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

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

Current Folder

Command Window

Workspace

Select a file to view details

New to MATLAB? See resources for "Getting Started"
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:

```plaintext
>> 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.

```matlab
>> 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.

```matlab
>> radius = 4
```

What do you see in your workspace window?

Now try this:

```matlab
>> 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
  \[ \texttt{4c} = 12 \]

- Names can include any combinations of letters, numbers, and underscores
  \[ \texttt{c_4} = 12 \]

- Maximum length for a variable name is 63 characters

- MATLAB is case sensitive. The variable name \texttt{A} is different than the variable name \texttt{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:

$\sqrt[3]{b+5.4}$

$b^2 - 4b + 5a$
Creating Strings (Text Variables)

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

```matlab
>> 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**.

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

\[
\begin{align*}
    a &= 4/3 \\
    b &= a - 1 \\
    c &= 3*b \\
    e &= 1 - c
\end{align*}
\]

What does MATLAB get?
Why should I care how data is stored in a computer?

What does MATLAB get?

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

\[
\begin{align*}
A &= 1000001 \\ (\text{Decimal} &= 65) \\
a &= 1100001 \\ (\text{Decimal} &= 97) \\
0 &= 0110000 \\ (\text{Decimal} &= 48)
\end{align*}
\]
### ASCII Code

ASCII stands for American Standard Code for Information Interchange

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>Null</td>
<td>32</td>
<td>20</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Start of heading</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>Start of text</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>End of text</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>End of transmit</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>Enquiry</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>Acknowledge</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>Audible bell</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>Backspace</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>Horizontal tab</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>i</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>Vertical tab</td>
<td>42</td>
<td>2A</td>
<td>^</td>
<td>74</td>
<td>4A</td>
<td>j</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>Form feed</td>
<td>43</td>
<td>2B</td>
<td>_</td>
<td>75</td>
<td>4B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Carriage return</td>
<td>44</td>
<td>2C</td>
<td>`</td>
<td>76</td>
<td>4C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Shift out</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>Shift in</td>
<td>46</td>
<td>2E</td>
<td>+</td>
<td>78</td>
<td>4E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Data link escape</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Device control 1</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Device control 2</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Device control 3</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>Device control 4</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Device control 5</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>Negative acknowledge</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>SYNCHRONOUS IDLE</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>END TRANSMISSION BLOCK</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>CANCEL</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>END OF TRANSMISSION BLOCK</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>SUBSTITUTION</td>
<td>58</td>
<td>3A</td>
<td>A</td>
<td>90</td>
<td>5A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>ESCAPE</td>
<td>59</td>
<td>3B</td>
<td>B</td>
<td>91</td>
<td>5B</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>FILE SEPARATOR</td>
<td>60</td>
<td>3C</td>
<td>C</td>
<td>92</td>
<td>5C</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>GROUP SEPARATOR</td>
<td>61</td>
<td>3D</td>
<td>D</td>
<td>93</td>
<td>5D</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>RECORD SEPARATOR</td>
<td>62</td>
<td>3E</td>
<td>E</td>
<td>94</td>
<td>5E</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>UNIT SEPARATOR</td>
<td>63</td>
<td>3F</td>
<td>F</td>
<td>95</td>
<td>5F</td>
<td></td>
</tr>
</tbody>
</table>
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.

```matlab
>> month = 'August'
>> whos
>> double(month)
>> whos
```
## Some Useful Math Functions

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

**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.

```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)
```
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.

\[
\begin{align*}
\gg B_H &= [b, b] & \text{horizontal concatenation} \\
\gg B_V &= [b; b] & \text{vertical concatenation} \\
\gg \text{whos}
\end{align*}
\]

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

\[
\begin{align*}
\gg e(3, 5) \\
\gg e(1:3, 5) \\
\gg e(3, :) \\
\gg e(3, :) \\
\end{align*}
\]
2D & 3D Plots
Simple Line Plot

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

```plaintext
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
```
A More Complicated Plot

```matlab
>> 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

```matlab
>> 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

```matlab
>> 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.

```matlab
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.

```plaintext
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

```matlab
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?

```matlab
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?

```matlab
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)
Terminology

A **bit** is short for **binary digit**. It has only two possible values: On (1) or Off (0).

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

A **kilobyte** (KB) is $1,024 \ (2^{10}) \text{ bytes}$.  
A **megabyte** (MB) is $1,024 \text{ KB or } 1,024^{2} \text{ bytes}$.  
A **gigabyte** (GB) is $1,024 \text{ MB or } 1,024^{3} \text{ bytes}$. 
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:

```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 **binary digit**. It has only two possible values: On (1) or Off (0).