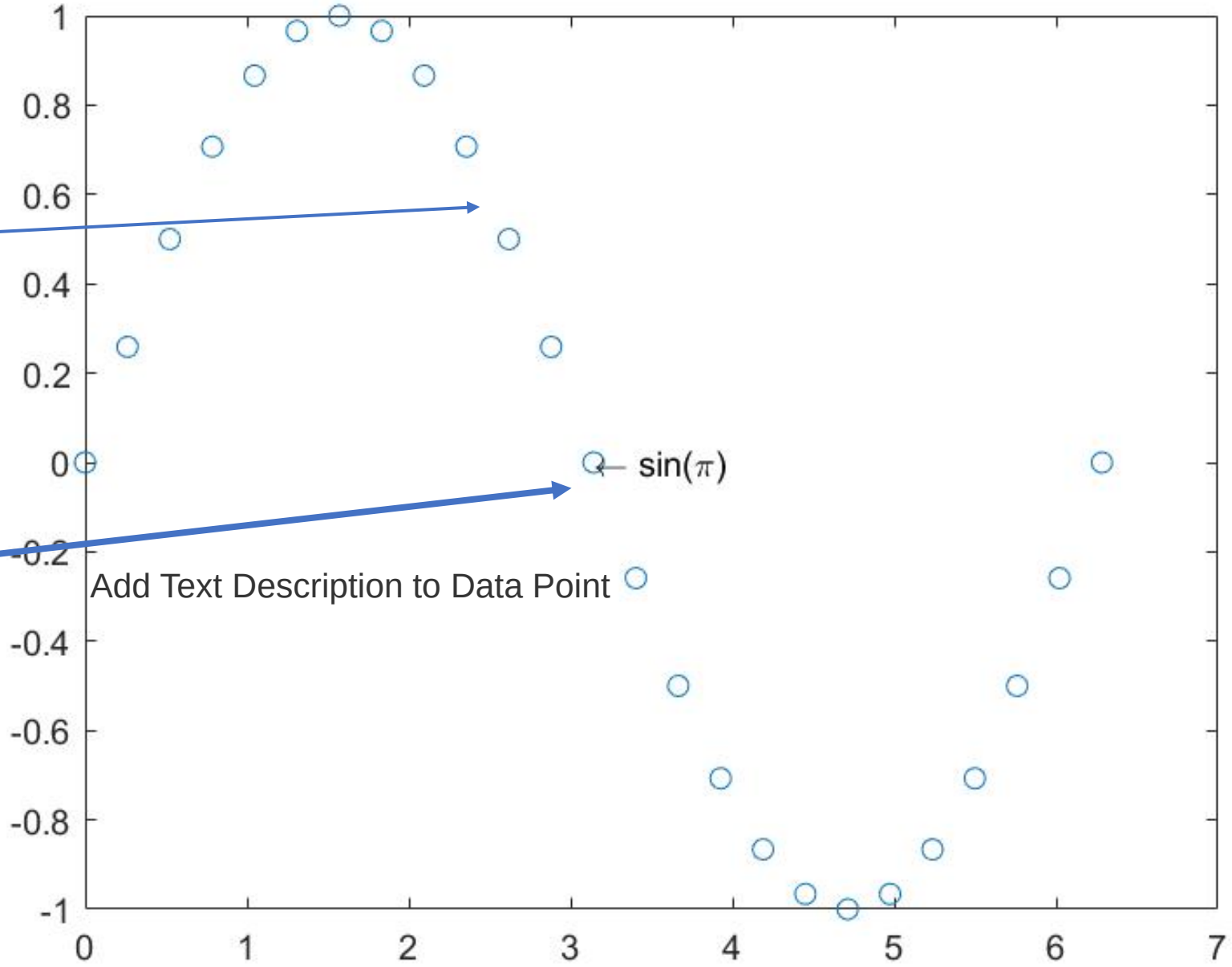
```
x = linspace(0,2*pi,50);
y = sin(x);

plot(x,y,':','Linewidth',3);
hold on

y2 = cos(x);
plot(x,y2,'--ro');
hold off

legend('Dotted line','Red color with marker');
```



Example

Plotting data points by omitting the line style option from the line specification

```
x = linspace(0,2*pi,25);
y = sin(x);
plot(x,y,'o');
text(pi,0,'\leftarrow sin(\pi)');
```



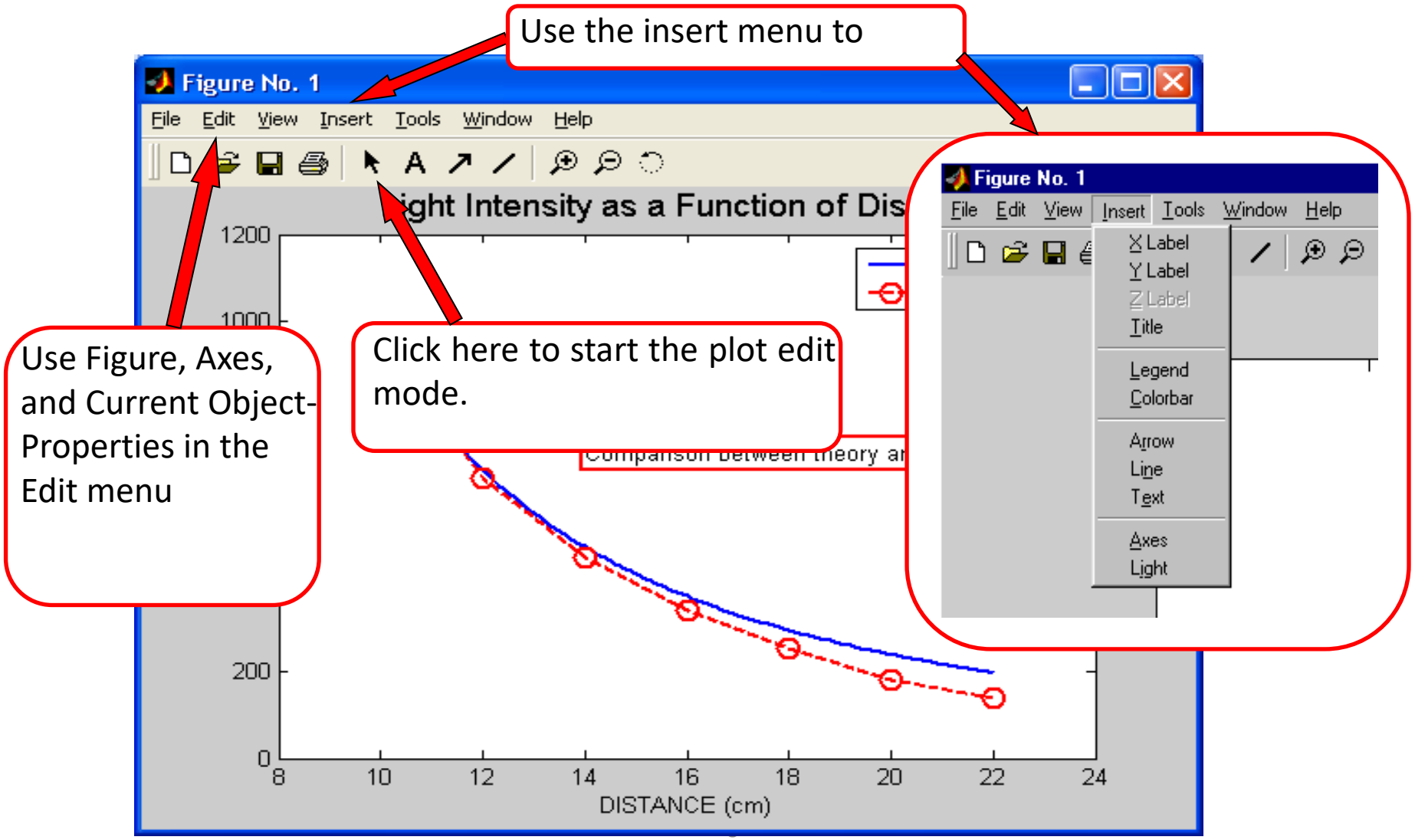
Change Line Object Properties

- Create a line plot.
- Assign the Line object created to the variable **ln**.
- The display shows commonly used properties, such as Color, LineStyle, and LineWidth.

```
ln.LineWidth = 2;
ln.Color = [0 0.5 0.5];
ln.Marker = 'o';
ln.MarkerEdgeColor = 'b';
```

Formatting a plot in the figure window

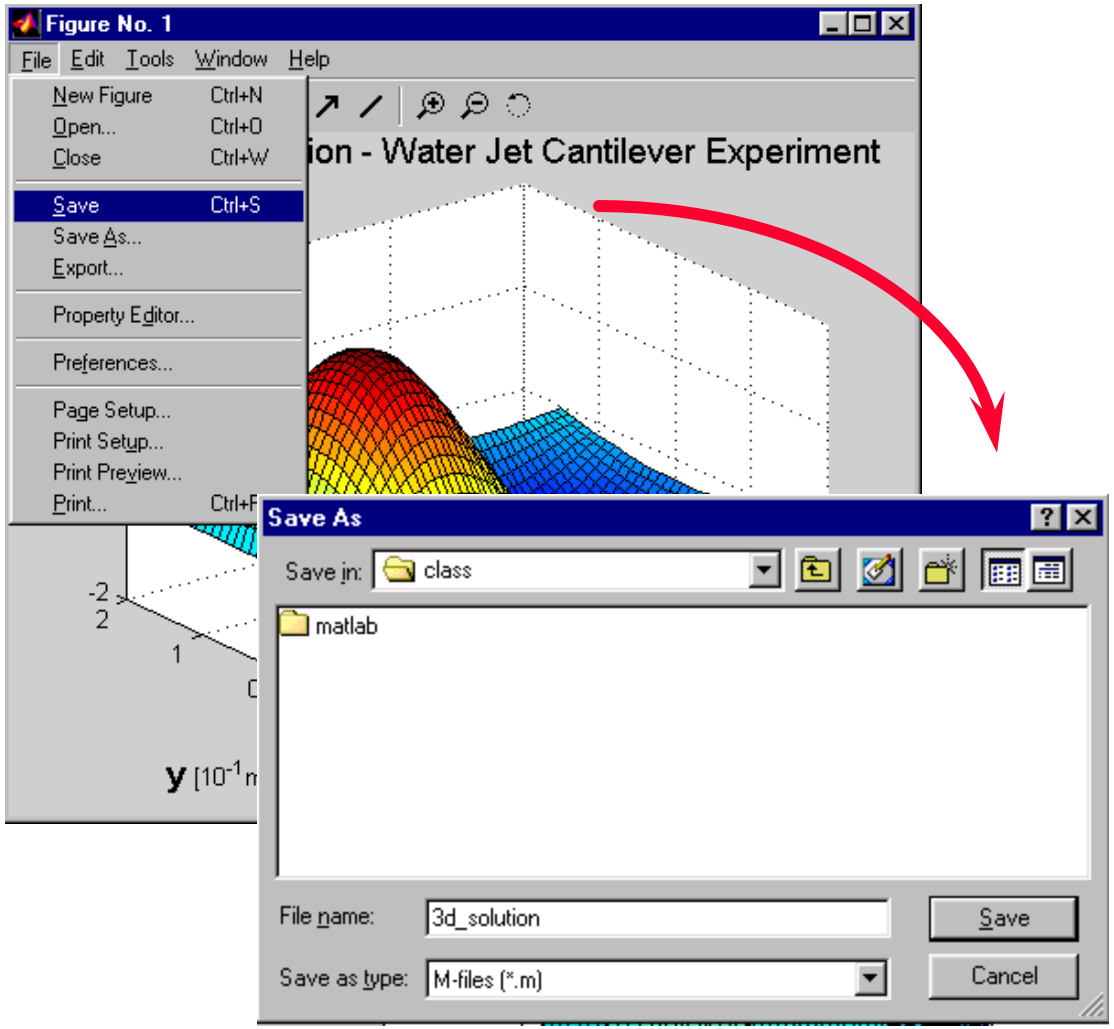
Once a figure window is open, the figure can be formatted interactively



Printing and Saving plots

Print/Save into different formats

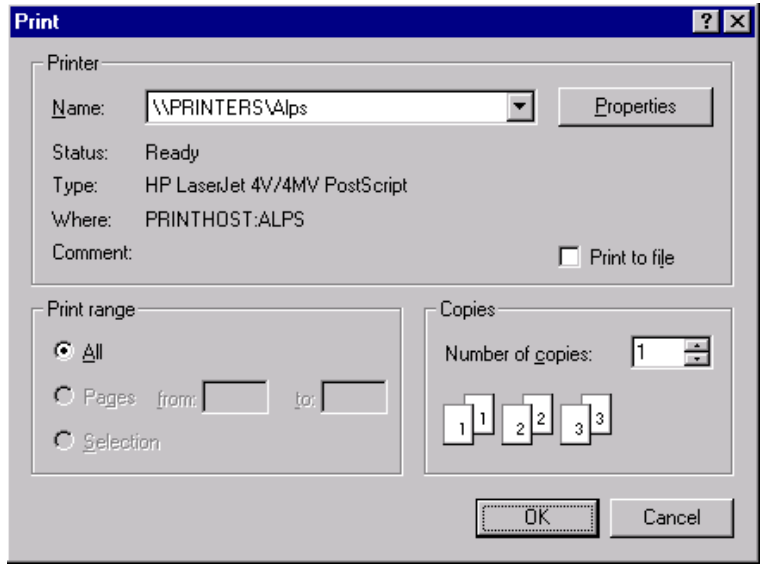
Saving Figures



Printing Figures

using the Dialog Box:
File Menu / Print...

>>printdlg



Export Axes as Image File

- **exportgraphics** - Save plot or graphics content to file

- Create a line plot and get the current axes. Then save the contents of the axes as a **JPEG** file.

- >> plot(rand(5,5))
- >> ax = gca;
- >> exportgraphics(ax,'**LinePlot.jpg**');

Specify Image Resolution

- Display an image and get the current axes. Then save the contents of the axes as a 300-DPI JPEG file.

- >> I = imread('peppers.png’);
- >> imshow(I);
- >> ax = gca;
- >> exportgraphics(ax,'Peppers300.jpg','**Resolution**',**300**);

Export Figure

- Display a plot with an annotation that extends beyond the bounds of the axes.
- Save the contents of the figure as a PDF file.

- >> plot(1:10);
- >> annotation('textarrow',[0.06 0.5],[0.73 0.5],'String','y = x ‘);
- >> f = gcf;
- >> exportgraphics(f,'**AnnotatedPlot.pdf**’);

Export Multipage PDF

- Create a line plot and save the contents of the axes to the file myplots.pdf

- **>> plot([0 0.3 0.1 0.6 0.4 1]);**
- **>> ax = gca;**
- **>> exportgraphics(ax,'myplots.pdf’);**

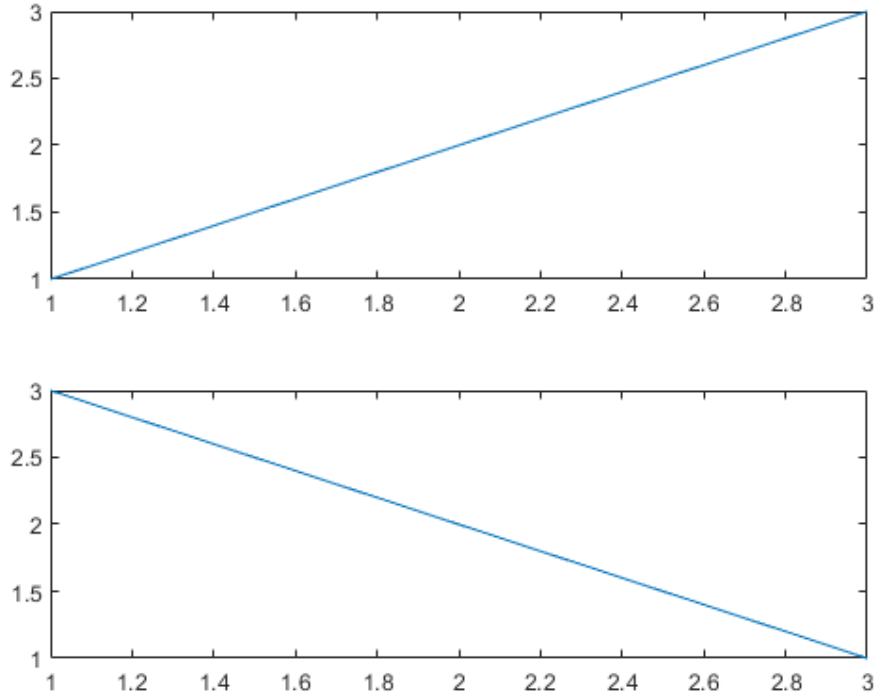
- Next, create a bar chart and save the contents of the axes as a second page in myplots.pdf.

- **>> bar(1:10);**
- **>> exportgraphics(ax,'myplots.pdf','Append',true);**

Export Tiled Chart Layout

- Display two plots in a tiled chart layout. Then save both plots as a PDF by passing the TiledChartLayout object to the exportgraphics function.

- >> t = tiledlayout(2,1);
- >> nexttile
- >> plot([1 2 3])
- >> nexttile
- >> plot([3 2 1])
- >> exportgraphics(t,'Layout.pdf')



Saving and opening FIG-File

- Save Current Figure to FIG-File
- **>> figure**
- **>> surf(peaks)**
- **>> savefig('PeaksFile.fig’)**

- To open the saved figure
- **>> openfig('PeaksFile.fig');**

saveas

- Save figure to **specific file format** defined by the file **extension**.
  - If you do not specify an extension, then saveas saves the figure to a **FIG-file**. To save the current figure, specify fig as **gcf**.

- Create a bar chart and save it as a PNG file.
- >> x = [2 4 7 2 4 5 2 5 1 4];
- >> bar(x);
- >> **saveas**(gcf,'Barchart.png’);

print

- Print figure or save to specific file format

- Create a plot and save it as a PNG image file.

- >> bar(1:10)
- >> print('BarPlot','-dpng’)

- Use **'-djpeg'** for JPG and **'-dpdf'** for full page Portable Document Format (PDF) in color

Symbolic computation

creating symbolic variables, expressions, math functions

Symbolic Math Toolbox

- In computation mathematics, there is **numeric** computation and **symbolic** computation

- MATLAB provides functions for solving, plotting, and manipulating symbolic math equations

- Ordinary Differential Equations
- Calculus
- Linear Algebra
- Equation Manipulation
- Equation Simplification

Numeric or Symbolic Arithmetic

- In **numeric** arithmetic, you represent numbers in **floating-point** format using either **double precision** or **variable precision**.
- In **symbolic** arithmetic, you represent **numbers in their exact form**.

Double-Precision Arithmetic

```
>>x = 10001/1001
>> y = pi
>> z = sqrt(2)
```

The results are converted to double-precision values.

x = 9.9910
y = 3.1416
z = 1.4142

Speed – **Fast**, Memory Usage - **Least**

Symbolic Arithmetic

```
x = sym(pi)
y = sqrt(sym(2))
```

Express the irrational numbers π and √2 in symbolic form.

x =pi

y = 2^(1/2)

Speed – **Slow**, Memory Usage - **Greatest**

Create Symbolic Numbers

>> sym(1/3) - **1/3**

Creating a Symbolic Expression

```
>> phi = (1 + sqrt(sym(5)))/2

phi = 5^(1/2)/2 + 1/2
```

Creating a Symbolic Matrix

```
>> syms a b c
>> A = [a b c; b c a; c a b]

A =

[a, b, c]
[b, c, a]
[c, a, b]
```

Create Matrix of Symbolic Numbers

```
A= [ 0.5 0.25;0.75 0.5]
sym(A)
```

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{2} \end{pmatrix}$$

Create Symbolic Variables

```
syms x   or syms a b c  or y=sym('y')
```

Creating Symbolic Math Functions

```
syms x y z
r = sqrt(x^2 + y^2 + z^2)
t = atan(y/x)
f = sin(x*y)/(x*y)
```

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$t = \operatorname{atan}\left(\frac{y}{x}\right)$$

$$f = \frac{\sin(x\,y)}{x\,y}$$

Create Symbolic Functions

```
syms f(x,y)
f(x,y) = x^2*y
f(3,2)
```

f(x, y) = $x^2 y$

Ans=18

Solve systems of equations

Solve Algebraic Equations with One Symbolic Variable

```
syms x
solve(x^3 - 6*x^2 == 6 - 11*x)
```

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

To solve the equation, $5x^2 + 6x + 3 = 10$

```
syms x
solve(5*x^2+6*x-7,0)
```

$$\begin{pmatrix} -\frac{2\sqrt{11}}{5} - \frac{3}{5} \\ \frac{2\sqrt{11}}{5} - \frac{3}{5} \end{pmatrix}$$

OR

```
syms x
e2= sym(5*x^2 + 6*x + 3== 10)
solve(e2)
```

```
syms a b c x
S = a*x^2 + b*x + c;
solve(S)
```

$$\begin{pmatrix} -\frac{b+\sqrt{b^2-4ac}}{2a} \\ -\frac{b-\sqrt{b^2-4ac}}{2a} \end{pmatrix}$$

Solve Systems of Algebraic Equations

```
syms x y z
one = sym(3*x + 2*y - z == 10);
two = sym(-x + 3*y + 2*z == 5);
three = sym(x - y - z == -1);
answer = solve(one,two,three)
answer.x
answer.y
answer.z

OR

[x y z]=solve(one, two, three)
```

X = -2
Y =5
Z = -6

factor(S)- Factors the expression or equation

```
syms x
factor(x^3-1)
```

$$(x-1 \quad x^2+x+1)$$

```
syms x
 g = x^3 + 6*x^2 + 11*x + 6;
factor(g)
```

$$(x+3 \quad x+2 \quad x+1)$$

expand(f)-rewrite a polynomial in the standard form

```
syms x
n= (x+ x^2+ 2* x+ 2)*(x+3)
expand(n)
```

n= $(x+3)\ (x^2+3x+2)$

$$x^3+6x^2+11x+6$$

subs-Substitutions in Symbolic Expressions

```
syms x
f = 2*x^2 - 3*x + 1;
subs(f, 1/3)
```

$$\frac{2}{9}$$

simplify(y)- Simplify Symbolic Expressions

```
syms x
n= (x+ x^2+ 2* x+ 2)*(x+3)
simplify(n)
```

$$(x+3)\ (x^2+3x+2)$$

Compare the programs

```
clc, clear
syms x
y1=(x+1)^2
y2=x^2+2*x+1
if(y1==y2)
disp('equal')
else
disp('notequal')
end
```

notequal

```
clc, clear
syms x
y1=(x+1)^2
simplify(y1)
y2=x^2+2*x+1
simplify(y2)
if(simplify(y1)==simplify(y2))
disp('equal')
else
disp('notequal')
end
```

equal

Reg. No.									

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

IV SEMESTER B. TECH
SPECIAL END SEMESTER EXAMINATION, NOVEMBER 2024
MATLAB FOR ENGINEERS [ELE 4303]

Time: 3 Hours **Date: 28, NOVEMBER 2024** **Max. Marks: 50**

Instructions to Candidates:

- Answer **ALL** the questions.
- Missing data may be suitably assumed.

Q No.	Question	Marks
1A.	What is the difference between a script and a function in MATLAB, and when would you use each?	***03***
1B.	Write a MATLAB code to create a 1x10 array using colon operator (:), in which the first five numbers are even (between 2 and 10) and last five numbers are odd (between 1 and 9).	***03***
1C.	List any FOUR features of MATLAB that make it particularly useful for engineers in fields such as control systems, signal processing, and data analysis?	***04***
2A.	Write a MATLAB **"for loop"** to iterate through the array A = [2, 4, 6, 8, 10] and display each element multiplied by 3.	***03***
2B.	MATLAB function: **sortrows**(x,n) - Sorts the rows in a matrix 'x' on the basis of the values in column 'n'. The following table gives the finishing time of different skaters in a skating race. Skater Number | 1 | 2 | 3 | 4 | 5 Time (min) | 42 | 43 | 41 | 40 | 45 Write a MATLAB code to sort the table in ascending order, based on the race finishing time.	***03***
2C.	Write a **RECURSIVE** function to search for an element in an unsorted array. Obtain its **TIME COMPLEXITY**.	***04***

3A.	Match the command with its correct description: **Command** 1. plot(x, y, 'r--') 2. plot(x, y, 'LineWidth', 2) 3. plot(x, y, 'o') 4. xlabel('Time (s)') 5. title('Sample Plot') 6. legend('Data 1', 'Data 2')	**Options** A. Adds a title to the plot. B. Plots x and y as a red dashed line. C. Sets the thickness of the line in the plot. D. Plots x and y as circular markers. E. Adds a label to the x-axis. F. Adds a legend for multiple data sets in the plot.	***03***

3B.	The following is the code plot to sinewave in MATLAB:
	
	freq = 0.2*pi;
	w = 2*pi*freq;
	A = 1.25;
	t = linspace(0,pi,10000);
	ft = A * sin(w*t);
	plot(t, ft), title("Sine Wave"),
	xlabel("Time"), ylabel("Sine wave function")
	
	Modify the code to plot only positive values.
			03

3C.	Consider the following code in MATLAB:
	x = '10';
	y = 5;
	z = x + y;
	
	Is the above code VALID or INVALID? Give suitable reasons. If INVALID suggest a way in which the code can be corrected.
			04

4A.	Consider the following MATLAB code:
	
	syms x
	f = x^3 + 4*x^2 - 10*x + 5;
	f_prime = diff(f, x);
	f_double_prime = diff(f_prime, x);
	roots = solve(f == 0, x);
	disp(f_prime);
	disp(f_double_prime);
	disp(roots);
	
	Give the output of each display command after it is executed.
			03

			03

4B. Obtain a smooth sine curve using the following data points:

x	0	45	135	180	225	315	360
y	0	0.7071	0.7071	0	-0.7071	-0.7071	0

MATLAB functions:

y = **polyval**(p,x) evaluates the polynomial p at each point in x. The argument p is a vector of length n+1 whose elements are the coefficients (in descending powers) of an nth-degree polynomial.

p = **polyfit**(x,y,n) returns the coefficients for a polynomial p(x) of degree n that is a best fit (in a least-squares sense) for the data in y. The coefficients in p are in descending powers, and the length of p is n+1.

4C. Example of solving linear equations in MATLAB is given below:

```
syms x y z
eqn1 = 2*x + y + z == 2;
eqn2 = -x + y - z == 3;
eqn3 = x + 2*y + 3*z == -10;
[x y z]=solve(eqn1, eqn2, eqn3);
```

In the similar way write MATLAB code to solve:

The ratio of two numbers is 3/5. If 4 is added to the first and 6 to the second, the ratio becomes the reciprocal of the original ratio. Find the numbers.

04

5A. Explain how MATLAB Simulink can be useful for modeling and simulating dynamic systems in engineering applications. Provide at least two specific examples of how Simulink is beneficial in these scenarios.

03

5B. Give the Simulink block diagram to obtain the step response of the following equation: $\dot{y} = 5f(t) - 7y$

03

5C. List any FOUR toolboxes in MATLAB. Elaborate on how they are useful in your field of work.

04

Reg. No.									

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

IV SEMESTER B.TECH (ELECTRICAL & ELECTRONICS ENGINEERING)

END SEMESTER EXAMINATIONS, MAY 2023

MATLAB for Engineers [ELE 4303]

REVISED CREDIT SYSTEM

Time: 3 Hours **Date: 02 JUNE 2023** **Max. Marks: 50**

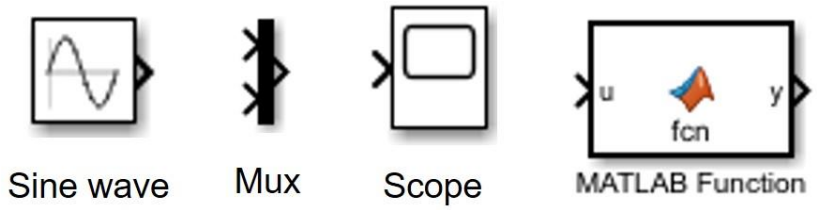
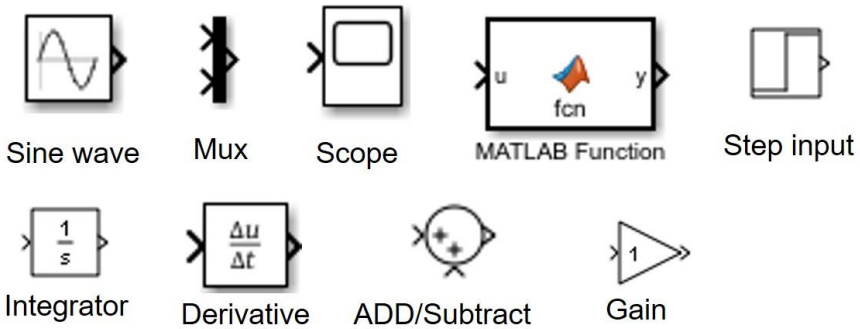
Instructions to Candidates:

- Answer **ALL** the questions.
- Missing data may be suitably assumed.
- Write MATLAB code wherever required.

Q.NO	Questions	Marks	CO	BTL
1A.	List any three typical applications of MATLAB.	(03)	01	01
1B.	$A = \begin{bmatrix} 3 & 7 & -4 & 12 \\ -5 & 9 & 10 & 2 \\ 6 & 13 & 8 & 11 \\ 15 & 5 & 4 & 1 \end{bmatrix}$ Array A is given as above, write MATLAB script to do the following: a) Create a 4×3 array **B** consisting of all elements in the second through fourth columns of **A.** b) Create a 3×4 array **C** consisting of all elements in the second through fourth rows of **A.** c) Find the array product such that **E=C*B**	(03)	01	03
1C.	Create a MATLAB script file to plot an astroid shape on the xy plane over the parametric interval $-2\pi \leq t \leq 2\pi$, where $x\_a = \cos^3(t)$ and $y\_a = \sin^3(t)$. Also, plot the catacaustic of the astroid shape on the same plot over the same range for t: $x = \frac{\cos(t)\,[8 + 5\cos(2t) + 3\cos(6t)]}{13 + 3\cos(4t)}$ $y = \frac{4\sin^3(t)\,[7 + 6\cos(2t) + 3\cos(4t)]}{13 + 3\cos(4t)}$	(04)	CO2	04

	Make sure to use enough points to create smooth curves. Provide a plot title, labels for the axes, and a legend for the two curves.			
2A.	What are classes and objects in MATLAB? What are the components of a 'class' in MATLAB? Explain a suitable example.	***(03)***	***CO1***	***02***
2B.	Create a MATLAB function file using a WHILE LOOP to plot the following function: $y = \begin{cases} sin(x) & for\ 0 \le x < \pi \\ -0.81057x^2 + 7.63944x - 16 & for\ \pi \le x < 2\pi \\ -1.6211x^2 + 25.465x - 96 & for\ 2\pi \le x \le 3\pi \end{cases}$ Illustrate the use of the function within a MATLAB script file.	***(03)***	***CO2***	***04***
2C.	Write a pseudocode RECURSIVE algorithm to reverse the contents of a given array. Illustrate how the contents of a given array can be reversed using basic MATLAB operators.	***(04)***	***CO2***	***04***
3A.	What is the primary data type in MATLAB? How are all numeric variables stored in MATLAB? How to choose the best data type for a given application?	***(03)***	***CO3***	***03***
3B.	Given a complex number C = 10+5i. Convert it to its equivalent polar form using MATLAB functions like abs(__), angle(____), real(__), imag(____), cos(__), sin(__), tan(__), acos(____), asin(__) and atan(__). Multiply C with itself in its polar form and convert the product to its equivalent rectangular form using only MATLAB functions.	***(03)***	***CO3***	***03***
3C.	Plot the values of sine and cosine function for 100 evenly spaced points between from 0 to 3pi the same figure window using SUBPLOT command. Illustrate the use of different color and line styles in the plots.	***(04)***	***CO3***	***03***
4A.	Solve the following puzzle using MATLAB symbolic computation code.	***(03)***	***CO4***	***04***

	Mrs. Benevides leaves Burbank at 9 a.m. and drives west on the Ventura Freeway at an average speed of 50 miles per hour. Ms. Twill leaves Burbank at 9:30 a.m. and drives west on the Ventura Freeway at an average speed of 60 miles per hour. At what time will Ms. Twill overtake Mrs. Benevides, and how many miles will they each have gone? Show necessary equations for solving the puzzle. Write MATLAB symbolic computation code for estimating the time at which Ms. Twill will overtake Mrs. Benevides and how many miles both would have travelled before Ms. Twill overtaking Mrs. Benevides.			
4B.	Current versus voltage characteristics of a white light emitting diode is given in the table below. Current In mA: 5, 10, 20, 30, 40 Voltage in Volts: 1.85, 1.88, 1.93, 1.96, 2.00 Using 1-D data interpolation write a MATLAB code to find: 1) Voltage at 25mA. 2) Current at 1.90 Volts. Description of the MATLAB function for 1-D data interpolation (table lookup) is as follows: vq = **interp1**(x,v,xq) returns interpolated values of a 1-D function at specific query points using linear interpolation. Vector x contains the sample points, and v contains the corresponding values, v(x). Vector xq contains the coordinates of the query points.	***(03)***	***CO4***	***03***
4C.	Current versus voltage characteristics of a white light emitting diode is given in the table below. Current In mA: 5, 10, 20, 30, 40 Voltage in Volts: 1.85, 1.88, 1.93, 1.96, 2.00	***(04)***	***CO4***	***03***

	Using Polynomial curve fitting in MATLAB write a MATLAB code to fit the Current/voltage data to a second order polynomial. Using the polynomial fit find the: 1) Voltage at 25mA. 2) Current at 1.90 Volts. MATLAB functions for reference: y = **polyval**(p,x) evaluates the polynomial p at each point in x. The argument p is a vector of length n+1 whose elements are the coefficients (in descending powers) of an nth-degree polynomial. p = **polyfit**(x,y,n) returns the coefficients for a polynomial p(x) of degree n that is a best fit (in a least-squares sense) for the data in y. The coefficients in p are in descending powers, and the length of p is n+1.			
5A.	Using MATLAB/SIMULINK, 1. Write a MATLAB function to obtain a HALF wave rectified output from a given sine wave. 2. Connect the SIMULINK blocks given below to show both original and rectified output on the display. 	***(03)***	***CO5***	***03***
5B.	The following equation corresponds to transient response of a RC circuit. dVc(t) / dt = (1/RC) [V(t) – Vc(t)] Show the block diagram to simulate the equation using SIMULINK. 	***(03)***	***CO5***	***04***

5C.	List any two MATLAB functions and toolboxes relevant in your field of engineering.	***(04)***	***CO5***	***02***

Reg. No.									

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

IV SEMESTER B.TECH (ELECTRICAL & ELECTRONICS ENGINEERING)

MAKEUP EXAMINATIONS, JULY 2023

MATLAB for Engineers [ELE 4303]

REVISED CREDIT SYSTEM

Time: 3 Hours **Date: 11 JULY 2023** **Max. Marks: 50**

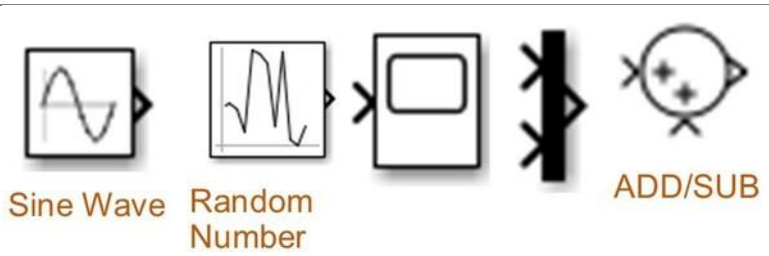
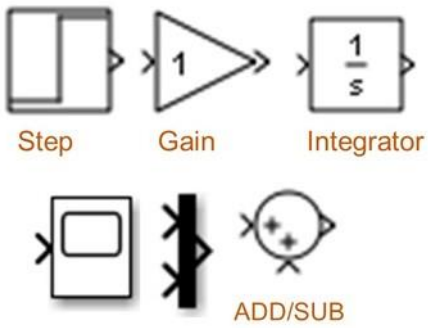
Instructions to Candidates:

- Answer **ALL** the questions.
- Missing data may be suitably assumed.
- Write MATLAB code wherever required.

Q.NO	Questions	Marks	CO	BTL
1A.	When MATLAB application is started on the **desktop**, the desktop appears in its default layout. What are the **different panels** available in **MATLAB Desktop environment**? Mention their uses.	(03)	01	02
1B.	The circumference (perimeter) of an ellipse can be approximated by calculating an intermediate parameter h: $h = \frac{(a-b)^2}{(a+b)^2}$ The approximate circumference of an ellipse can be found from a, b, and h as: $C \simeq \pi(a+b)(1+\frac{3h}{10+\sqrt{4-3h}})$ Create a script file that defines a and b, calculates h, and then calculates the final circumference. Assume that a=5 and b=10.	(03)	01	03
1C.	In MATLAB, **eye(n)** returns an n-by-n **identity matrix** with ones on the main diagonal and zeros elsewhere. Given a matrix **A = eye(4)**, give the output of the following MATLAB operations on matrix A. a) A (2, :) b) A (1:2, 2 : end) c) A (1:2,3:4) d) A(11)	(04)	CO2	04
2A.	Write a MATLAB program to evaluate a function f(x,y) for any two user-specified values x and y. The function f(x,y) is defined as follows:	(03)	CO1	03

	$f(x,y) = \begin{cases} x+y & for\ x \geq 0\ and\ y \geq 0 \\ x+y^2 & for\ x \geq 0\ and\ y < 0 \\ x^2+y & for\ x < 0\ and\ y \geq 0 \\ x^2+y^2 & for\ x < 0\ and\ y < 0 \end{cases}$			
2B.	What is the difference between an array, matrix, scalar and a vector in MATLAB? Give examples using MATLAB statements.	***(03)***	***CO2***	***04***
2C.	Write a MATLAB code to do the following: Create a **identity matrix A** of size 5 x 5. Write the contents of the **matrix A** into a file named "**ONE.txt**". Read the file "**ONE.txt**" and write the ODD columns from the file into "**TWO.txt**". Delete "**ONE.txt**" from the current directory. Some MATLAB functions for reference: • **A = readmatrix(FILENAME)** creates a homogeneous array by reading from a file. • **writematrix(A, FILENAME)** writes the homogenous array A to the file FILENAME as column-oriented data. delete file_name **deletes the named file from disk.**	***(04)***	***CO2***	***04***
3A.	Write a MATLAB code to check if a given **word or a whole number** is a palindrome or not. Use **WHILE loop** in your code. Find the **time complexity of the code**. Palindrome is a word, phrase, or sequence that reads the same backwards as forwards, e.g. madam, 10101 etc.	***(03)***	***CO3***	***03***
3B.	Let y=sin(x), where 'x' is a row vector of 100 evenly spaced points between 0 and 10pi. Write a MATLAB code to PLOT half wave rectified 'y' (i.e. show only positive cycles of the wave form). Illustrate the use of **FOR** loop in the code.	***(03)***	***CO3***	***03***
3C.	Write a MATLAB code to construct a pie chart to visually display the favourite fruits of the students in a class based on the given data: Mango - 45; Orange - 30; Plum - 15; Pineapple - 30; Apple – 30. Use labels in the chart. MATLAB function for reference: **pie(X)** draws a pie chart using the data in X. Each slice of the pie chart represents an element in X. **pie(X, labels)** specifies options for labelling the pie slices. In this case, X must be numeric. **bar(y)** creates a bar graph with one bar for each element in y.	***(04)***	***CO3***	***03***

4A.	Write a MATLAB code to evaluate the polynomial p(x) = 5x+10 for 30 evenly spaced points between -2 and 2. MATLAB function for reference: y = **polyval**(p,x) evaluates the polynomial p at each point in x. The argument p is a vector of length n+1 whose elements are the coefficients (in descending powers) of an nth-degree polynomial.	***(03)***	***CO4***	***04***
4B.	Solve the following puzzle using MATLAB symbolic computation code. **"John bought some pens and pencils at a bookstore. If 5 pens and 3 pencils cost Rs 8.40 while 5 pencils and 3 pens cost Rs 6.00 instead, find the cost of a pen and pencil."** Show necessary equations for solving the puzzle.	***(03)***	***CO4***	***03***
4C.	The following data relates to indirect labour expenses and the level of output. **Months** | Jan | Feb | Mar | Apr | May | June **Units of output** | 200 | 300 | 400 | 640 | 540 | 580 **Indirect labour expenses (Rs)** | 2500 | 2800 | 3100 | 3820 | 3220 | 3640 Write a MATLAB code to estimate the expenses at a level of output of 350 and 500 units. Use 1-D data interpolation in MATLAB. Description of the MATLAB function for 1-D data interpolation (table lookup) is as follows: vq = **interp1**(x,v,xq) returns interpolated values of a 1-D function at specific query points using linear interpolation. Vector x contains the sample points, and v contains the corresponding values, v(x). Vector xq contains the coordinates of the query points.	***(04)***	***CO4***	***03***
5A.	Create a SIMULINK model to add noise to a SINE wave and display the output on a scope. Use Simulink's built in random number generator to simulate the noise. Make use of the following SIMULINK blocks to create the model.	***(03)***	***CO5***	***03***

				
5B.	Create a SIMULINK model to obtain the step response of the equation: dy/dt = 5f(t) - 7y; Make use of the following SIMULINK blocks to create the model. 	***(03)***	***CO5***	***04***
5C.	List any two features and any two applications of SIMULINK in your field of engineering.	***(04)***	***CO5***	***02***

ELE 4303 MATLAB for Engineers (OE 1)

MID TERM EXAMINATION

Date : 29-Oct-2024 Marks : 30

Questions 1 to 10 carries ½ M each.

1. The MATLAB command that lists all the MATLAB script files in a given folder is

A) list = ls('*.m') B) list = ls('*MATLAB*') C) list = ls('*.doc') D) list = ls('*.txt')

2. In MATLAB, _____ gives the 'Number of function input arguments'.

A) nargin B) nargout C) varargin D) varargout

3. The command to terminate/quit/exit MATLAB

A) Either Quit or Exit B) Quit C) Exit D) Ctrl + C

4. The output of the command a=1+1:5 in MATLAB

A) 2 3 4 5 6 B) 2 3 4 5 C) 1 2 3 4 5 D) 5 4 3 2 1

5) The command that returns the total number of elements in a matrix

A) numel B) length C) size D) count

6) In symbolic computation, for the function f=(x+2)^3, which of the following command gives the output as : **x^3 + 6*x^2 + 12*x + 8**

A) simplify(f) B) factor(f) C) expand(f) D) solve(f)

7) What is the output of the following symbolic code : >> **syms f(x,y); f(x,y) = x^2*y; f(3,2)**

A) 6 B) 9 C) 18 D) 5

8) What is the outcome of the command **rem(40, 12)**?

A) 4 B) -4 C) 3.33 D) 3

9) _____ divides the current figure into an **m-by-n** grid and creates axes in the position specified by **p**.

A) subplot(p,m,n) B) subplot(m,n,p) C) plot(m,n,p) D) plot(p,m,n)

10) Which command is used to clear a command window?

A) clear B) close all C) clc D) clear all

ELE 4303 MATLAB for Engineers (OE 1)

MID TERM EXAMINATION

11) List any TWO toolboxes of MATLAB. (2M)

12) What would be the output of the following code (in editor window)? (2M)

>> A = [0 1; 1 0] ; B=2 ; C = A + B

13) List any TWO tools/components of Development Environment (GUI). (2M)

14) Variable T ranges from 0 to 2π, write a MATLAB code to plot the sine and cosine values of T in the same plot. (3M)

15) Differentiate between Iterative and recursive algorithms. Which is Better? (3M)

16) Example of solving linear equations in MATLAB is given below:

```
syms x y z
eqn1 = 2*x + y + z == 2;
eqn2 = -x + y - z == 3;
eqn3 = x + 2*y + 3*z == -10;
[x y z]=solve(eqn1, eqn2, eqn3);
```

In the similar way write MATLAB code to solve :

The ratio of two number is 2/3. If 2 is subtracted from the first and 8 from the second, the ratio becomes the reciprocal of the original ratio. Find the numbers. (3M)

17) What are the rules to be followed while naming VARIABLES, FILES, and FUNCTIONs in MATLAB (3M)

18) Write a MATALB code to evaluate the polynomial $g(x) = - x^5 + 3x^3 - 2.5x^2 - 2.5$ over the interval [0,1] (3M)

19) List any FOUR applications of MATLAB in your field of engineering. (4M)