

**Armstrong State University**  
**Engineering Studies**  
**MATLAB Marina – Introduction to MATLAB Primer**

**Prerequisites**

The Introduction to MATLAB Primer assumes no prior knowledge of MATLAB or computer programming.

**Learning Objectives**

1. Be able to use the MATLAB Command Window to perform numeric computations.
2. Be able to use MATLAB's help to look up information on functions, keywords, and operators.
3. Be able to use the MATLAB IDE and MATLAB editor to create and execute scripts.

**Terms**

MATLAB, IDE, main desktop, command window, command history, workspace, current folder, search path, variable, script, program, comment, syntax highlighting

**MATLAB Functions, Keywords, and Operators**

help, =, ans, clear, clc, close all, who, whos

**MATLAB**

MATLAB is a high level interpreted language. The compilation step is generally hidden from users. Each command in a MATLAB program is compiled and executed one at a time when encountered. Solutions can be produced fairly quickly compared to a compiled language. MATLAB has a large library of predefined functions and a large collection of toolboxes and is particularly useful for numerical calculations involving arrays and matrices.

MATLAB can be launched by double clicking on the MATLAB icon on the desktop or choosing MATLAB from the Windows Start menu.

Videos providing an overview of new MATLAB features by release changes can be found at the MathWorks site,

[http://www.mathworks.com/products/matlab/whatsnew.html?s\\_tid=main\\_release\\_ML\\_rp](http://www.mathworks.com/products/matlab/whatsnew.html?s_tid=main_release_ML_rp)

**MATLAB IDE**

The MATLAB IDE (integrated development environment) has a main desktop with several panes. There are three main panes in the default MATLAB main desktop: the Command Window (center), Workspace (right), and Current Folder (left), see Figure 1a. Additional windows for plotting and creating scripts can be opened and panes/windows can be docked and undocked from the MATLAB main desktop. Across the top of the main desktop starting at the left side is the MATLAB Toolstrip. The Toolstrip has three tabs (HOME, PLOT, APPS) which contain icons for commands and action menus. The top right of the main desktop has the Quick

Access Toolbar and the Search Bar. The Current Folder Bar is on the left of the main desktop just below the Toolstrip.

MATLAB has a built in editor, see Figure 1b, which can be used to create and run MATLAB scripts (programs). The editor window is docked as the top center pane in the default layout; Figure 1b shows the editor as it would be if undocked as a separate window. MATLAB scripts are sequences of statements in a text file that are compiled and executed one statement at a time.

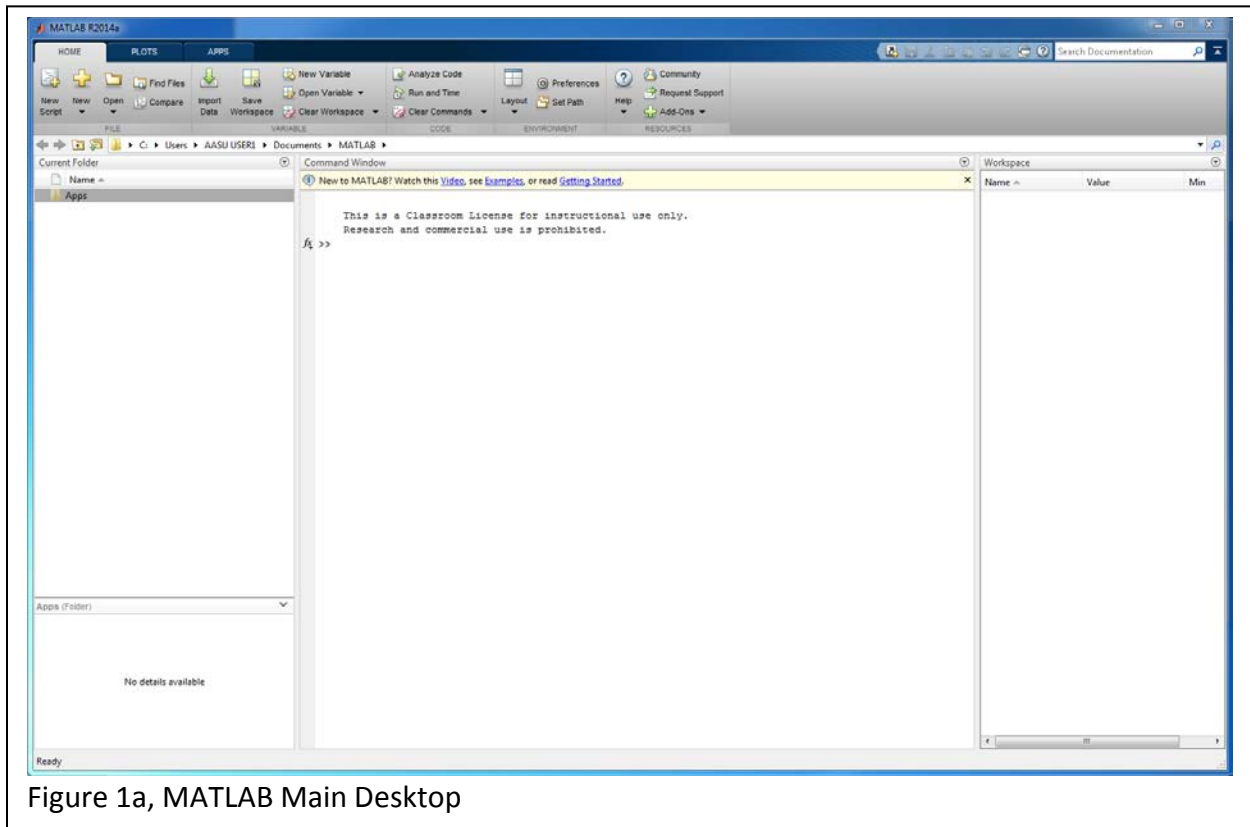


Figure 1a, MATLAB Main Desktop

The default layout can be restored by selecting Default from the Layout Toolstrip icon.

### **MATLAB Current Folder and Workspace**

The Current Folder pane shows the default folder/directory that MATLAB saves files to. The pane shows the files in the folder and allows new folders and files to be created.

The Workspace pane shows all of the variables that have been defined and their values. Variables can be selected by double clicking on them which will open the Variable Editor. The Variable Editor can be used to inspect or modify the contents of a variable.

### **MATLAB Command Window**

MATLAB statements and programs can be executed in the Command Window pane. MATLAB can be used like a calculator if commands are typed and executed in the Command Window. The basic arithmetic operations addition, subtraction, multiplication, division, and power are

indicated by the symbols +, -, \*, /, and ^ respectively. You have the option of using parenthesis ( ) to indicate order of precedence. Recall, expressions are evaluated from left to right. Powers have the highest precedence, followed by multiplication and division, followed by addition and subtraction. Spaces may be used to separate things in statements to make the statement easier to read. Spaces in statements are generally ignored by MATLAB. After the enter key is hit, MATLAB returns the result of the executed statement.

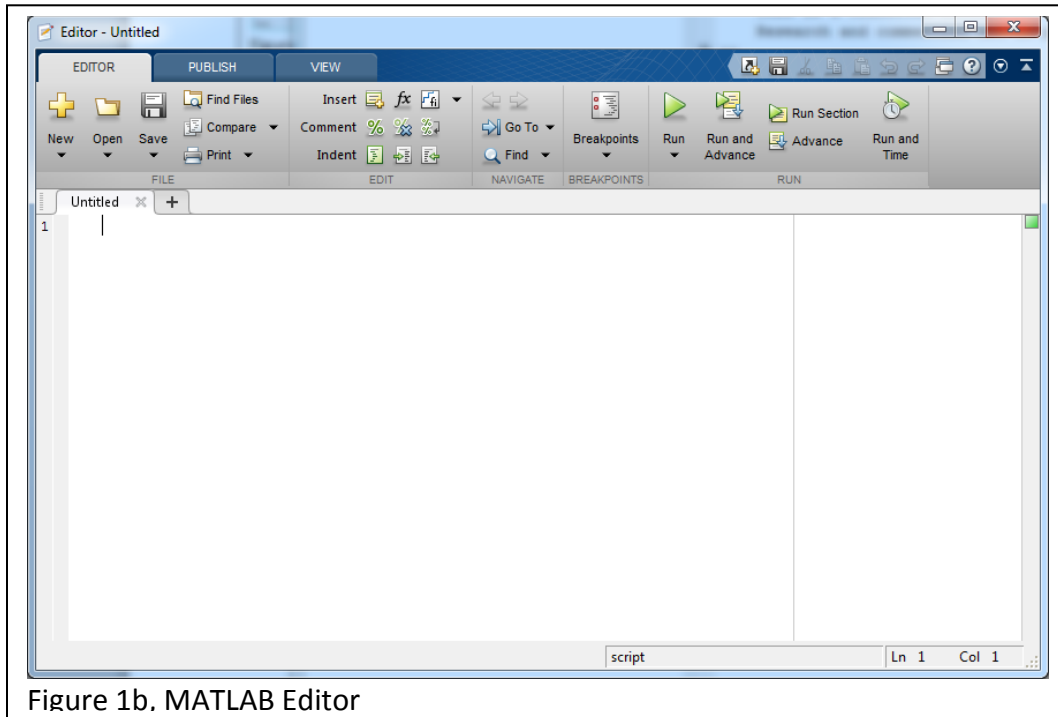


Figure 1b, MATLAB Editor

Figure 2a shows examples of evaluating the expressions  $(8 + 2 \times (-1.5)) \div 4$  and  $(81^{0.5} + 2 \times (-3)) \div 6$  in the Command Window (>> is the Command Window line prompt).

```
>> (8 + (2 * -1.5))/4
ans = 1.2500
>> result = (81^0.5 + (2 * -3))/6
result = 0.5000
```

Figure 2a, Evaluating Expressions using MATLAB Command Window

Variables are names referring to data stored in a computer's memory. The equal sign (=) allows one to assign the result of the right hand side of a statement to a variable on the left hand side of the statement. The right hand side of a statement is evaluated according to operator precedence and the resulting value is assigned to the variable on the left hand side. The left hand side of the assignment operator (=) can only have a single variable. If no variable is given for the result, MATLAB places the result in the default variable `ans`. The result of a statement is also displayed in the Command Window. The result displayed in the command window can be suppressed by ending commands with a semicolon. Once a variable has a value, it can then be used in other statements. Variables can be redefined by assigning a new value(s) to them.

Figure 2b shows an example of assigning values to two variables and then using the variables in a statement.

```
>> length = 5.0;
>> height = 2.7;
>> rectangleArea = length * height
rectangleArea = 13.5000
```

Figure 2b, Use of Variables in Statements

Variables can be cleared from the workspace using the `clear` statement. Individual variables are deleted by `clear` followed by the variable name. The entire workspace (all variables) is deleted by using `clear` with no argument. The Command Window can be cleared and the cursor homed using the `clc` statement.

The MATLAB statement `who` returns a list of all currently defined variables. The statement `whos` returns a list of all currently defined variables, variable sizes, numbers of bytes used, and class. The currently defined variables and their values can also be seen in the Workspace pane. The MATLAB workspace (all currently defined variables) can be saved to a data file using the `save` function. The statement `save data1;` will save the workspace to the file `data1.mat`. This file is a binary file that can be loaded into MATLAB using the `load` function.

```
>> a = 10;
>> b = 5;
>> c = 7;
>> who
Your variables are:  a  b  c
>> clear b
>> whos
  Name      Size      Bytes      Class      Attributes
  a         1x1         8         double
  c         1x1         8         double
>> clear all
>> who
>>
```

Figure 2c, Use of clear, who, and whos

### MATLAB Pop Up Command History

The command history which was a separate pane at the lower right of the IDE in MATLAB releases prior to R2014a is now a pop window. The pop up command history can be accessed by hitting the up arrow in the Command Window. The pop up command history shows statements that have been executed in the Command Window in chronological order.

Previously executed statements can be accessed from the pop up command history using the  $\uparrow$  key and the  $\downarrow$  key (this allows one to scroll through past statements) or the mouse. The statement can then be edited or executed in the Command Window.

### MATLAB Functions

Functions are program modules that are passed arguments and return a result. MATLAB has built in functions for most mathematical, engineering, and science functions that are commonly needed. MATLAB functions are called by their name. The variables or data inside the parenthesis of the call are the arguments for the function and the result from a function is saved by assigning it to a variable. Pay attention to the domain, the range, and the units of the MATLAB functions. For instance, the built in cosine function, `cos`, takes an angle in radians. MATLAB also has a built in function, `cosd`, that takes an angle in degrees.

Figure 3 shows an example of using the MATLAB square root and cosine functions to evaluate the law of cosines. Notice that functions can be called from within statements and the results of functions can either be immediately used or saved in a variable.

```
>> sideA = 4;
>> sideB = 6;
>> thetaC = pi/3;
>> sideC = sqrt(sideA^2+sideB^2-2*sideA*sideB*cos(thetaC))
sideC = 5.2915
```

Figure 3, Use of MATLAB Square Root and Cosine Functions

Some commonly used MATLAB functions are:

MATLAB Built in Functions	
Function name	Operation
<code>cos, acos</code>	cosine, inverse cosine
<code>sin, asin</code>	sine, inverse sine
<code>tan, atan</code>	tangent, inverse tangent
<code>log, log10, log2</code>	natural log, common log (base 10), base 2 log
<code>exp</code>	exponential function
<code>sqrt</code>	Square root
<code>abs</code>	absolute value (magnitude of complex number)
<code>angle</code>	phase angle
<code>real</code>	real part of a complex number
<code>imag</code>	imaginary part of a complex number
<code>conj</code>	complex conjugate

Table 1, Commonly Used Built in MATLAB Functions

## MATLAB Help

Help on built in functions and other MATLAB topics can be accessed from the Help window and from the Command Window. The Help window can be opened from the Help in the HOME tab of the Toolstrip or from the Help icon (?) in the Quick Access Toolbar. The Help window is shown in Figure 3a and the doc help for the cosine function is shown in Figure 3b. The Help window will also be opened when searching the MATLAB documentation using the Search Bar.

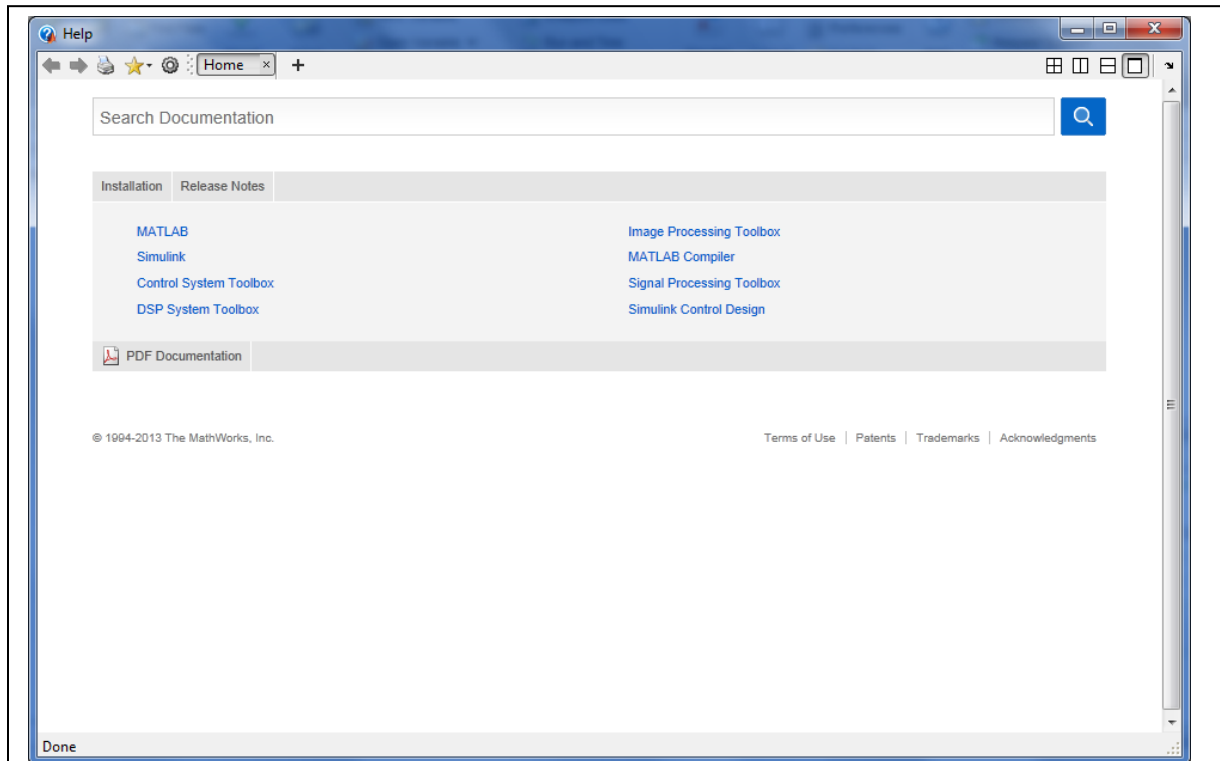


Figure 4a, MATLAB Help Window

The Command Window help is best used for finding information about functions and operators one knows the name/symbol of. Typing the statement `help` followed by the name of the function/operator will provide information about that function/operator in the Command Window. Figure 4c shows an example of MATLAB's Command Window help for the cosine function.

```
>> help cos
cos    Cosine of argument in radians.
      cos(X) is the cosine of the elements of X.
      See also acos, cosd.

      Reference page in Help browser
      doc cos
```

Figure 4c, MATLAB Command Window Help

Online documentation for MATLAB can be found at the MathWorks website  
<http://www.mathworks.com/help/index.html>

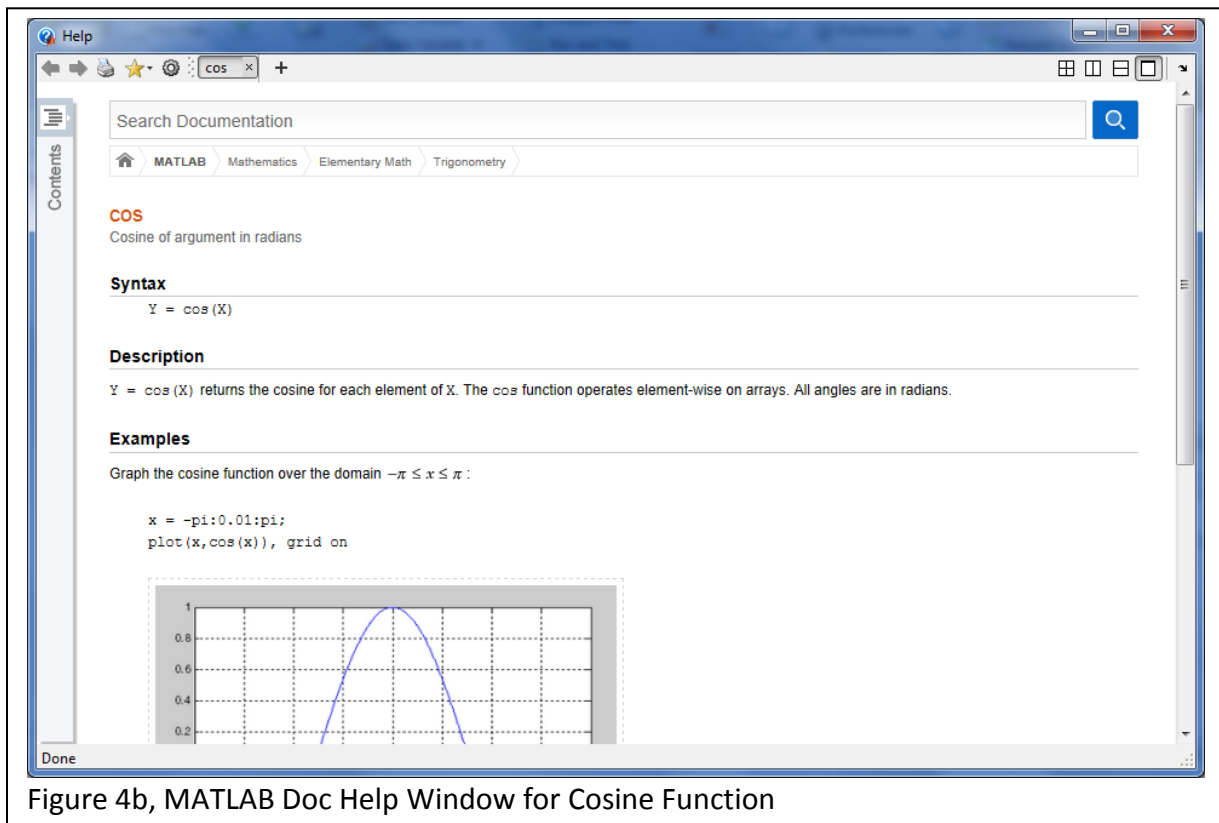


Figure 4b, MATLAB Doc Help Window for Cosine Function

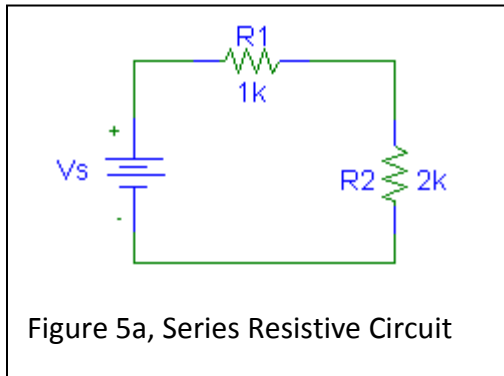
### MATLAB Scripts

MATLAB scripts or programs are sequences of MATLAB statements that can be used to execute sequences of MATLAB statements. Scripts can be created in any text editor, but the MATLAB editor has syntax highlighting for MATLAB keywords, reserved words, and many MATLAB constructs; and allows breakpoints to be set for debugging. The MATLAB editor can be opened using the New Script icon on the HOME tab of the Toolstrip or by selecting Script from the New menu on the HOME tab of the Toolstrip. The script can be saved from the Editor Toolstrip. The script will be saved in the current folder unless a different is specified. Script files must have the extension .m.

Statements in the MATLAB script file are executed in sequential order just as if all the statements were typed in the MATLAB Command Window. Statements in the script, when executed, have access to all current workspace variables, and anything created or modified by statements in the script will be saved in the workspace. Normally, the MATLAB statements in the script file are not displayed as they are executed. The results of these commands are displayed though. Semicolons placed after MATLAB statements have the effect of suppressing the result of the statements.

Scripts can be run two ways: either by typing the script name in the Command Window or by selecting Run from the Editor tab of the Editor Toolstrip. MATLAB expects scripts to be in the current folder or in one of the default search paths. MATLAB's processing of a script can be interrupted at any time by hitting `Ctrl - C` (control C).

Figure 5b shows MATLAB a script that solves for the voltages and currents in the series resistive circuit of Figure 5a. For conciseness, the program header describing what the script does has been omitted.



```
clear;
clc;
close all;

% source voltage (Volts)
Vs = 10;
% resistances (Ohms)
R1 = 1000;
R2 = 2000;

% compute current in loop
I = Vs/(R1 + R2);
% compute voltage across each resistor
VR1 = I * R1;
VR2 = I * R2;

% display the voltages and current
disp(I), disp(VR1), disp(VR2)
```

Figure 5b, circuitAnalysis Script

Notice that sections of the script are separated using blank lines and comments. Comments are denoted in MATLAB using the percent sign (%). Everything after the percent sign is a comment and is ignored by the MATLAB interpreter. Multiple statements can be placed on the command line if they are separated by commas or semicolons.



There are MATLAB statements that are rarely used in the Command Window but are commonly used in scripts. One of these is `input`. The `input` function displays a string as a user prompt and reads in a line from the user. Instead of setting the source voltage to a constant 10 Volts in the script of Figure 5b, the source voltage could be read from the user using the statement:

```
Vs = input('Enter the source Voltage in Volts: ');
```

Figure 5c shows a script that determines that initial velocity of a water balloon thrown straight up for it to reach one mile. The program header is omitted for conciseness.

```
% acceleration due to gravity in ft/s^2
a = -32.14;
% initial and final position in feet
x0 = 0.0;
x = 5280.0;    % one mile
% final velocity in ft/s
v = 0.0;

% initial velocity to reach one mile
v0 = sqrt(v^2 - 2*a*(x-x0));
disp(v0)
```

Figure 5c, balloonHeight Script

The balloonHeight script was developed as follows:

- Problem is described: two dimensional motion can be described using kinematic equations relating position to velocity, acceleration, and time  $x - x_0 = v_0 t + \frac{1}{2} a t^2$  and relating velocity to position and acceleration  $v^2 = v_0^2 + 2a(x - x_0)$ ; where  $x$  is position ( $x_0$  is the initial position),  $v$  is velocity ( $v_0$  is the initial velocity),  $a$  is acceleration, and  $t$  is time.
- Problem is solved analytically: only the second equation relating velocity to position and acceleration is needed, solving for the initial velocity the equation  $v_0^2 = v^2 - 2a(x - x_0)$  is obtained, and the initial velocity is the square root of  $v_0^2$ ,  $v_0 = \sqrt{v^2 - 2a(x - x_0)}$
- Inputs and outputs of the script were determined: output is the initial velocity, there are no inputs (although a more general problem might have the altitude to reach as the input) but the initial position, final position, and acceleration due to gravity need to be specified
- Sequence of operations is determined and operations are specified using one or more MATLAB statements: define acceleration due to gravity, define initial and final position, define final velocity, compute initial velocity using equation, display initial velocity

Take advantage of the syntax highlighting when writing programs using the MATLAB editor. Keywords show up in blue, comments show up in green, unterminated strings show up in maroon and completed strings in purple. The rest of the program text will be black. Syntax

errors are indicated by red squiggly lines under the text and warnings are indicated by orange squiggly lines under the text.

Last modified Tuesday, August 19, 2014



This work by Thomas Murphy is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](https://creativecommons.org/licenses/by-nc-nd/3.0/).