

# Introduction to MATLAB

Mughees Asif

3<sup>rd</sup> Year Aerospace Engineering

QMUL MathWorks Student Ambassador



# MATLAB

- Stands for **MAT**rix **LAB**oratory
- Interpreted language
- Scientific programming environment
- Very good tool for the manipulation of matrices
- Great visualisation capabilities
- Loads of built-in functions
- Easy to learn and simple to use

# Desktop

The image shows the MATLAB Desktop interface with three callout boxes highlighting key components:

- Workspace / Current Directory:** A table listing variables in the workspace.
- Command Window:** A text area for entering and executing MATLAB commands.
- Command History:** A list of previously executed commands.

Name	Size	Bytes	Class
C1_4096	1x4	2359536	cell array
C2_4096	1x4	6029552	cell array
Dist1_4096	4096x8x2	3580696	cell array
Dist2_4096	4096x22x2	10731648	cell array
Dist4_4096	4096x70x2	35755056	cell array
Ind4096	4096x7	229376	double array
M4096	1x4096	11264400	cell array
Mout4096	2x4096	22528800	cell array
Neigh4096	4096x8	262144	double array
NeighN2_4096	2x4096	1080464	cell array
NeighN4_4096	4x4096	2951648	cell array
S1_4096	1x4	2359536	cell array
S2_4096	1x4	6029552	cell array
W4096	4096x2	65536	double array
ans	1x2	16	double array
tri4096	8106x3	194544	double array

```
>> size(Ind4096,2)
ans =
    7
>> size(Ind4096)
ans =
    4096         7
>> whos
Name      Size      Bytes  Class
C1_4096    1x4        2359536  cell array
C2_4096    1x4        6029552  cell array
Dist1_4096 4096x8x2    3580696  cell array
Dist2_4096 4096x22x2   10731648  cell array
Dist4_4096 4096x70x2   35755056  cell array
Ind4096    4096x7      229376   double array
M4096      1x4096     11264400  cell array
Mout4096   2x4096     22528800  cell array
Neigh4096  4096x8      262144   double array
NeighN2_4096 2x4096     1080464  cell array
NeighN4_4096 4x4096     2951648  cell array
S1_4096    1x4        2359536  cell array
S2_4096    1x4        6029552  cell array
W4096      4096x2      65536   double array
ans        1x2         16   double array
tri4096    8106x3      194544   double array

Grand total is 8186522 elements using 105422504 bytes
>>
```

Command History:

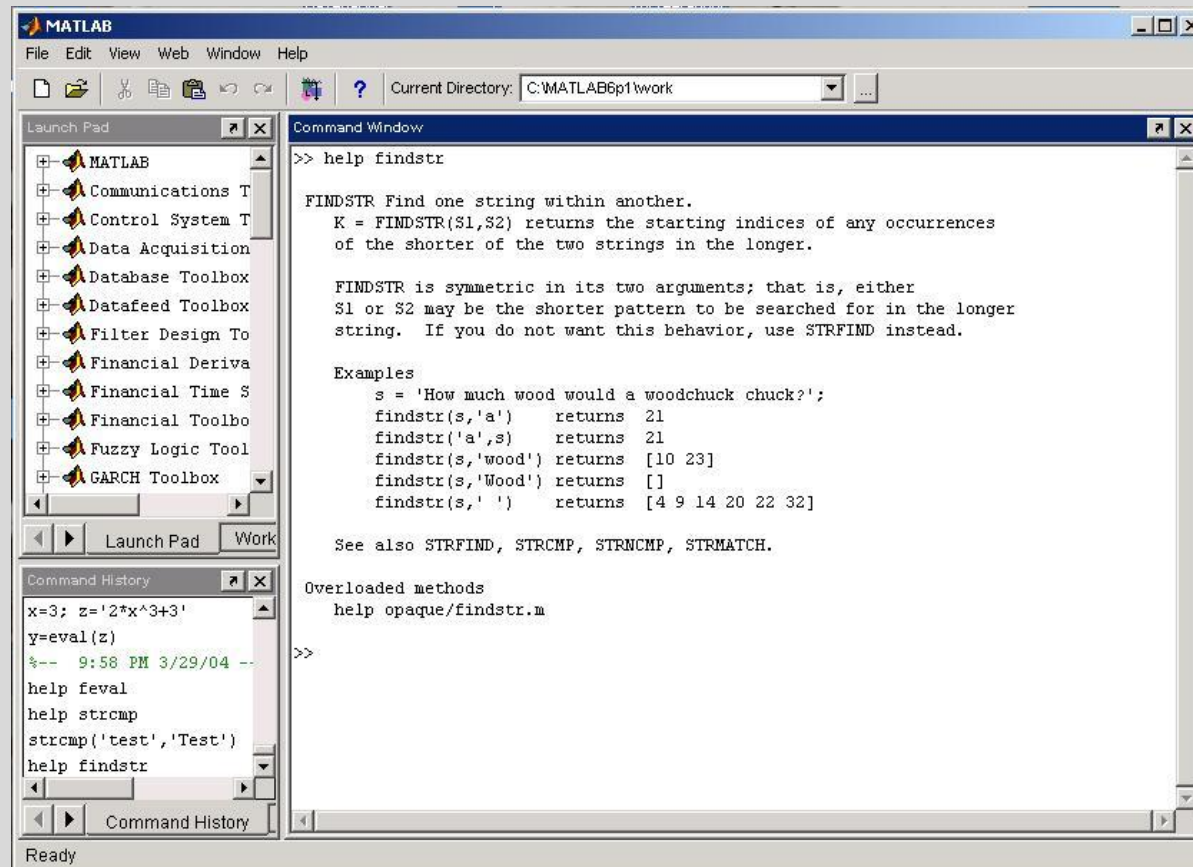
```
W4096( (NeighN2_4096(1,1),1), (NeighN2_4096(1,1),1) )
plot(W4096(NeighN2_4096(1,1),1), W4096(NeighN2_4096(1,1),1))
hold on
plot(W4096(1,1), W4096(1,'r.'))
plot(W4096(1,1), W4096(1,2),'r.')
size(Ind4096)
size(Ind4096,2)
size(Ind4096,1)
size(C1_4096)
size(Ind4096,2)
size(Ind4096)
```



Explore the MATLAB Desktop

# Getting Help and Looking Up Functions

- To get help on a function type “help function\_name”, e.g., “help plot”.
- To find a topic, type “lookfor topic”, e.g., “lookfor matrix”



# Workspace

- **who, whos** – current workspace vars.
- **save** – save workspace vars to \*.mat file.
- **load** – load variables from \*.mat file.
- **clear all** – clear workspace vars.
- **close all** – close all figures
- **clc** – clear screen
- **clf** – clear figure

# MATLAB symbols

- >> prompt
- . . . continue statement on next line
- , separate statements and data
- % start comment which ends at end of line
- ;
  - (1) suppress output
  - (2) used as a row separator in a matrix

# Matrices

- Don't need to initialise type, or dimensions

```
>>A = [3 2 1; 5 1 0; 2 1 7]
```

```
A =
```

```
    3    2    1
```

```
    5    1    0
```

```
    2    1    7
```

```
>>
```

square brackets to define matrices

semicolon for next row in matrix

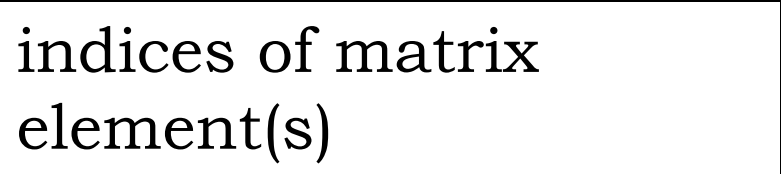
# Manipulating Matrices

- Access elements of a matrix

```
>>A(1,2)
```

```
ans=
```

```
2
```



indices of matrix  
element(s)

```
A =  
 3  2  1  
 5  1  0  
 2  1  7
```

- Remember Matrix\_name(row,column)
- Naming convention Matrix variables start with a capital letter while vectors or scalar variables start with a simple letter



# The : operator

- VERY important operator in MATLAB
- Means '**to**'

```
>> 1:10
```

```
ans =
```

```
1    2    3    4    5    6    7    8    9   10
```

```
>> 1:2:10
```

```
ans =
```

```
1    3    5    7    9
```

# Manipulating Matrices

>> A.'	% transpose	A =
>> B * A	% matrix multiplication	3 2 1
>> B. * A	% element by element multiplication	5 1 0
>> B / A	% matrix division	2 1 7
>> B. / A	% element by element division	B =
>> [B A]	% join matrices (horizontally)	1 3 1
>> [B; A]	% join matrices (vertically)	4 9 5
		2 7 2

# For loops

- $x = 0;$   
    **for**  $i = 1:2:5$                       % start at 1, increment by 2  
         $x = x + i;$                       % end with 5.  
    **end**

This computes  $x = 0 + 1 + 3 + 5 = 9$

# While loops

- $x = 7;$   
**while** ( $x \geq 0$ )  
     $x = x - 2;$   
**end;**

This computes  $x = 7 - 2 - 2 - 2 - 2 = -1$

# If statements

- **if (x == 3)**  
    disp('The value of x is 3.');
- elseif (x == 5)**  
    disp('The value of x is 5.');
- else**  
    disp('The value of x is not 3 or 5.');
- end;**

# Switch statement

- **switch dice\_face**  
  case {1}  
    disp('Rolled a 1');  
  case {2}  
    disp('Rolled a 2');  
    ....  
  case {5}  
    disp('Rolled a 5');  
  otherwise  
    disp('Rolled a 6');  
  **end**

# Break statements

- **break** – terminates execution of for and while loops. For nested loops, it exits the innermost loop only.

# Vectorization

- Because MATLAB is an interpreted language, i.e., it is not compiled before execution, loops run slowly.
- *Vectorized code runs faster in MATLAB.*
- Example: `x = [1 2 3];`

**for i = 1:3**

`x(i) = x(i) + 5;`

**end;**

VS.

Vectorized:

**`x = x + 5;`**



# Example

- This code computes the *sine* of 1,001 values ranging from 0 to 10:

```
i = 0;  
for t = 0:.01:10  
    i = i + 1;  
    y(i) = sin(t);  
end
```

- This is a vectorized version of the same code:

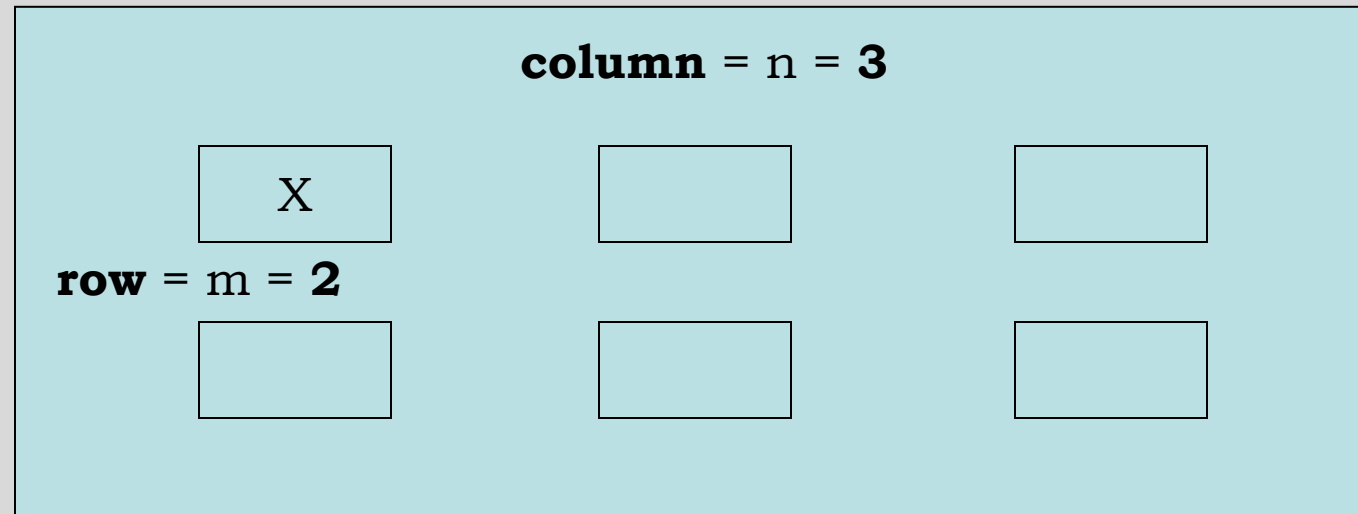
```
t = 0:.01:10;  
y = sin(t);
```

# Graphics

- **plot(x, y);** % plots y vs. x.
- **plot(x, y, 'k-');** % plots a black line of y vs. x.
- **hold on;** % put several plots in the same  
% figure window.
- **figure;** % open new figure window.

# Graphics

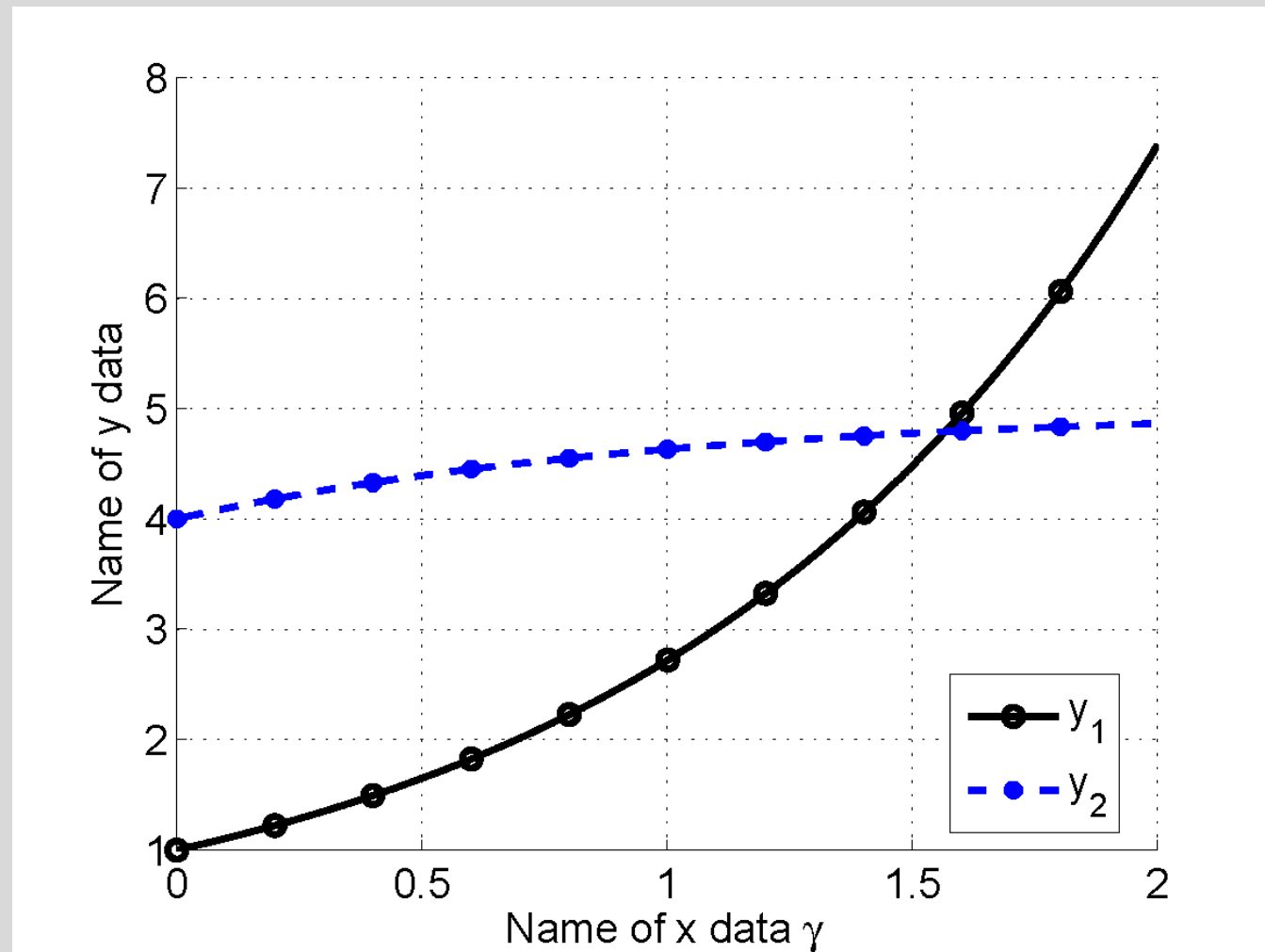
- **subplot(m, n, 1)**    % Makes an **m x n** array  
% for plots. Will place plot in 1<sup>st</sup>  
% position.



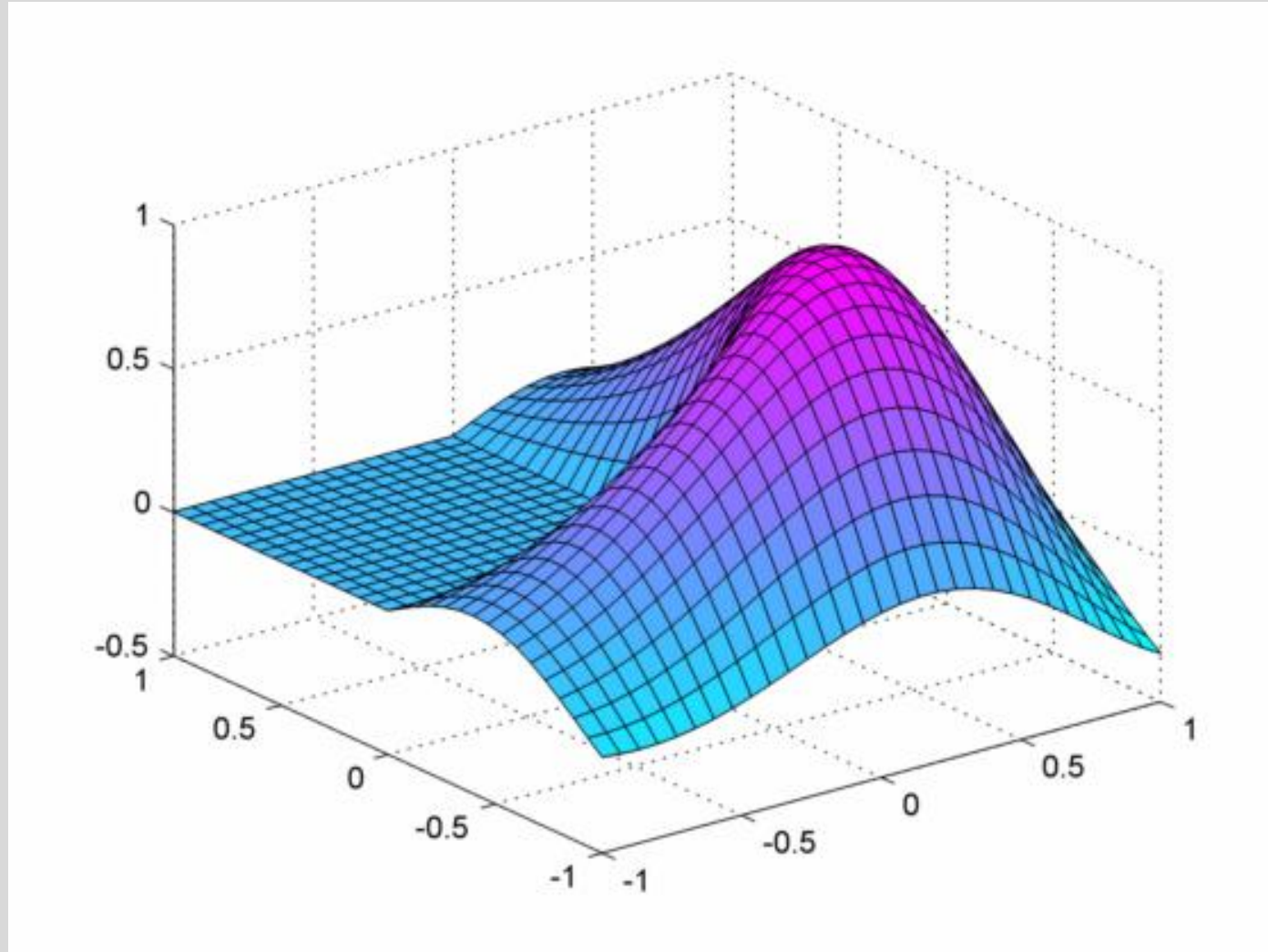
# Graphics

- **plot3**( $x$ ,  $y$ ,  $z$ ) – **plot** 2D function.
- **mesh**( $x$ ,  $y$ ,  $z$ ) – **surface** plot.
- **contour**( $z$ ) – **contour** plot of  $z$ .
- **axis**( $[x_{min} \ x_{max} \ y_{min} \ y_{max}]$ ) – change **axes**
- **title**('My title') – add **title** to figure;
- **xlabel**('x label'), **ylabel**('y label') – **label** axes.
- **legend** – add **key** to figure.

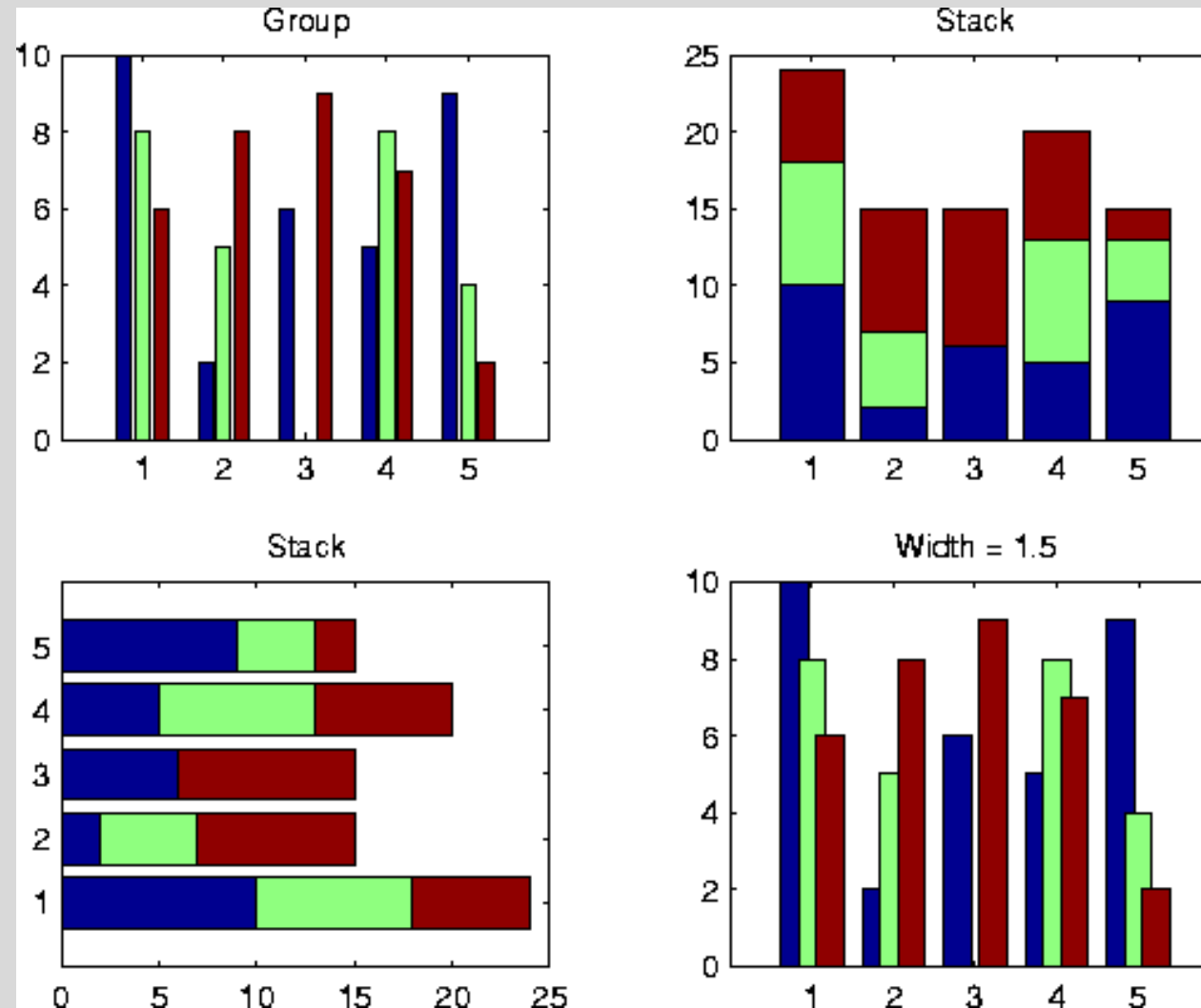
# Examples of Plots – $x$ vs $y$



# Examples of Plots – 3D surface



# Examples of Plots – *Bar charts*



# Scripts and Functions

- Two kinds of M-files:
  - **Scripts**, which do not accept input arguments or return output arguments. They operate on data in the workspace. *FIXED*
  - **Functions**, which can accept input arguments and return output arguments. Internal variables are local to the function. *VARIABLE*



# Advantages

- May behave as a calculator or as a programming language
- Has powerful graphics generation/visualisation of data
- Relatively easy to learn
- Interpreted (not compiled), errors are easy to fix
- Optimized to be relatively fast when performing matrix operations

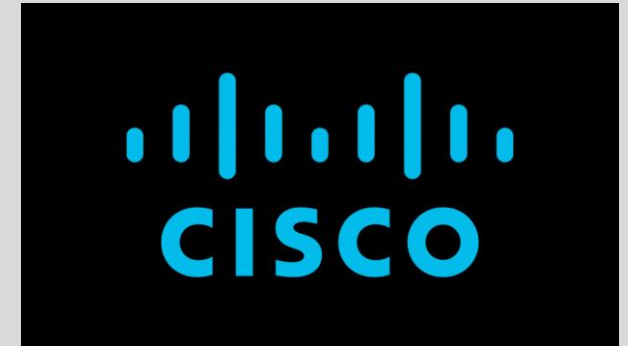
# Dis-advantages

- Not a general-purpose programming language such as C, C++, or FORTRAN
- Designed for scientific computing, and is not well suitable for other applications
- Interpreted language, slower than a compiled language such as C++
- MATLAB commands are specific for MATLAB usage. Most of them do not have a direct equivalent with other programming language commands



Let's look  
at some  
**code** now!

# Cisco



- **Campus Ambassador**
- Do join the **LinkedIn page** for graduate/internship opportunities: <https://www.linkedin.com/groups/12470630/>
- Join the **email newsletter** to stay up-to-date: <https://forms.gle/QZamk91qoJm3pt4s6>
- To **access** the job opportunities, use the link: <https://shorturl.at/gzCFQ>



# Developer Student Clubs

Queen Mary, University of London

- Work in a team of 17 students, in collaboration with Google
- Leveraging Google products to solve local community challenges
- Do join the **LinkedIn page** to get involved:  
<https://www.linkedin.com/groups/12467711/>
- Check out the team and for future updates, do **join the chapter**:  
<https://dsc.community.dev/queen-mary-university-of-london/>

# KAHOOT! Challenge

Will post all  
prizes by next  
day

All sizes  
available

<b>First</b>	1 x <i>t-shirt</i> 1 x <i>baseball cap</i> 1 x <i>sunglasses</i> 10 x <i>pens</i>
<b>Second</b>	1 x <i>drawstring bag</i> 1 x <i>baseball cap</i> 10 x <i>pens</i> 20 x <i>stickers</i>
<b>Third</b>	10 x <i>pens</i> 10 x <i>stickers</i>