An Introduction to Using MATLAB as a Research Tool

Instructor: Dirk Colbry, Ph.D.
Institute for Cyber-Enabled Research
colbrydi@msu.edu

http://wiki.hpcc.msu.edu/x/cYI2AQ

“Learning your first computer language is like learning French poetry when you don’t know French and you don’t know poetry.”

– Bill Punch, MSU Computer Science Professor
How this workshop works

• We are going to cover some basics, lots of hands on examples, in a very short period of time.
• When you get tired of listening to me talk, skip ahead to an exercise and give it a try.
• All commands you can try are shown with the command prompt (>>). You do not need to type the “>>”
Red and Green Flags

• Use the provided sticky notes to tell me what you are doing:
  - **NO Sticky** = I am working
  - **Green** = I am done and ready to move on
  - **Red** = I am stuck and need more time and/or I could use some help
Agenda

• Motivation
• The MATLAB Interface
• MATLAB Command Syntax
• Programming with Scripts
  – Loop statements and block code
• Programming with Functions
• Loading and saving data
Main Learning Tasks

• Where to find help with MATLAB
• Getting data inside of MATLAB
• Working with data in MATLAB
• Visualizing data using MATLAB
Motivation and Background
What is MATLAB?

• (Mat)rix (Lab)oratory
  – MATLAB is a high-level programming language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

  – This is accomplished by providing the user with extensive libraries of commonly used built-in functions. These functions allow users to focus on their research goals and avoid getting overrun by many unnecessary programming details.
Alternatives to MATLAB

- R
- S-Plus
- SAS
- Mathematica
- Python
- Java
- C++
- Many more…
Why use MATLAB?

- MATLAB is designed to make it quick and easy to develop programs:
  - Uses an interpretive language, instead of a programming language that needs a compiler
  - Has an extensive library of existing functions
  - There are many existing resources online
A Few Examples

• Data Generation
• Data Analysis
• Data Visualization
The MATLAB Interface
Navigating the Program

Address of the Current Directory

Current Directory

Command Window

Workspace

Command History
Interface Style

• You can drag and drop the different components of the MATLAB interface to make the program look and feel the way you want.

• You can use the button in the upper right corner of a component to “dock” a window or use the button to undock a window.

• You can always go back to the default interface arrangement by selecting Desktop ➔ Desktop Layout ➔ Default from the MATLAB menu.
Using MATLAB as a calculator

• Try typing the following examples into the MATLAB command window:
  » 10 + 20
  » sqrt(99)
  » r = 2
  » C = 2*pi*r^2

• What variables do you see in the workspace?
Exercise: Calculate \( c \)

\[
a = 4 \\
\]
\[
b = 3 \\
\]
\[
c = \sqrt{3^2 + 4^2}
\]

Hint: Order of operations is important. Calculate the square root last.
MATLAB Variable Editor

• Set up a basic variable:
  \[
  x = 0;
  \]

• Double click on the variable in the workspace.
  – The Variable Editor window will pop up.

• Cut and paste values to and from the Variable editor to Windows excel.
Variable Editor
Command Line Navigation

- The `>>` symbol is called the “command prompt.”
- You can always double click on a command in the command history and the computer will run that line of code again.
- You can also use the up and down arrows to search though the command history.
- If you type the first few letters of a command and then use the up and down arrows, you will search only for commands starting with those letters.
Text Editor

• The editor is not in the workspace by default.
• You can start it by typing “edit” on the command line (or clicking the “new” button).
• Separate text regions by using the “%%” operator. (more about this later).
Language Syntax
Getting HELP!

• From the command line type:
  » help
  » doc

• If you do not know what a command does, type help and then the command name:
  » help plot
  » doc datatypes

• Do not be afraid to try the examples
  ─ Copy and paste the example to the command line

• Use the following commands to start over:
  » close all; clear all; clc;
Doing a help Example

• Find a help message with an example:
  » help avifile
• Copy the entire help message verbatim to the command window
• See the file this example created:
  » ls
MATLAB Central

http://www.mathworks.com/matlabcentral/

- File exchange with free MATLAB software
- Newsgroups and online help
• Resource developed as part of CSE 131 here at MSU

http://ceer.egr.msu.edu/matlab-resource
The MATLAB Interface (everything is text)

• Base expressions
  Numbers, **Strings**, +, −, *, ^, /, etc…

• Commands (functions and scripts)
  help, plot, sqrt, rand, etc.

• Variables
  x, data, ans, etc.

• Comments
  % **Ignored text.**
Matrixes Assignments

• Basic Scalar Assignment:
  » \( x = 5 \)

• Basic Vector Assignment:
  » \( v = [1 2 3 7 8] \)

• Basic Matrix Assignment:
  » \( m = [ 1 2 3 7 8 ; \ 5 2 4 5 3] \)
Displaying data out in a matrix

• Show the contents of a matrix
  – Just type variable name
    » \texttt{m}
• \texttt{M(list of rows, list of columns)}
• What are these values?
  » \texttt{m(2,5)}
  » \texttt{m(5,2)}
Matrix Multiplication

• Inverse of a matrix
  » \( \mathbf{x} = [1 \ 2; \ 3 \ 4] \)
  » \text{inv}(\mathbf{x})

• Transpose of \( \mathbf{x} \)
  » \( \mathbf{x}' \)

• Matrix Multiplication
  » \( \mathbf{x} \times \text{inv}(\mathbf{x}) \)

• Item by item Multiplication
  » \( \mathbf{x} .\* \text{inv}(\mathbf{x}) \) % notice the period
Matrix Manipulation

• Vertical Concatenation
  \[ m2 = [ v; v; m] \]

• Horizontal Concatenation
  \[ m3 = [ v \ v \ m] \]

• Accessing only the first row of a matrix
  \[ x = m2(1,:) \]

• Accessing only the first column of a matrix
  \[ y = m2(:,1) \]
The : colon character

• It can be used to define a vector of numbers
  » `x = 1:10`
  » `y = 1:2:20`
  » `z = 20:-1:1`

• It can also be used to index a matrix
  » `x = rand(10)`
  » `x(1:2, 3:5)`
  » `x(1:2, :)`
Lets get some data

- http://wiki.hpcc.msu.edu/x/cYI2AQ
- Download GROUP_YEARLY_2013
  - High Performance Computing Data
  - Fixed Width Format
Text Data Properties

• To read Text needs to be in a common format
• Delimited vs. Fixed Width
  – Space Delimited
  – Comma Delimited
• Try “Import Data”
Generally more than one solution

- Open file in Excel, copy and paste the results to MATLAB
- Open file in Excel, convert to csv and open in MATLAB using “csvread”
- Use MATLAB “Import Data” wizard in the file menu
- Use MATLAB “textread” command
- Others?
Which do you pick

• Try one, if it doesn’t work, try another
• Sometimes finding a solution can be frustrating
• The trick is to keep trying
• Google is your friend
Getting a feel for the data

- \text{sum}(x)
- \text{mean}(x)
- \text{stddev}(x)
- \text{plot}(\text{sort}(x))
- \text{semilogy}(\text{sort}(x))
- \text{hist}(x)
Data Questions

• What is the maximum CPUh?
• What group generated the max CPUh?

• Hint: help max

• We will come back to this later
Basic Command Syntax

\[ [\text{output1}, \text{output2}, \ldots] = \text{command}(\text{input1}, \text{input2}, \ldots); \]

- **Command name**
  - This is the name of the script or function.
  - Both functions and scripts have command names, however, scripts do not have inputs or outputs.
  - The command name is normally the same name as the file which defines the command.
  - Typing “help <command name>” will cause the help message for that command to appear.
  - The command name is case sensitive, but MATLAB will search for the closest match if the case sensitive one is not found.
Command Name Examples

• Example Commands:
  » `figure`
  » `rand`
  » `ls`

• Type ‘help’ and then the command names.
• Type ‘open’ and then a command name.
  - Warning: you can edit commands that are open in the editor. Be careful to not make or save any changes to built-in MATLAB commands!

• Try adding capital letters to commands:
  » `LS`
  » `RAND`
  » `Figure`
Basic Command Syntax

\[ [\text{output1}, \text{output2}, \ldots] = \text{command}(\text{input1}, \text{input2}, \ldots); \]

- **Inputs:**
  - Comma separated list in parentheses.
  - A function is able to take different numbers of inputs and may perform differently for different numbers of inputs.
  - String inputs must be surrounded by single quotes.
  - If the inputs are all strings, the parentheses, commas and single quotes can be replace with white space.
    - Note: in this special case, no outputs will be assigned.
  - Note: scripts do not have inputs.
Input Examples

• Example commands with inputs:
  » `rand(2);`

• Example of different behavior (overloading)
  » `linspace(0,2*pi)`
  » `linspace(0,2*pi,10)`

• Special case with strings as the only input
  » `ls('c:\')`
  » `ls c:\`
  » `clear all`
Basic Command Syntax

\[
[\text{output1}, \text{output2}, \ldots] = \text{command}(\text{input1}, \text{input2}, \ldots);
\]

• Assignment and output
  – Comma separated list of variables in brackets.
  – A function may perform differently depending on the number of outputs that are requested.
  – If only one output is required, then the brackets and commas are not needed.
  – If the assignment and output variables are removed the system will automatically assign `output1` to `\text{ans}`, the default output variable.
  – Note: scripts do not have outputs.
Output Examples

• Example commands with outputs:
  » x = rand([1,2])
  » f = figure
  » im = imread('ngc6543a.jpg')
  » h = image(im)
  » [x, y] = ginput(1)

• Using the default assignment
  » rand(1)
  » sqrt(26)

Note: if you are working with images consider the image processing toolbox and the newer imshow command.
Ex: The Max Function

• Help max
• max(x)
  – What if x is a vector?
  – What if x is a matrix?
• [m i] = max(x)
• What is i?
• Which group is the maximum group?
Basic Command Syntax

\[ \text{output1, output2, \ldots} = \text{command(input1, input2, \ldots)}; \]

- Display Output semicolon (Optional)
  - If the semicolon is not included, then MATLAB will automatically display the contents of the output variables to the terminal display.
  - If the semicolon is included, then the command will run “quietly” and not output to the terminal display.

- Semicolon also ends a command
  - Two commands can be placed on the same line of input
Semicolon Examples

• Display results
  » `x = linspace(0,2*pi)`

• Do not display results
  » `x = linspace(0,2*pi);`

• More than one command on a line
  » `y = sin(x); plot(x,y);`
Overloading

• Functions can change what they do based on the type and number of inputs and outputs.

```matlab
» x = linspace(1,100);
» y = rand([100 1]);
» y = sort(y);
» plot(x,y);
» plot(x, y, '*r');
```

Same function different numbers of inputs and different results.
Naming Commands and Variables

• There are special characters that cannot be used in names, including:
  \( \langle \text{space}\rangle, : \, \setminus \, * \, \& \, + \, - \, ( \, ) \, [ \, ] \, \{ \, \} \, \# \, \% \, @ \text{ etc…} \)

• Names should be short and make sense

• Try not to reuse existing command and variable names

• Some good names include:
  - beedata
  - timedata
  - videoplotfun
  - etc.

• Some bad names include:
  - sqrt
  - var
  - a, b, c, d, e,
  - etc.
Project 1:
Fitting Polynomial Functions

• Use the following set of functions to input data, display them in a figure and fit a polynomial to the data. (Hint: use the help command.)

```
figure
axis
ginput
plot
polyfit
hold
ezplot
```
Example Review

```matlab
>> figure;
>> axis([0 100 0 100]);
>> [x y] = ginput(10);
>> plot(x,y,'dr');
>> p = polyfit(x,y,1)

p =

0.8415   6.6390

>> hold on;
>> ezplot('0.8415*x + 6.6390', [0 100 0 100]);
>> hold off;
```

Function will wait until you click on your figure 10 times:

Results will vary depending on what points you clicked
Text and Title Commands

» help title

• Sometimes you get strange results
  » figure
  » title('hello_world')

• This is because MATLAB uses a tex interpreter to display mathematical functions
  » xlabel('2\pi r^2');

• Most of the time you do not want to use the tex interpreter.
  » ylabel('time_seconds', 'Interpreter', 'none');

• If you want to learn how to use the tex interpreter, you can just Google tex or latex and read about the math environment.
• The output of one command can be the input to another command.
  – The value of the input will be the same as output of the nested command.
  – Nesting can continue as long as you like.

\[ \text{output1, output2, …} = \text{command(command2(), input2, …);} \]
Example Nested Commands

• Here is an example of a non-nested command:
  » `x = rand([100 1]);`
  » `y = sort(x);`
  » `plot(y);`

• Or using nested commands:
  » `plot(sort(rand([100 1])));`

• Note: there is only one semi-colon.
Data Types
(skipping)
Numeric
(integer, single, double, unit8, etc)

• A double is the default numeric class in MATLAB
• Numeric operators include:
  (+ add) (- subtract) (* multiply) (/ divide) (^ power)
• The different numeric datatypes are needed to represent different classes of numbers:
  – Floating points
  – Negative numbers
  – Memory requirements
• A double will be able to do most of what you want. It can represent large floating point numbers with negative and positive values.
Casting

• Changing from one numerical type to another

• If you want to change from a floating point to an integer
  – \texttt{round(5.6)} or \texttt{uint64(5.6)}

• If you want to change an integer to a double you need to cast
  – \texttt{double(x)}
Memory Storage

• A bit is a one (1) or a zero (0)
• A byte is eight bits (a byte is the smallest amount of data represented in MATLAB)
• Different datatypes have different sizes
  » clear all
  » d = double(10);
  » ui8 = uint8(10);
  » ui32 = uint32(10);
  » ui64 = uint64(10);
  » s = single(10);
Examples

• Integers are required to index a matrix
  » \texttt{x = rand(5);}
  » \texttt{x(1,2)}
  » \texttt{x(1.5,2.5) \%This causes an error}

• Color images are normally represented by a three dimensional matrix (rows, columns, color) of uint8.
  – In other words: three, two dimensional arrays representing red, green and blue.
  – Each item in this 3D matrix is traditionally represented by a number from 0-255, which is an 8 bit binary number.
(Char)acter

• A char is a number between 0 and 65535.
  – How many bits is this?
• Each number is mapped to a specific letter in the alphabet; like a code.
• Different languages and fonts can have different mappings.
• ASCII is a universal standard for mapping the characters on a keyboard to one of the first 127 numbers.
<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>NUL</td>
<td>20</td>
<td>0</td>
<td>040</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>&amp;#64;</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>140</td>
<td>&amp;#96;</td>
<td>`</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH</td>
<td>21</td>
<td>0</td>
<td>041</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>&amp;#65;</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>141</td>
<td>&amp;#97;</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX</td>
<td>22</td>
<td>0</td>
<td>042</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>&amp;#66;</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>142</td>
<td>&amp;#98;</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX</td>
<td>23</td>
<td>0</td>
<td>043</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>&amp;#67;</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>143</td>
<td>&amp;#99;</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT</td>
<td>24</td>
<td>0</td>
<td>044</td>
<td></td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>&amp;#68;</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>144</td>
<td>&amp;#100;</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ</td>
<td>25</td>
<td>0</td>
<td>045</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>&amp;#69;</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>145</td>
<td>&amp;#101;</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK</td>
<td>26</td>
<td>0</td>
<td>046</td>
<td>&amp;#38;</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>&amp;#70;</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>146</td>
<td>&amp;#102;</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL</td>
<td>27</td>
<td>0</td>
<td>047</td>
<td></td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>&amp;#71;</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>147</td>
<td>&amp;#103;</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS</td>
<td>28</td>
<td>0</td>
<td>050</td>
<td></td>
<td>72</td>
<td>48</td>
<td>110</td>
<td>&amp;#72;</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>150</td>
<td>&amp;#104;</td>
<td>h</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>TAB</td>
<td>29</td>
<td>0</td>
<td>051</td>
<td>(</td>
<td>73</td>
<td>49</td>
<td>111</td>
<td>&amp;#73;</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>151</td>
<td>&amp;#105;</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF</td>
<td>30</td>
<td>0</td>
<td>052</td>
<td></td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>&amp;#74;</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>152</td>
<td>&amp;#106;</td>
<td>j</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT</td>
<td>31</td>
<td>0</td>
<td>053</td>
<td></td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>&amp;#75;</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>153</td>
<td>&amp;#107;</td>
<td>k</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF</td>
<td>32</td>
<td>0</td>
<td>054</td>
<td></td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>&amp;#76;</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>154</td>
<td>&amp;#108;</td>
<td>l</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR</td>
<td>33</td>
<td>0</td>
<td>055</td>
<td></td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>&amp;#77;</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>155</td>
<td>&amp;#109;</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO</td>
<td>34</td>
<td>0</td>
<td>056</td>
<td></td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>&amp;#78;</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>156</td>
<td>&amp;#110;</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI</td>
<td>35</td>
<td>0</td>
<td>057</td>
<td></td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>&amp;#79;</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>157</td>
<td>&amp;#111;</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE</td>
<td>36</td>
<td>0</td>
<td>058</td>
<td></td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>&amp;#80;</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>160</td>
<td>&amp;#112;</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1</td>
<td>37</td>
<td>0</td>
<td>059</td>
<td></td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>&amp;#81;</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>161</td>
<td>&amp;#113;</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2</td>
<td>38</td>
<td>0</td>
<td>060</td>
<td></td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>&amp;#82;</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>162</td>
<td>&amp;#114;</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3</td>
<td>39</td>
<td>0</td>
<td>061</td>
<td></td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>&amp;#83;</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>163</td>
<td>&amp;#115;</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4</td>
<td>40</td>
<td>0</td>
<td>062</td>
<td></td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>&amp;#84;</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>164</td>
<td>&amp;#116;</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK</td>
<td>41</td>
<td>0</td>
<td>063</td>
<td></td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>&amp;#85;</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>165</td>
<td>&amp;#117;</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN</td>
<td>42</td>
<td>0</td>
<td>064</td>
<td></td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>&amp;#86;</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>166</td>
<td>&amp;#118;</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB</td>
<td>43</td>
<td>0</td>
<td>065</td>
<td></td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>&amp;#87;</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>167</td>
<td>&amp;#119;</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN</td>
<td>44</td>
<td>0</td>
<td>066</td>
<td></td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>&amp;#88;</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>170</td>
<td>&amp;#120;</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM</td>
<td>45</td>
<td>0</td>
<td>067</td>
<td></td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>&amp;#89;</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>171</td>
<td>&amp;#121;</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB</td>
<td>46</td>
<td>0</td>
<td>072</td>
<td></td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>&amp;#90;</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>172</td>
<td>&amp;#122;</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC</td>
<td>47</td>
<td>0</td>
<td>073</td>
<td></td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>&amp;#91;</td>
<td></td>
<td>123</td>
<td>7B</td>
<td>173</td>
<td>&amp;#123;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS</td>
<td>48</td>
<td>0</td>
<td>074</td>
<td></td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>&amp;#92;</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td>174</td>
<td>&amp;#124;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS</td>
<td>49</td>
<td>0</td>
<td>075</td>
<td></td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>&amp;#93;</td>
<td>]</td>
<td>125</td>
<td>7D</td>
<td>175</td>
<td>&amp;#125;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS</td>
<td>50</td>
<td>0</td>
<td>076</td>
<td></td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>&amp;#94;</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>176</td>
<td>&amp;#126;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US</td>
<td>51</td>
<td>0</td>
<td>077</td>
<td></td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>&amp;#95;</td>
<td>_</td>
<td>127</td>
<td>7F</td>
<td>177</td>
<td>&amp;#127;</td>
<td>DEL</td>
<td></td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Understanding Characters

• A string is just a vector of characters:
  » `x = 'hello'`
  » `y = 'world'`
  » `x + y`
  » `[x y]`
  » `[x ' ' y]`

• An integer from 0-255 can be turned into a character:
  » `x = [72 73];`
  » `x = char(x)`

• Or you can change a character back into its number:
  » `x = 'Hello World';`
  » `double(x);`
Numbers and Character Paradox

• Here is an odd example:
  » x = '5'
  » double(x);
• Why does it print out 53 and not 5?
• We could subtract 48 and get the number.
• Or we can use a built in functions:
  − \texttt{str2double} and \texttt{num2str}
String Compare - strcmp

• Compare two strings and return a boolean
  » h1='hello'; h2='world';
  » h1==h2  % doesn’t work
  » sum(h1 == h2)  % doesn’t work
  » sum(~(h1 == h2))  % doesn’t work
  » sum(~(h1 == h2)) == 0  % works
  » sum(~(h1 == h1)) == 0  % works

• Or use strcmp, which is much easier
  » strcmp(h1, h2)
  » strcmp(h1, h1)
Why doesn’t this work?

• List of strings
  » \( \mathbf{x}(1,:) = 'Hello everybody'; \)
  » \( \mathbf{x}(2,:) = 'Ha Ha'; \)
  » \( \mathbf{x}(3,:) = 'Thank you, come again'; \)
  » \( \mathbf{x}(4,:) = 'Eat my shorts'; \)
  » \( \mathbf{x}(5,:) = 'Excellent'; \)
  » \( \mathbf{x}(6,:) = 'D''oh'; \)
Cells (note \{curly\} brackets)

- List of strings
  - \texttt{x\{1\}} = 'Hello everybody';
  - \texttt{x\{2\}} = 'Ha Ha';
  - \texttt{x\{3\}} = 'Thank you, come again';
  - \texttt{x\{4\}} = 'Eat my shorts';
  - \texttt{x\{5\}} = 'Excellent';
  - \texttt{x\{6\}} = 'D''oh';
Scalar $\rightarrow$ Vector $\rightarrow$ Matrix

- These are the most restrictive container class, but also the most widely used.
  - i.e., all of the components of the vector or matrix much be of the same data type and size.

- Accessing a Vector or Matrix:

  $X(1,2) \leftarrow$ returns the component of the first row and the second column.
Cell → Cell Array

• A Cell is a container for any type of object. A Cell array allows you to make an array of objects that vary in type or size.
• Example cell array:

\[
x = \{ '100' \ 100 \ 10000 \ 'hello world' \}
\]
• Accessing a cell array:

\[
x{1} \leftarrow \text{returns the contents of the first cell}
\]
\[
x(1) \leftarrow \text{returns the first cell as a cell}
\]
• Examples to try:

\[
x{5} = 'bob';
\]
\[
x(5)
\]
\[
x{5}
\]
A struct is a structure of data types in MATLAB. These structures are also called objects.

Example struct:

```matlab
>> X.bob = 10;
>> X.cat = 20;
>> X.hello = 'Good day';
```

Example struct array:

```matlab
>> d = dir

13x1 struct array with fields:
   name
date
   bytes
   isdir
```

Accessing a struct array:

- `d.name` returns all of the names in the array.
- `d(4).name` only returns the name of the fourth struct.
Printing more complex output

» help sprintf

• There are special characters that can be used in a formatted string:
  - \t – tab
  - \n – new line
  - \\ – ‘\’ backslash character
  - ’’ – single quote

• Example:
  » sprintf('Dirk''s email:\n\tdirk@colbry.com\n')
Programming With Scripts
Scripts
(The “Dark Side” of MATLAB programming)

• Scripts are “seductively” easy, but will cause you a lot of problems in the long run.
• Most of the time you want to use a function instead.

• However, we will be using scripts in our examples.
• Just remember, not to give in to the “Dark Side”
Scripts

• Put all of your commands in a single text file (you can use MATLAB’s built-in editor).
• Name the file with the .m extension (filename.m).
• Type in the text file name to run the commands.
• Script do not have their own workspace. Instead, they use the current workspace. (I will explain this more when I talk about functions.)
% This is a comment. The system will ignore anything with a comment.

% This is an example script program.

% This script plots some two dimensional data on the screen and then fits
% some curves to the data.

% [X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);

% Extra commands that are commented out.
%hold on;
%surf(X,Y,Z);
%hold off;
%colormap cool;
%axis([-3 3 -3 3 -10 5])
Crop Image Example

• A grayscale image is a matrix of values between 0 and 255.

```matlab
im = imread('ngc6543a.jpg');
image(im);

im2 = im(70:530, 90:520, :);
image(im2);
```

• Note: Images can get warped
  − (type “axis off equal;” to see a clean image).
Block Code
“if / else” Statement

• If something is true do x, otherwise, do something else.

```python
x = input('Enter a number and then enter ');  
if(x > 9)  
    % This code will only execute if x > 9  
    disp('Number is greater than 9');  
else  
    % This code will only execute if x ~= 9  
    disp('Number is less than 9');  
end
```
Truth Statements

• Relationship Operators
  • Logical Operators

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

& - logical AND
| - logical OR
~ - logical NOT
“for” Statement

• Cycle though a vector one item at a time

```plaintext
figure;
hold on;
a = [0 100 0 100];
axis(a);
for i = 1:10
    [x(i) y(i)] = ginput(1);
    plot(x,y,'*');
    axis(a);
end
```
Group Practice

Let's turn this into a script (hint: use `num2str`)

```matlab
>> figure;
>> axis([0 100 0 100]);
>> [x y] = ginput(10);
>> plot(x,y,'dr');
>> p = polyfit(x,y,1)

p =

0.8415    6.6390
```

```matlab
>> hold on;
>> ezplot('0.8415*x + 6.6390', [0 100 0 100]);
>> hold off;
```
Response time experiment

• Write a script that measures the response time of a user.

• Outline of the task:
  – Describe research objective
  – Flow chart the program
  – Look up the necessary functions
  – Write the program
Project 2: 
Response time experiment

• Write a script that waits for a random amount of time between 1 and 2 seconds and then asks for user input (return key). Repeat 20 times.

  for, end
  rand
  pause
  tic, toc
  beep
  input
  hist

• Display a histogram showing how long it took between prompting the user and getting a response.
“while” Statement

• Keep doing something while a statement is true.

```python
x = input('Type a number and then enter ');
while (x != 9)
    x = input('Type a number and then enter ');
end
```
Consecutive if statements

```matlab
x = input('Type in a number and press <enter> '); 
if(x == 1) 
    disp('one');
else 
    if(x == 2) 
        disp('two');
    else 
        if(x == 3) 
            disp('three');
        else 
            disp('more than three');
        end
    end
end
```
“switch / case” Statement

• Simple way to display a series of if statements.

```python
x = input('Type in a number and press <enter> ');
switch(x)
    case(1)
        disp('one');
    case(2)
        disp('two');
    case(3)
        disp('three');
    otherwise
        disp('more than three');
end
```
“try / catch” Statement

• Try to do a command, if there is an error, address it and move on.

```matlab
name = input('Type in an image file name with '' marks ');
try
    im = imread(name);
    image(im);
catch
disp('could not open file');
end
disp('program did not exit');
```
Block code Review

- **if / else** – Do the “if” block only if the statement is true. If the statement is not true, do the “else” block.

- **for** – Do block for a fixed number of times.

- **while** – Keep doing a block while a statement stays true.

- **switch/case** – Switch between blocks based on different cases of a variable.

- **try/catch** – Try a block. If the block fails, catch the error and do this other block.

- **end** – The end of a Block.
Function Programming
Functions

• Functions take a set of inputs and return a separate set of outputs.
• Functions have their own workspace.
  - This makes naming variables easier because different workspaces can have the same variable name.
Functions

- To change a script into a function the following line needs to be the first line in your file:

```matlab
function [outputlist] = name(input list)
```
Example Function (imagecrop.m)

```matlab
function im2 = imagecrop(imname)
% Written by Dirk Colbry
% 01-27-2014
% Tool to select and crop an image

im = imread(imname); done=false;

while(~done)
    image(im); axis off; axis equal;
    title('Select upper right corner of cropped area')
    [x1,y1] = ginput(1);

    title('Select lower right corner of cropped area')
    [x2,y2] = ginput(1);
    im2 = im(y1:y2, x1:x2, :);
    image(im2); axis off; axis equal;

    in = input('Is this correct (Yes/No)', 's');
    if (strcmp(in,'Yes'))
        done = true;
    end
end
```

Function Declaration

‘Help’ Comment Block

Program
Every function has its own workspace

- When a function starts, its workspace only contains the inputs to the function (plus some special variables).

- When a function exits, only the output variables are in the main workspace.

- Variables that are inside and outside of the workspace are different, regardless of the variable names.

- For instance, if the variable ‘x’ is in the main workspace and there is also a variable named ‘x’ in my function workspace, they can have different values and it will not cause an error.
Try it out

» help imagecrop
» im = imread('ngc6543a.jpg');
» cr = imagecrop(im);
» close all;
» image(cr)
Scripts vs. Functions

• Why Scripts are bad:
  – They share the same variable space (workspace) as the main program.
  – So, every time you need a new variable you have to make sure that you did not use the same name in the past or it could cause unwanted errors

• Why Functions are good:
  – Each function has its own variable space.
  – Functions make your code simple because any change you want to make only needs to be made once.
  – Functions help you organize your code.
Loading and Saving Data

File I/O
Saving and restarting MATLAB

• At any point you can save your MATLAB session:
  >> save(‘mysession’);

• Then you can exit MATLAB and reload your session latter:
  >> load(‘mysession’);
Types of files

• Just like variables, every file is a group of numbers.
• The program needs to know what the numbers mean in order to read the files.
• Since the numbers could mean anything, some standards have been adopted that make reading the file easier.
• There are generally two major classes of files, ASCII and Binary.
All files are given a file ID

• The `fopen` command opens a file and returns the file ID.
• Any command that can read or write to a file will normally take the file ID as an input.
  - `fread`, `fwrite`, `fprintf`, `fgets`, `fgetl`, `fscanf`, `fseek`, etc.
• After you are done accessing the file you should always use the `fclose` command.
• `fid = fopen(filename, permissions)`

• The permissions string can include:
  
  - `'r'`    read
  - `'w'`    write (create if necessary)
  - `'a'`    append (create if necessary)
  - `'r+'`   read and write (do not create)
  - `'w+'`   truncate or create for read and write
  - `'a+'`   read and append (create if necessary)
  - `'W'`   write without automatic flushing
  - `'A'`   append without automatic flushing
Example Function

function showfile(filename)
    %SHOWFILE - display the contents of a file as ASCII

    fid = fopen(filename, 'r');

    while 1
        tline = fgetl(fid);
        if ~ischar(tline)
            break
        end
        disp(tline)
    end
    fclose(fid);
Text (ASCII) files

• In a text file, the list of numbers is taken from the ASCII table.
• Many programs can read text files (Notepad, MATLAB, etc).
• Some common text formats are:
  – Web pages (.html)
  – MATLAB programs (.m)
  – Text file (.txt)
Special ASCII files

• MATLAB can read any file. However, you need to tell MATLAB what you want it to mean.
  – Line Delimited files
  – Space Delimited files
  – Comma Delimited files
Binary files

- Binary files are more compact than text files. However, it is difficult to load binary files because the format of the file is unknown.
- Some binary files follow a known standard. The file extension tells the computer which standard is being used:
  - Image files (bmp, jpg, etc)
  - Sound files (mp3, wav, au, etc)
  - Proprietary formats (doc, pdf, mat, etc)
Specific I/O Commands

• General
  – load / save

• ASCII
  – csvread / csvwrite – comma separated data
  – dlmread / dlmwrite – ASCII delimitated data
  – textscan – specialized format data

• Binary
  – wk1read / wk1write – lotus notes spreadsheet file
  – xlsread / xlswrite – excel files
  – imread / imwrite – image files
  – aviread / aviwrite – movie files
Solution to Group Practice

```matlab
figure
axis([0 100 0 100]);
[x y] = ginput(10);
plot(x,y, 'dr');
p = polyfit(x,y,1);
hold on;
equ_str=[num2str(p(1)) 'x + ' num2str(p(2))];
ezplot(equ_str, [0 100 0 100]);
hold off;
```
Solution to Project 2

```matlab
for i = 1:20
    pause(rand(1)*2);
    tic;
    x = input('press the (enter) key');
    t(i) = toc;
end
hist(t);
```