

EE 1322: Intro to Electrical & Computer Engineering

Module 4: Intro to Programming in Matlab

Dr. Ahmad F. Taha

ahmad.taha@utsa.edu || <http://engineering.utsa.edu/ataha>



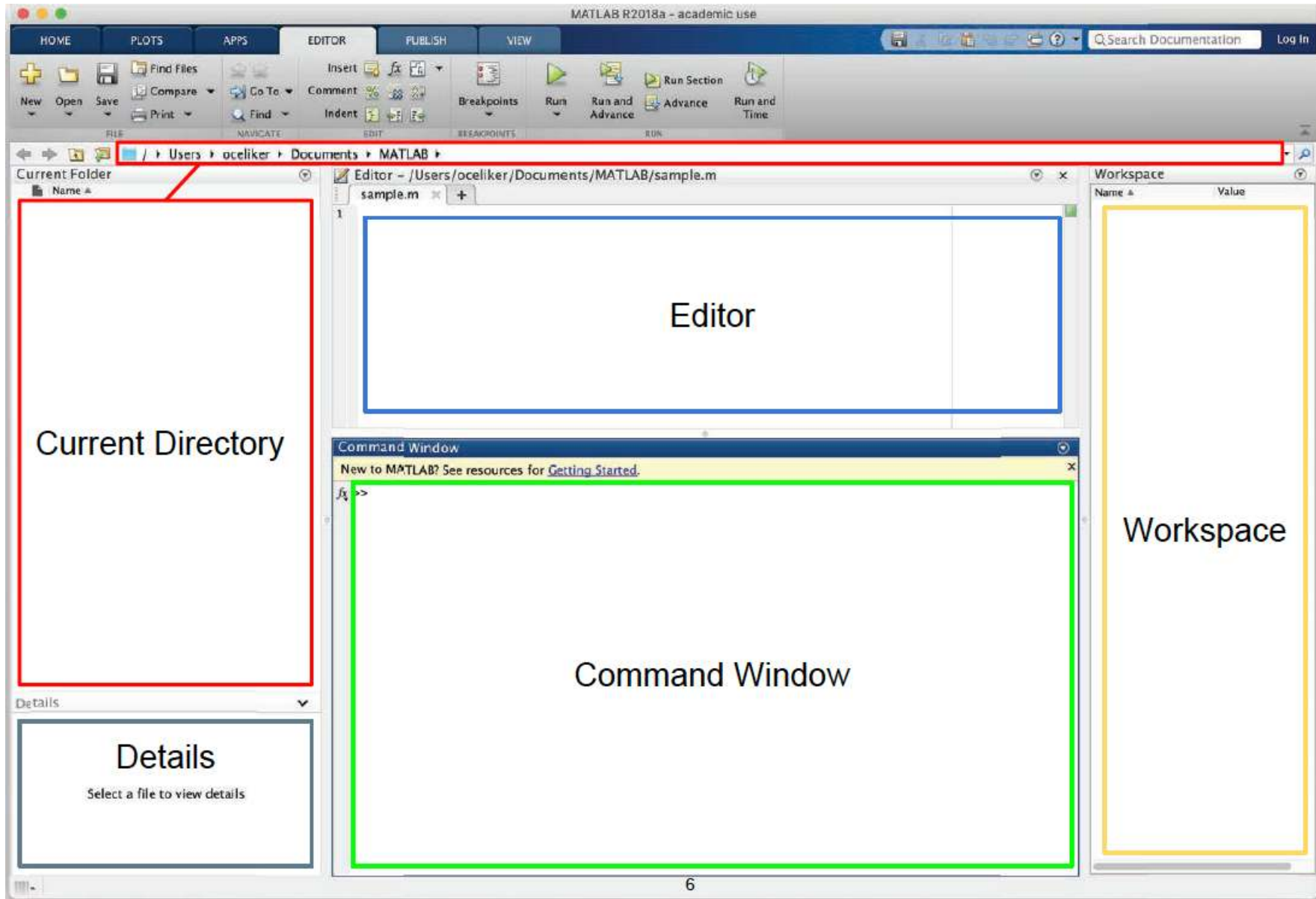
1. The very basics
2. Functions and programming
3. Plotting and graphs
4. Numerical computing

Part 1:

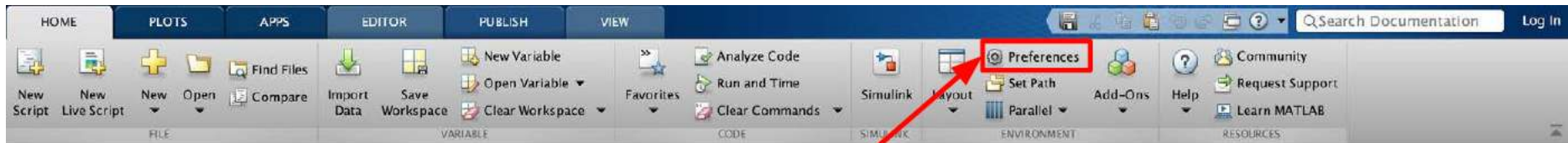
The Basics

- Stands for MATrix LABoratory
- 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

Matlab desktop

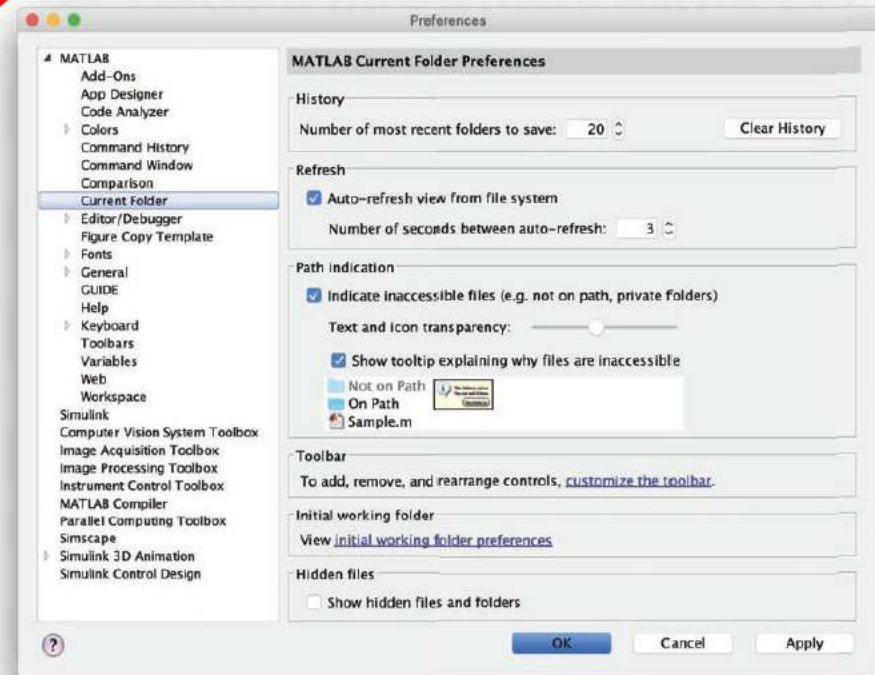


Making it your own



Customization

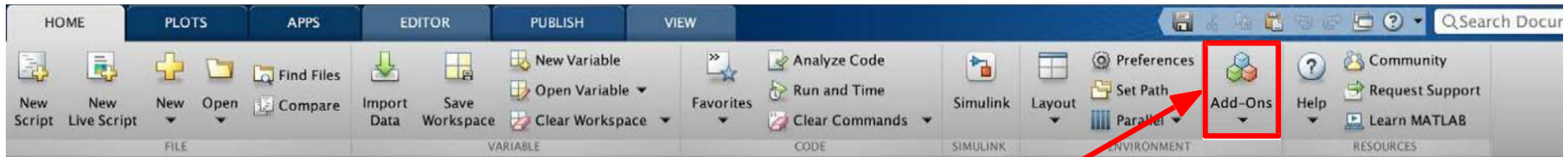
- In the top ribbon, navigate to:
Home -> Environment -> Preferences
- Allows you to customize your
MATLAB experience (colors, fonts,
etc.)



7

MATLAB R2018a. Courtesy of The MathWorks, Inc. Used with permission. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Installing toolboxes

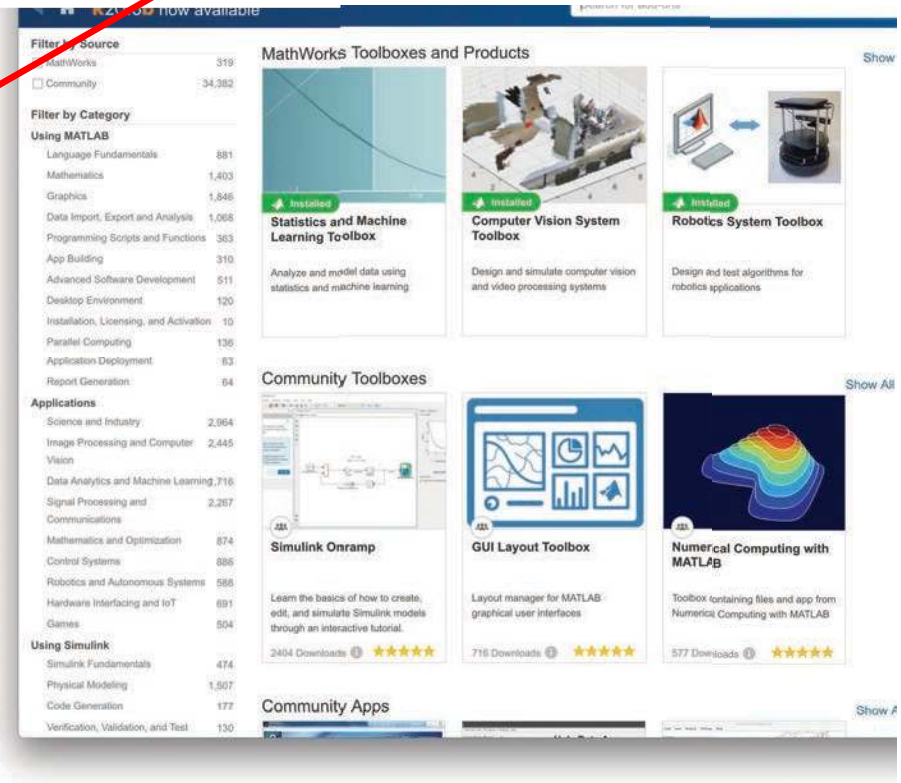


- In the top ribbon, navigate to:
Home -> Environment -> Add-Ons

- Allows you to install toolboxes included with your license

Recommended toolboxes:

- - Curve Fitting Toolbox
 - Computer Vision System Toolbox
 - Image Processing Toolbox
 - Optimization Toolbox
 - Signal Processing Toolbox
 - and anything related to your field!

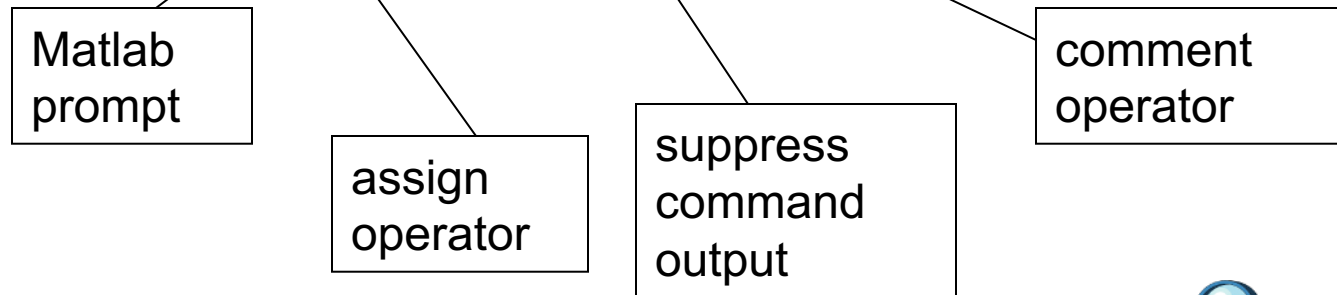


Variables

- Don't have to declare type
- Don't even have to initialize
- Just assign in command window

>>

>> a=12; % variable a is assigned 12



Try the same line without the semicolon and comments

Variables (continued ...)

- View variable contents by simply typing the variable name at the command prompt

```
>> a
```

```
a =
```

```
    12
```

```
>>
```

```
>> a*2
```

```
a =
```

```
    24
```

```
>>
```

Workspace

- The workspace is Matlab's memory
- Can manipulate variables stored in the workspace

```
>> b=10;
```

```
>> c=a+b
```

```
c =
```

```
    22
```

```
>>
```

Workspace (continued ...)

- Display contents of workspace

```
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array
c	1x1	8	double array

```
Grand total is 3 elements using 24 bytes
```

```
>>
```

- Delete variable(s) from workspace

```
>> clear a b; % delete a and b from workspace
```

```
>> whos
```

```
>> clear all; % delete all variables from workspace
```

```
>> whos
```

Matlab help commands

- help

>> help whos % displays documentation for the function whos

>> lookfor convert % displays functions with convert in the first help line

- Start Matlab help documentation

>> helpdesk

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

```
>>
```

- 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(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
```



Try the following

```
>> x=0:pi/12:2*pi;
```

```
>> y=sin(x)
```

The ':' operator and matrices

```
>>A(3,2:3)
```

```
ans =
```

```
1    7
```

```
>>A(:,2)
```

```
ans =
```

```
2
```

```
1
```

```
1
```

A =

3	2	1
5	1	0
2	1	7



What'll happen if you type A(:,,:) ?

Manipulating Matrices

A =			B =		
3	2	1	1	3	1
5	1	0	4	9	5
2	1	7	2	7	2

```
>> A'           % transpose
>> B*A          % matrix multiplication
>> B.*A         % element by element multiplication
>> B/A          % matrix division
>> B./A         % element by element division
>> [B A]        % Join matrices (horizontally)
>> [B; A]       % Join matrices (vertically)
```



Enter matrix
B into the
Matlab
workspace



Create matrices A and B and try out the the matrix operators in this slide

Rules on Variables and File Names

- Variable, function, file names

- is 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: `% = + - ~ ; : ! ' [ ] ( ) , @ # $ & ^`
- up to 63 characters
- A function performs a pre-defined task based on input to yield certain outcome.

- 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.,

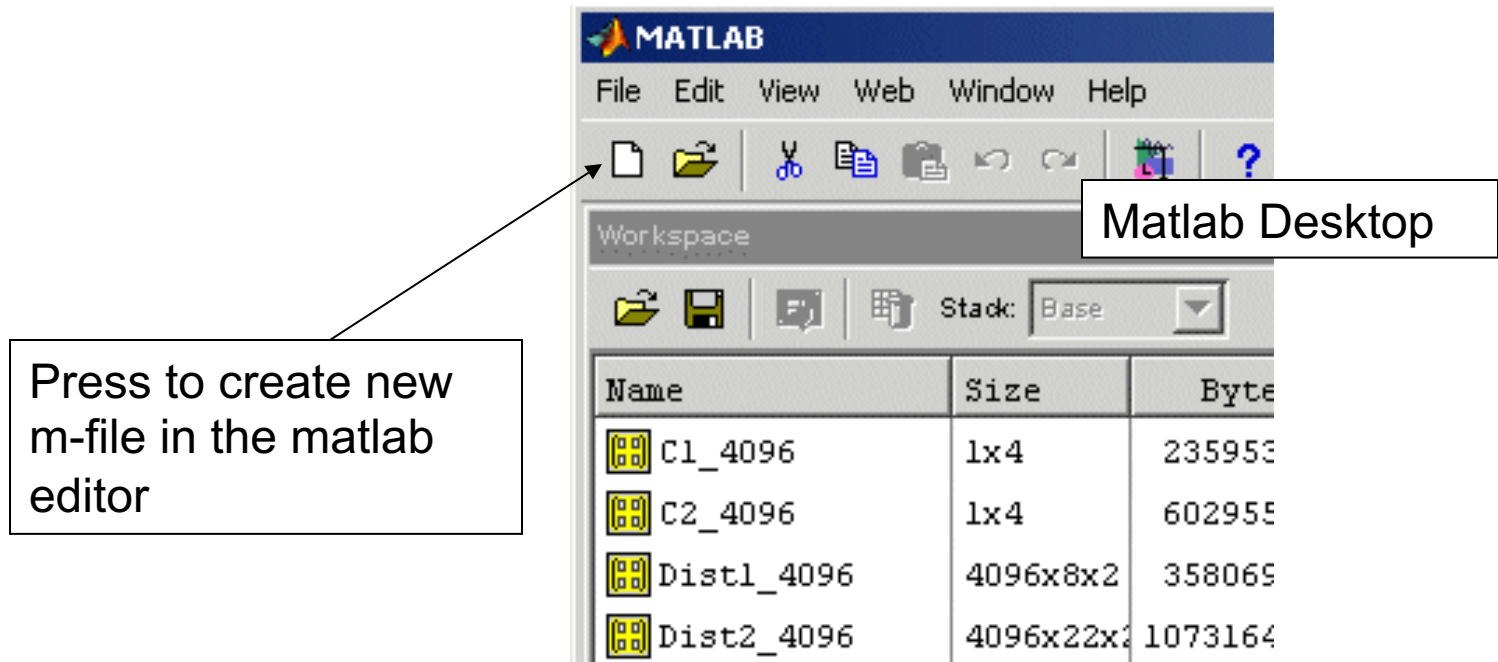
>> myfile

Part 2:

Scripts, Functions, and Programming

Scripts

- Matlab editor
- Use scripts to execute a series of Matlab commands



Scripts (continued)

- Scripts will manipulate and store variables and matrices in the Matlab Workspace (memory)

- They can be called from the Matlab command line by typing the (case sensitive!) filename of the script file

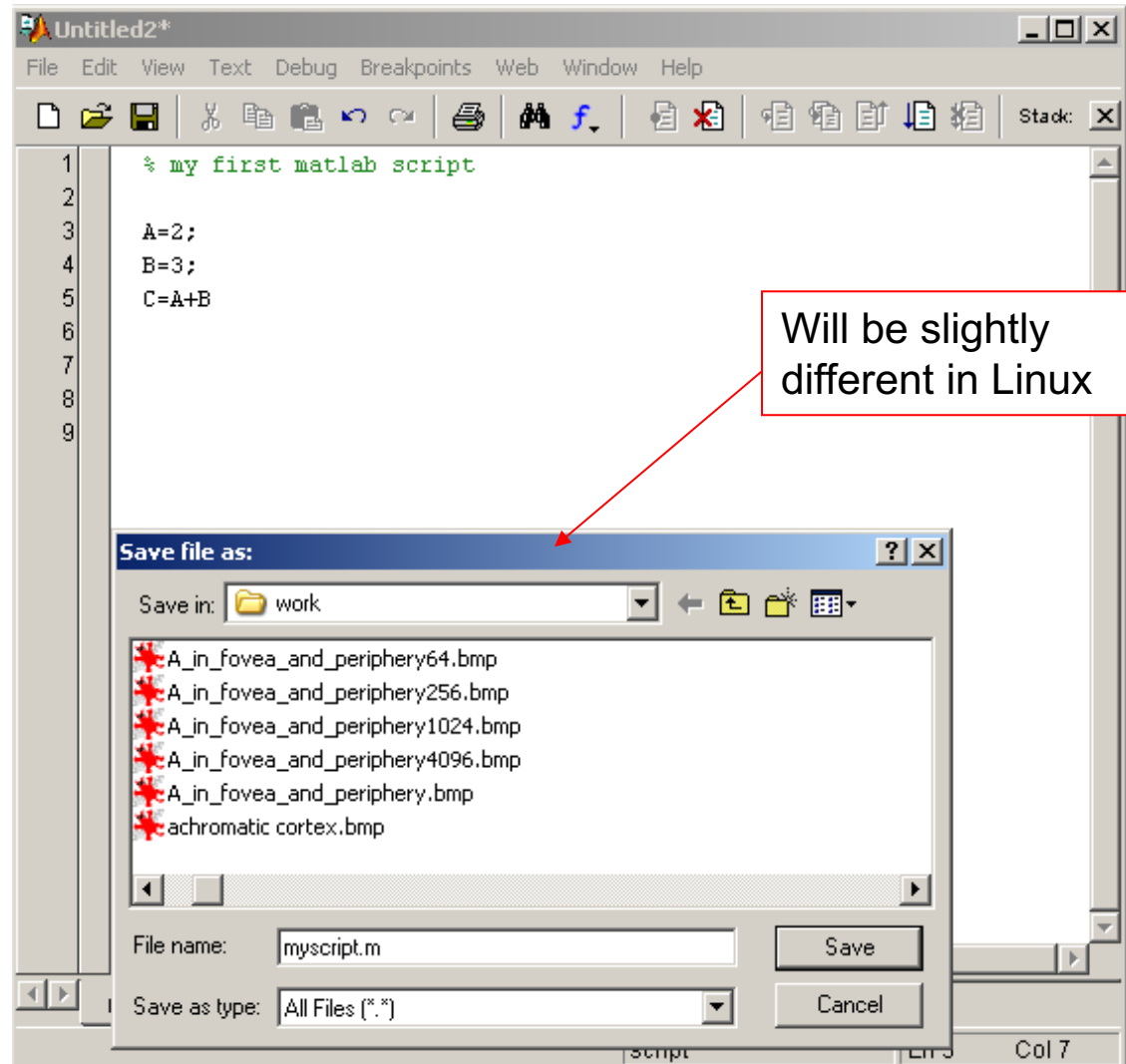
>> myscript

- Scripts can be opened in the editor by the following

>> open myscript



Highlight a few lines of your script by left- clicking and dragging the mouse over the lines. Right-click the highlighted lines and select Evaluate Selection.



Functions

- Programming in Matlab
- Users can write functions which can be called from the command line
- Functions can accept input variable(s)/matrice(s) and will output variable(s)/matrice(s)
- Functions will **not** manipulate variable(s)/matrice(s) in the Matlab Workspace
- In Matlab functions closely resemble scripts and can be written in the Matlab editor
- Matlab functions have the **function** keyword
- Remember that the filename of a function will be its calling function name
- Don't overload any built-in functions by using the same filename for your functions or scripts!
- Functions can be opened for editing using the **open** command
- Many built-in Matlab functions can also be viewed using this command

Function Basics

R2020a

Create functions, including anonymous, local, and nested functions

Functions contain sequential commands and can accept inputs and return outputs. Named functions must be defined within a program file and not at the command line. To define a function, use the syntax

```
function [y1,...,yN] = myfun(x1,...,xM)
```

where $y_1, \dots, y_N$ are outputs, $x_1, \dots, x_M$ are inputs, and `myfun` is the function name. For more information, see [Create Functions in Files](#).

- Btw, functions are nearly identical to scripts with one difference:
 - Functions must have a function declaration
 - + I like them better :P

Functions (continued)

```
>> l=iterate(5)
```

```
l =
```

```
1    4    9   16   25
```

output

function name

input

function keyword

help lines for function

for statement block

```
function O=iterate(n)
% iterate(n) outputs the square of the
% integers upto integer n

for i=1:n
    O(i)=i*i;
end
```



Access the comments
of your Matlab functions
>> help iterate

Make sure you save changes to
the m-file before you call the
function!

Functions (continued)

```
>> [i j]=sort2(2,4)
```

```
i =
```

```
4
```

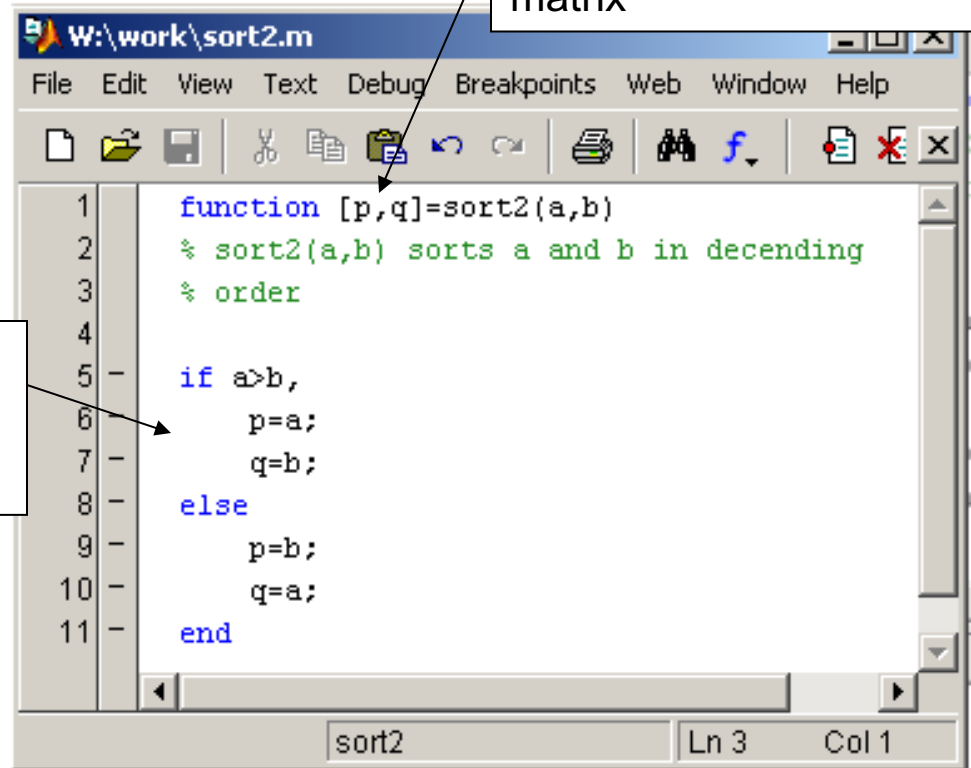
```
j =
```

```
2
```

```
>>
```

if
statement
block

Functions can have many
outputs contained in a
matrix



```
function [p,q]=sort2(a,b)
% sort2(a,b) sorts a and b in decending
% order

if a>b,
    p=a;
    q=b;
else
    p=b;
    q=a;
end
```



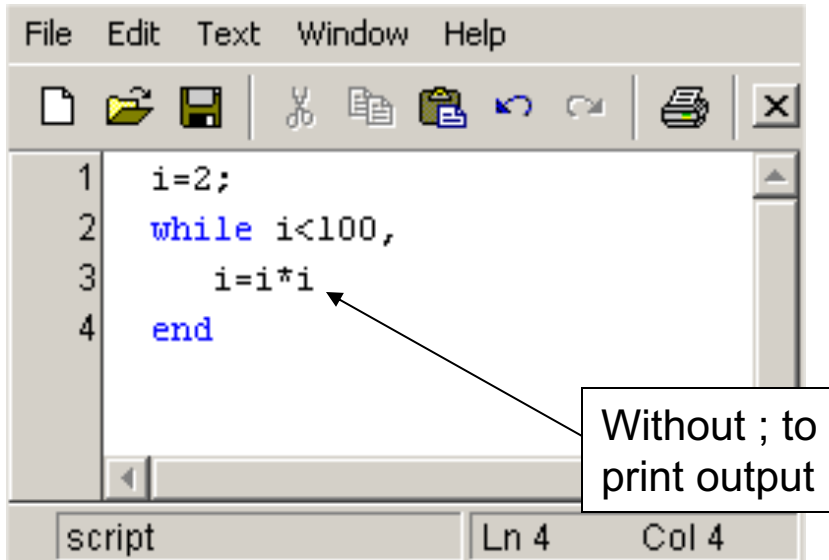
Remember to use the
Matlab help command
for syntax
>> help if

Some Frequently Used Functions

```
>> magic(n)    % creates a special  $n \times n$  matrix; handy for testing
>> zeros(n,m)  % creates  $n \times m$  matrix of zeroes (0)
>> ones(n,m)   % creates  $n \times m$  matrix of ones (1)
>> rand(n,m)   % creates  $n \times m$  matrix of random numbers
>> repmat(a,n,m) % replicates  $a$  by  $n$  rows and  $m$  columns
>> diag(M)      % extracts the diagonals of a matrix M
>> help elmat   % list all elementary matrix operations ( or elfun)
>> abs(x);      % absolute value of x
>> exp(x);      %  $e$  to the  $x$ -th power
>> fix(x);      % rounds  $x$  to integer towards 0
>> log10(x);    % common logarithm of  $x$  to the base 10
>> rem(x,y);    % remainder of  $x/y$ 
>> mod(x, y);   % modulus after division – unsigned rem
>> sqrt(x);     % square root of  $x$ 
>> sin(x);      % sine of  $x$ ;  $x$  in radians
>> acoth(x)     % inversion hyperbolic cotangent of  $x$ 
```

More flow control

While statement block



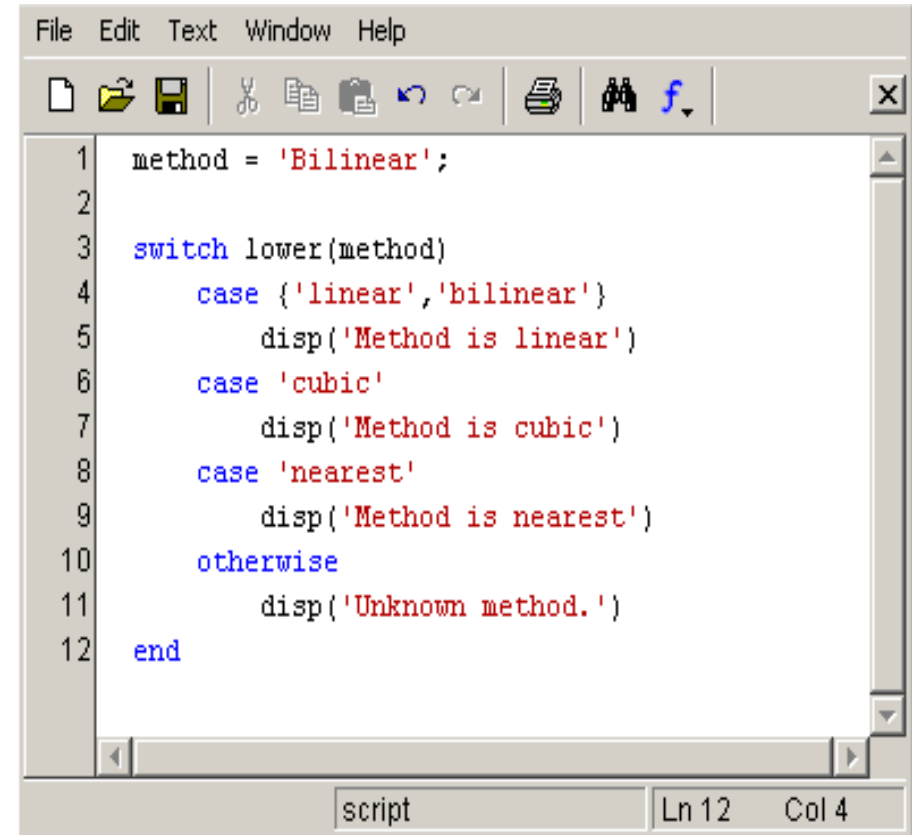
```
File Edit Text Window Help
[Icons]
1 i=2;
2 while i<100,
3     i=i*i
4 end
```

Without ; to print output

script Ln 4 Col 4

```
i =
    4
i =
   16
i =
  256
```

Switch statement block



```
File Edit Text Window Help
[Icons]
1 method = 'Bilinear';
2
3 switch lower(method)
4     case {'linear','bilinear'}
5         disp('Method is linear')
6     case 'cubic'
7         disp('Method is cubic')
8     case 'nearest'
9         disp('Method is nearest')
10    otherwise
11        disp('Unknown method.')
12 end
```

script Ln 12 Col 4

```
Method is linear
>>
```

Debugging

- Set breakpoints to stop the execution of code

```
>> [i j]=sort2(2,4)
```

```
K>>
```

```
K>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array

Grand total is 2 elements using 16 bytes

```
K>> a
```

```
a =
```

```
2
```

```
K>> return
```

```
i =
```

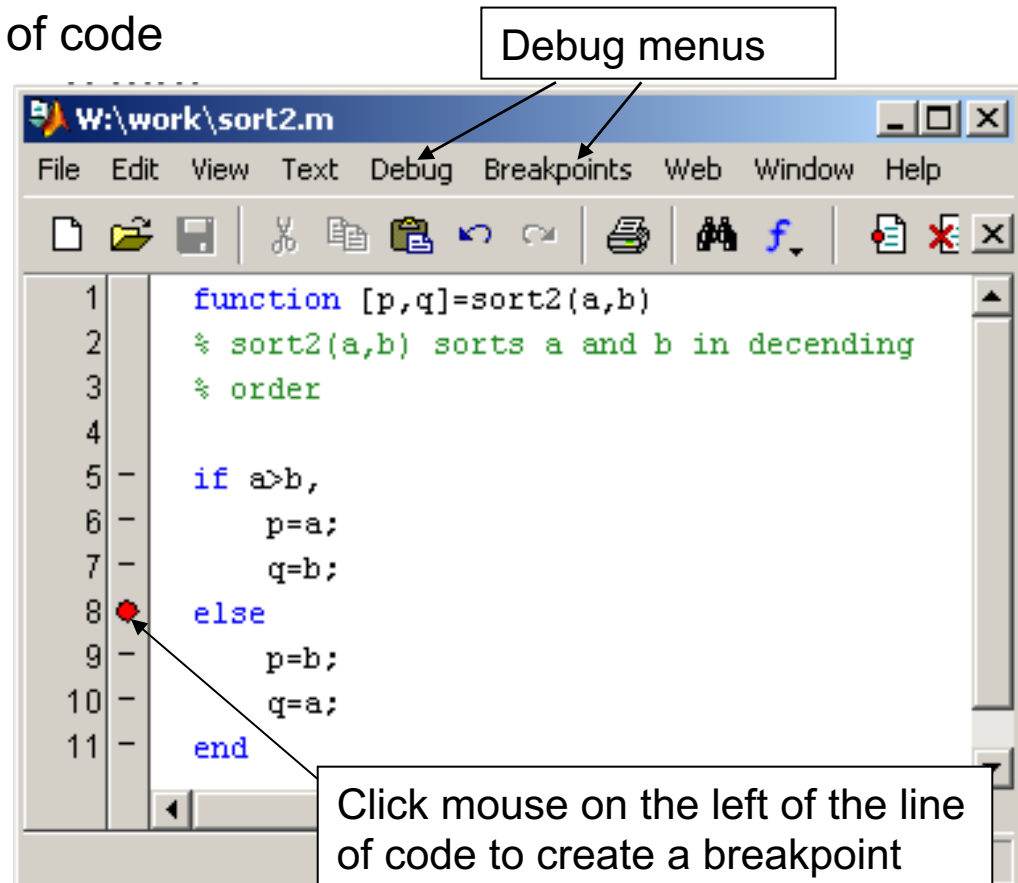
```
4
```

```
j =
```

```
2
```

local function workspace

exit debug mode



Visualisation - plotting data and functions

```
>> figure % create new figure
```

```
>> t=0:pi/12:8*pi;
```

```
>> y=cos(t);
```

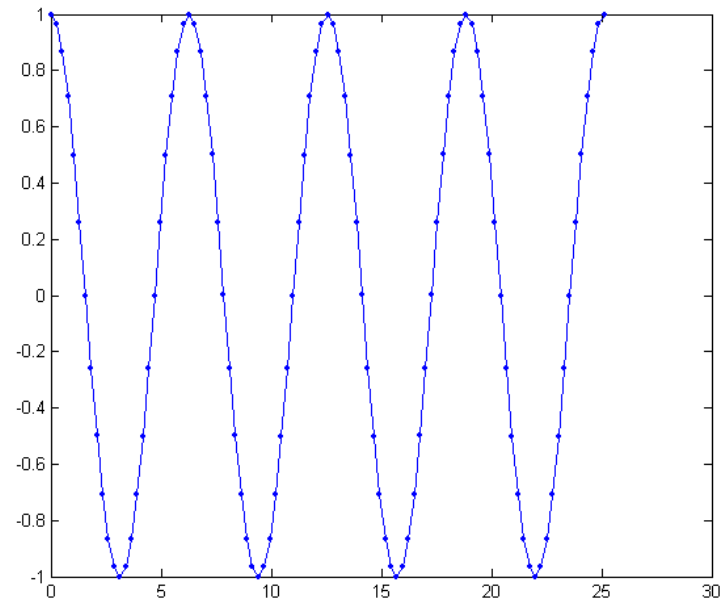
```
>> plot(t,y,'b.-') ← Plot style
```

Investigate the function

```
>> y=A*cos(w*t+phi);
```



for different values of ϕ (eg: 0, $\pi/4$, $\pi/3$, $\pi/2$), w (eg: 1, 2, 3, 4) and A (eg: 1, 0.5, 2). Use the **hold on** Matlab command to display your plots in the same figure. Remember to type **hold off** to go back to normal plotting mode. Try using different plot styles (**help plot**)



A = amplitude

ϕ = phase

w = angular frequency = $2\pi \times \text{frequency}$

So much more in the lab



- The lab sessions will have more hands on experience with Matlab
- You will write your first lines of code and learn a lot

EE 1322: Intro to Electrical & Computer Engineering

Module 4: Intro to Programming in Matlab

Dr. Ahmad F. Taha

ahmad.taha@utsa.edu || <http://engineering.utsa.edu/ataha>

