Introduction to MATLAB® Programming

Abishek Gopal

agopal@tamu.edu

Fall 2019 HPRC Short Course

Nov 15, 2019

Acknowledgements

- Slides adapted from the previous Introduction to Matlab short course by Dr. Jian Tao at HPRC.

- The original slides were created based on the educational materials from Kathleen Ossman and Gregory Bucks under BSD license.

Relevant Short Courses and Workshops

Introduction to the MATLAB Parallel Toolbox

https://hprc.tamu.edu/training/matlab_parallel_toolbox.html

Python for MATLAB Users

https://hprc.tamu.edu/training/python_matlab.html

Bring-Your-Own-Code Workshop

https://coehpc.engr.tamu.edu/byoc/

Offered regularly

What is MATLAB?

MATLAB is a powerful software tool for:

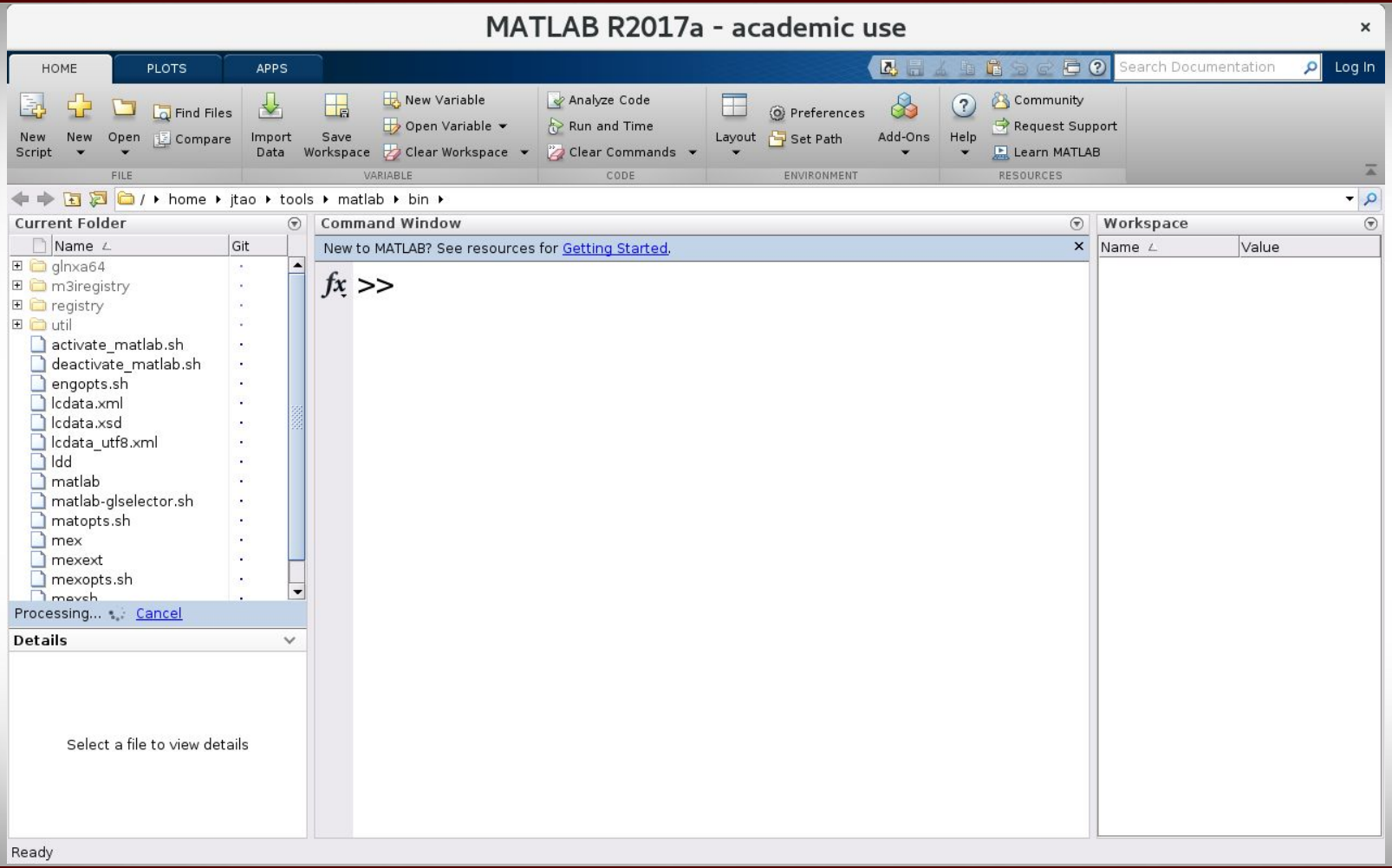
- Performing mathematical computations and signal processing
- Analyzing and visualizing data (excellent graphics tools)
- Modeling physical systems and phenomena
- Testing engineering designs
- https://www.mathworks.com/help/matlab/index.html

Industry Applications - I

- **Aircraft/Defense:** control and guidance system design and simulation, communications
- **Robotics:** design and control
- **Automotive:** cruise control, stability enhancement, fuel injection systems, hybrid power-train, sound suppression …
- **Communications:** voice over internet, cell-phone, satellite, antenna design, wireless, error coding …
- **Biotech, Pharmaceutical, Medical:** drug discovery and development, imaging procedures, cancer diagnosis …

Industry Applications - II

- **Electronics:** chip design, acoustics, voice processing and recognition
- **Industrial Automation and Machinery:** sensor design, machinery design and control
- **Utilities and Energy:** power conversion and control
- **Computers:** security systems, printer design
- **Financial:** portfolio management and risk, commodity trading, currency markets


MATLAB R2017a - academic use
HOME
PLOTS
APPS
Search Documentation
Log In
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Layout
Preferences
Set Path
Add-Ons
Help
Community
Request Support
Learn MATLAB
FILE
VARIABLE
CODE
ENVIRONMENT
RESOURCES
/ ▸ home ▸ jtao ▸ tools ▸ matlab ▸ bin ▸
Current Folder
Name
Git
glnxa64
m3iregistry
registry
util
activate_matlab.sh
deactivate_matlab.sh
engopts.sh
lcdata.xml
lcdata.xsd
lcdata_utf8.xml
ldd
matlab
matlab-glselector.sh
matopts.sh
mex
mexext
mexopts.sh
Processing... Cancel
Details
Select a file to view details
Command Window
New to MATLAB? See resources for Getting Started.
>>
Workspace
Name
Value
Ready

MATLAB Desktop

- The **Command Window** is where you type MATLAB commands following the prompt: >>
- The **Workspace Window** shows all the variables you have defined in your current session. Variables can actually be manipulated within the workspace window.
- The **Current Folder** window displays all the files in whatever folder you select to be current.

MATLAB as an Advanced Calculator

Arithmetic Operators and Order of Operations

Some Examples:

```
>>  10/5*2
>>  5*2^3+4(2)
>> -1^4
>>  8^1/3
>>  pi
```

Arithmetic Operators and Order of Operations

- Addition (+), Subtraction (-), Multiplication (*), Division (/), Power (^)
- Order of Operations (same rules you should already know from math class and using a calculator)
    1. Complete all calculations inside parentheses or brackets using the precedent rules below
    2. Powers (left to right)
    3. Multiplication and Division (left to right)
    4. Addition and Subtraction (left to right)

Variables - I

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

```
>>who                % list current variables
>>b = 1              % a scalar - 1x1 array
>>whos b             % same as who but with more info.
>>c = [1,2,3;4,5,6;7,8,9] % a matrix - 3x3 array
>>whos c
```

Variables - II

You can create your own variables.

```
>>  radius = 4
```

What do you see in your workspace window?

Now try this:

```
>>  area = pi*radius^2
```

What do you see in your workspace window now?

Naming Rules for Variables - I

- Variable names must begin with a letter

```
>>4c = 12
```

- Names can include any combinations of letters, numbers, and underscores

```
>>c_4 = 12
```

- Maximum length for a variable name is 63 characters
- MATLAB is case sensitive. The variable name A is different than the variable name a.

Naming Rules for Variables - II

- Avoid the following names: i, j, pi, and all built-in MATLAB function names such as length, char, size, plot, break, cos, log, …

```
>>clear
>>i^2
>>j^2
```

- It is good programming practice to name your variables to reflect their function in a program rather than using generic x, y, z variables.

Creating Variables & Assigning Values

At the MATLAB command prompt (>>) type:

```
x = 10.57;
```

What happens? (Hint: look at workspace window)
Several things happen with this simple MATLAB command:

- A variable, **x**, of type double is created
- A memory location for the variable **x** is assigned
- The value **10.57** is stored in that memory location called **x**.

Creating Variables & Assigning Values

At the MATLAB command prompt (>>) type:

```
x = 73.65
```

What happens?

- The old value for **x** (10.57) is replaced by the new value (**73.65**)
- Also, since the semicolon was left off the end, we see the result in the command window (as well as in the workspace window)

Exercise

In MATLAB create two variables: a = 4 and b = 17.2

Now use MATLAB to perform the following set of calculations:

$(b+5.4)^{1/3}$ $b^2-4b+5a$

Creating Strings (Text Variables)

Variables do not have to be numbers. At the MATLAB command prompt type:

```
>> month = 'Aug'
>> name = 'Adam'
>> months = {'Aug', 'Sep', 'Oct'}  %cell array
>> names = ["Adam", "Bob", "John"] %string array
>> whos
```

Displaying Variables

We can display a variable (i.e., show its value) by simply typing the name of the variable at the command prompt (leaving off the semicolon).

We can also use a function called **`disp`** to display variables. Type the following commands at the command prompt:

```
>>  disp('The value of x is:'); disp(x)
```

Numeric Data Types

Unless you specify otherwise, all numbers in MATLAB are stored as doubles.

Name	Description	Range
double	64 bit floating point	-1.79769313486232E308 to -4.94065645841247E-324 4.94065645841247E-324 to 1.79769313486232E308
single	32 bit floating point	−3.402823E38 to −1.401298E-45 1.401298E-45 to 3.402823E38
uint8	8 bit unsigned integer	Integers from 0 to 255
int8	8 bit signed integer	Integers from −128 to 127
uint16	16 bit unsigned integer	Integers from 0 to 65535
int16	16 bit signed integer	Integers from −32768 to 32767
uint32	32 bit unsigned integer	Integers from 0 to 4294967295
int32	32 bit signed integer	Integers from −2147483648 to 2147483647

Integer Data Types: int8, uint8, int16, uint16, int32, uint32

- These data types work for integers as long as the integers don't exceed the range for the data type chosen.
- They take up less memory space than doubles.
- They don't work for non-integers. If you create a variable that is an int8 and try to assign it a value of 14.8, that variable will be assigned a value of 15 instead (closest integer within the range).
- One common application for integer data types is image data (jpeg, png, …)

Why should I care how data is stored in a computer?

Perform each of the following calculations in your head.

```
a = 4/3
b = a - 1
c = 3*b
e = 1 - c
```

What does MATLAB get?

Why should I care how data is stored in a computer?

What does MATLAB get?

```
>>a = 4/3 = 1.3333
>>b = a - 1 = 0.3333
>>c = 3*b = 1.0000
>>e = 1 - c = 2.2204e-016
```

It is not possible to perfectly represent all real numbers using a finite string of 1s and 0s.

ASCII Code

When you press a key on your computer keyboard, the key that you press is translated to a binary code.

```
A = 1000001      (Decimal = 65)
a = 1100001      (Decimal = 97)
0 = 0110000      (Decimal = 48)
```

ASCII Code

ASCII stands for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Strings in MATLAB

- MATLAB stores strings as an array of characters using the ASCII code.
- Each letter in a string takes up two bytes (16 bits) and the two bytes are the binary representation of the decimal number listed in the ASCII table.

```
>> month = 'August'
>> whos
>> double(month)
>> whos
```

Some Useful Math Functions

Function	MATLAB®		Function	MATLAB®
cosine	cos or cosd		square root	sqrt
sine	sin or sind		exponential	exp
tangent	tan or tand		logarithm (base 10)	log10
cotangent	cot or cotd		natural log (base e)	log
arc cosine	acos or acosd		round to nearest integer	round
arc sine	asin or asind		round down to integer	floor
arc tangent	atan or atand		round up to integer	ceil
arc cotangent	acot or acotd			

Note: cos(α) assumes α in radians; whereas, cosd(α) assumes α in degrees. acos(x) returns the angle in radians; whereas, acosd(x) returns the angle in degrees.

Other Notes about Variables

- **clear** clears all variables in the MATLAB workspace.
- **clear a, b** just clears variables a & b.
- **clc** clears the command window
- **save MYFILE.mat** saves data for later usage in a compressed file with a .mat extension.
- **load MYFILE.mat** loads data from the .mat file to your current workspace.

Help & Doc

- The **`help & doc`** commands provide information about a function. Type **`help cos`** or **`doc cos`** at the command prompt. This only works if you know the name of the function you want help with.
- **`doc`** opens the function document in a separate window.
- **`help`** shows an abbreviated text version of the function documentation in the Command Window.

Arrays & Matrices

Create an Array

There are multiple ways to create an array in MATLAB.

```
>> a = [1 2 3 4 5 6 7 8 9]
>> b = [1 2 3;4 9 6;7 8 9]
>> c = zeros(3,3)
>> d = ones(3,3)
>> e = magic(8)
>> f = 0:10:100
>> g = rand(3,5)
>> h = eye(5)
```

Array & Matrix Operations - I

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

```
>> b + 10     %add 10 to each element
>> sin(b)     %sin function
>> b'         %transpose
>> inv(b)     %inverse
>> b*inv(b)   %matrix multiplication
>> b.*b       %element-wise multiplication
>> b.^2       %element-wise square
```

Array & Matrix Operations - II

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

```
>>B_H = [b, b]        %horizontal concatenation
>>B_V = [b; b]        %vertical concatenation
>>whos
```

The most common way to refer to a particular element in an array is to specify row and column subscripts.

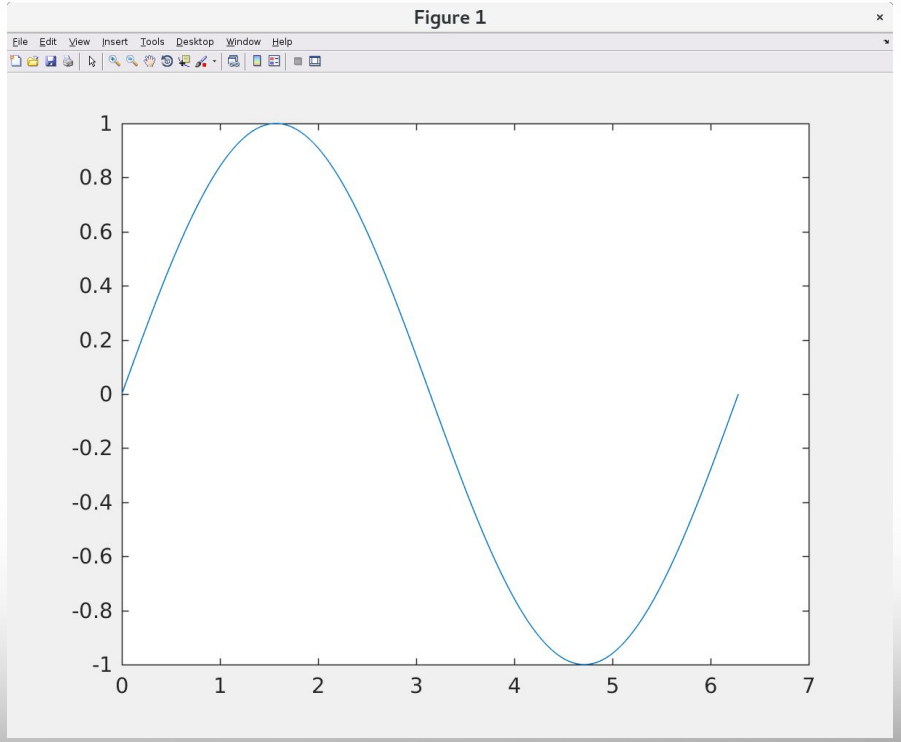
```
>>e(3, 5)
>>e(1:3, 5)
>>e(3, :)
```

2D & 3D Plots

Simple Line Plot

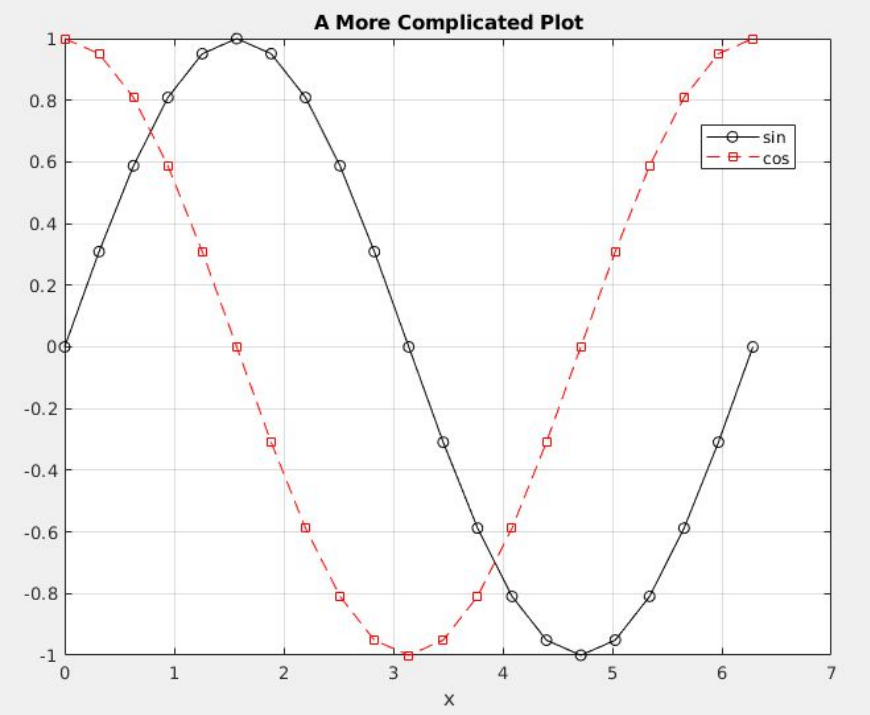
To create two-dimensional line plots, use the **plot** function.

```
>>x = 0:pi/100:2*pi;
>>y = sin(x);
>>plot(x,y)
```



A More Complicated Plot

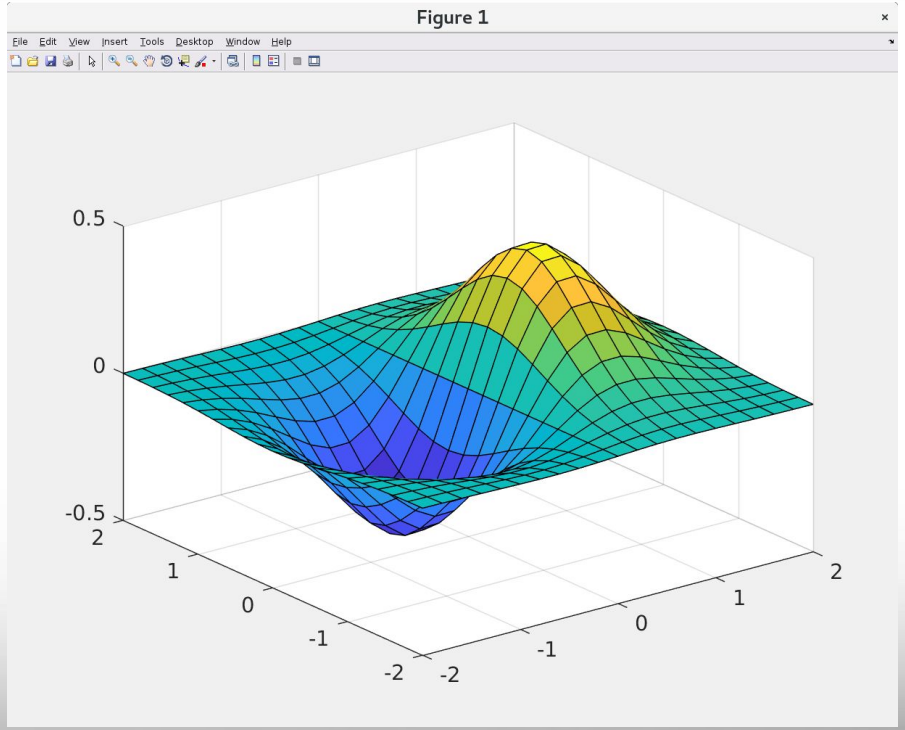
```
>> x = 0:pi/10:2*pi;
>> y = sin(x);
>> plot(x,y,'-ko')
>> hold on;
>> y2 = cos(x);
>> plot(x,y2,'--rs')
>> xlabel('x')
>> title('A More Complicated Plot')
>> legend('sin','cos')
>> grid on
```



Simple 3D Plot

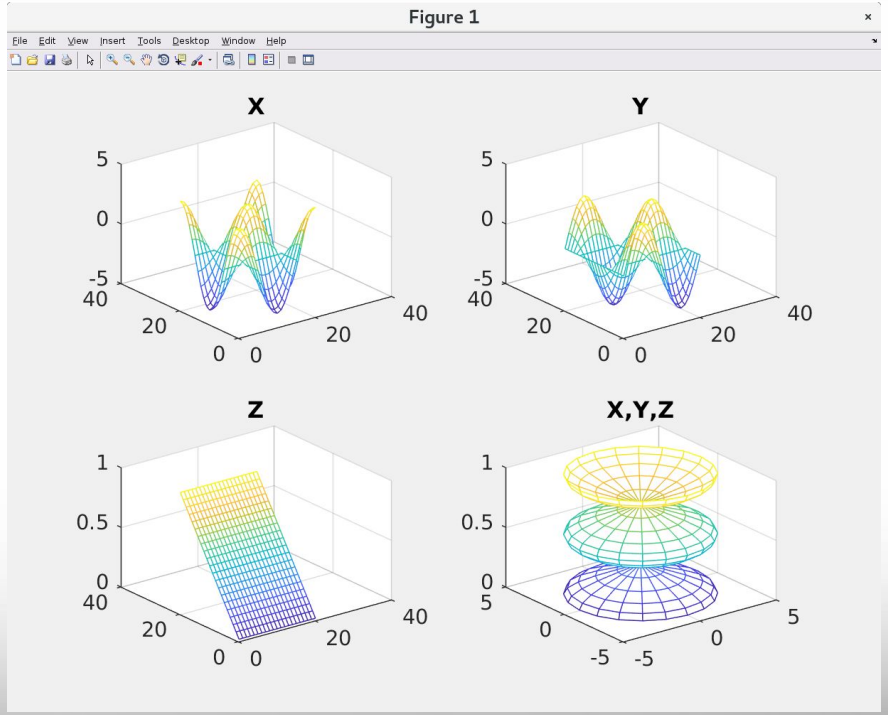
3D plots typically display a surface defined by a function in two variables, **z=f(x,y)**.

```
>>[x,y] = meshgrid(-2:.2:2);
>>z = x .* exp(-x.^2 - y.^2);
>>figure %new figure window
>>surf(x,y,z)
```



Subplots

```
>>t = 0:pi/10:2*pi;
>>[X,Y,Z] = cylinder(4*cos(t));
>>subplot(2,2,1); mesh(X);
>>title('X');
>>subplot(2,2,2); mesh(Y);
>>title('Y');
>>subplot(2,2,3); mesh(Z);
>>title('Z');
>>subplot(2,2,4); mesh(X,Y,Z);
>>title('X,Y,Z');
```



Saving plots

```
>> savefig(myfigure.fig')
>> openfig('myfigure.fig');
```

```
>> plot()
>> saveas(gcf,'myfigure.png')
>> print('myfigure','-dpng')

>> saveas
(gcf,'myfigure','epsc')

>>print ('-clipboard','-dmeta')
```

Scripts & Functions

Script Files

- A script file is simply a collection of executable MATLAB commands. To create a new script file, click on the New Script icon on the left side of the Home Tab.

Script File: Procedure

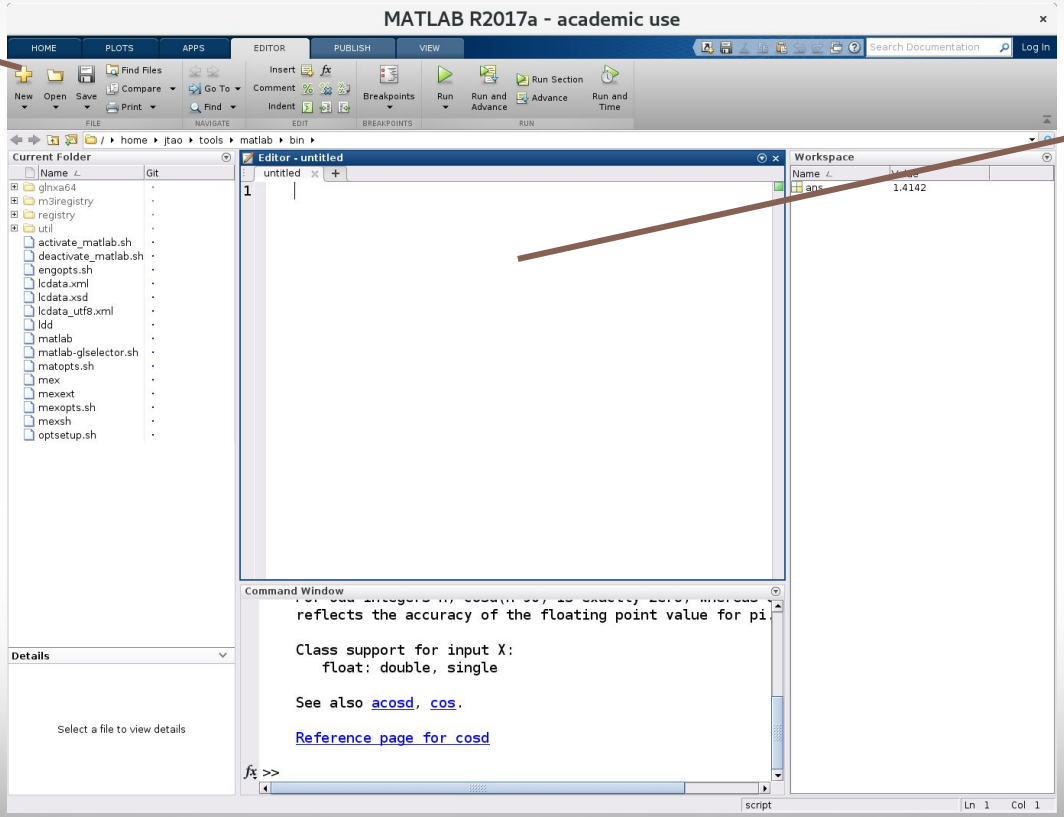
1. Type a set of executable commands in the editor window.
2. Save the file in an appropriate folder. ***When you pick a name for the file you must follow the same rules that MATLAB has for naming variables.***
3. To run the script file:
   a. hit the green **Run** Arrow in the toolbar

      or

   b. type the name of the file (without the .m extension) at the command prompt in the MATLAB command window.

Script Files

New Script



Script Editor

Exercise : New Script File

- Right click in the current folder window in MATLAB and create a new folder named whatever you would like.
- Double click on the folder to make it your current folder.
- Clear your MATLAB workspace by typing `clear` at the command prompt.
- Click on ***New Script*** to open a blank script file.
- **Write a script to calculate the area of a circle with a radius of 4cm.**
- **Run the script with the green arrow button & in the command window.**

A Bit More about Script Files

- Comments start with **%**
- A new script can be created in the Command Window with

```
>>edit SCRIPT_NAME
```

- Make use of Command History window to look for commands that were previously used.
- Run your script from the Command Window using

  >>SCRIPT_NAME

  without the .m extension

Loops & Conditional Statements

Loop Control Statements - *for*

for statements help repeatedly execute a block of code for a certain number of iterations

```
x = zeros(1,10);
for n = 1 : 10
    x(n) = n;
end
```

Loop Control Statements - *while*

while statements repeatedly execute a block of code as long as a condition is satisfied.

```
n = 1;
sum = 0;
while n <= 100
    sum = sum + n;
    n = n + 1;
end
```

Conditional Statements

Execute statements if condition is true

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

```
if a>10
    disp('a > 10');
elseif a<10
    disp('a < 10')
else
    disp('a = 10')
end
```

Nested-Loop: Simple Example

```
for r = 1:4
    for c = 1:4
        fprintf('r = %i and c = %i\n', r, c);
    end
end
```

Adding Break Statements

What if we add a break statement in the outer loop?

```
for r = 1:4
    for c = 1:4
        fprintf('r = %i and c = %i\n', r, c);
    end
    if r == 2
      break;
    end
end
```

Adding Break Statements

What if we add a break statement in the inner loop?

```
for r = 1:4
   for c = 1:4
      if r == 2
         break;
      end
          fprintf('r = %i and c = %i\n', r, c);
   end
end
```

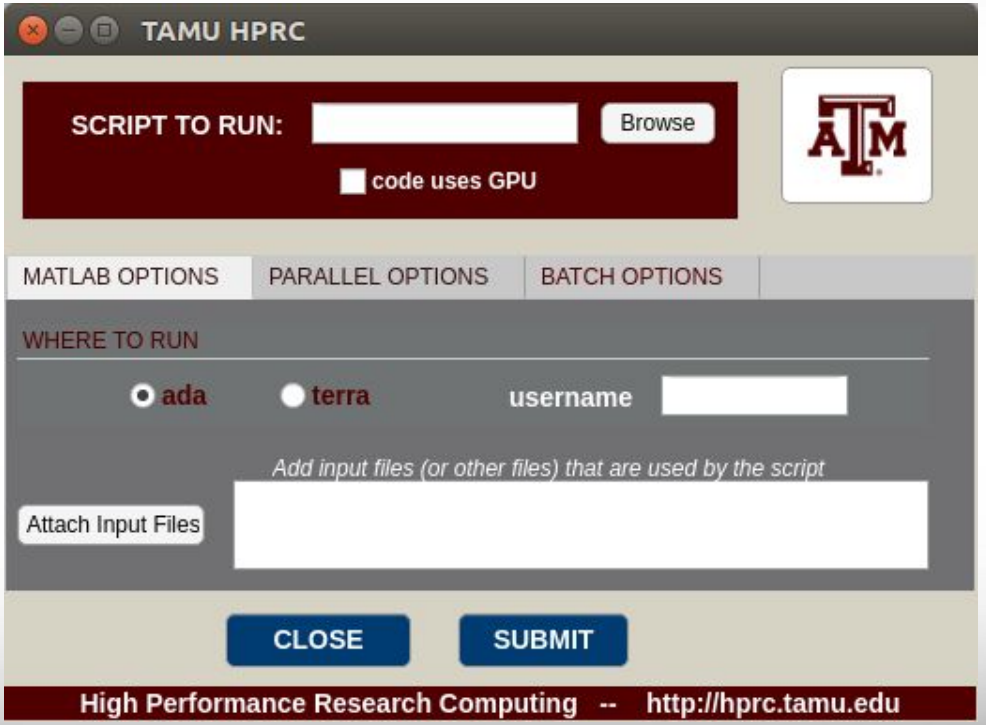
Exercise: Output

Write a script that will display each of the following shapes using asterisks *

Solid Square Open Square Triangle

HPRC MATLAB App

- Run MATLAB script directly on HPRC cluster from your personal laptop/desktop
- Need a valid HPRC account
- Download app from https://hprc.tamu.edu/wiki/SW:Matlab (section 2.1)



Appendix

Terminology

A **bit** is short for **bi**nary digi**t.** It has only two possible values: On (1) or Off (0).

A **byte** is simply a string of 8 bits.

A **kilobyte** (KB) is 1,024 (2^10) bytes.

A **megabyte** (MB) is 1,024 KB or 1,024^2 bytes.

A **gigabyte** (GB) is 1,024 MB or 1,024^3 bytes.

Data Types: double and single

- A double uses 64 bits to store a number.
- A single uses 32 bits to store a number.
- Doubles and singles can be used to represent both integers and non-integers.

How Computers Store Variables

Suppose we type the following commands in MATLAB:

```
>>  y = 42;
>>  Day = 'Friday'
```

We know MATLAB stores the values associated with the variables, y and Day, in memory.
How are these values stored?

How Computers Store Variables

Computers store all data (numbers, letters, instructions, …) as strings of 1s and 0s (bits).

A **bit** is short for **bi**nary digi**t.** It has only two possible values: On (1) or Off (0).