A Brief Introduction to Engineering Computation with MATLAB

By:
Serhat Beyenir

A Brief Introduction to Engineering Computation with MATLAB

By:
Serhat Beyenir

Online:
< http://cnx.org/content/col11371/1.11/ >

OpenStax-CNX

Collection structure revised: November 17, 2015
PDF generated: February 24, 2016
For copyright and attribution information for the modules contained in this collection, see p. 156.

Table of Contents

Acknowledgements[1]

I would like to express my appreciation to all of my students in the Power and Process Engineering program between the academic years 2011-12 and 2015-16 for their feedback and understanding that this book is a work in-progress.

Special thanks go to those who supported the book along the way:

- Dr. David Porter[2], the founder of BCIT OER Group in 2015 who constantly provides me with the latent energy to continue working on this book,
- Mr. Alex Podut with whom I co-taught MATLAB in 2015. His input was very valuable,
- Dr. Sanja Boskovic who promoted this book at management levels in 2012,
- Mr. Sergiy Yatlo who gave feedback to the first iterations of the book in early 2011.

[1] This content is available online at <http://cnx.org/content/m50977/1.3/>.

[2] BCIT, Associate Vice President, Education Support and Innovation (<http://cnx.org/content/m50977/latest/ http://commons.bcit.ca/update/2015/05/bcit-welcomes-new-associate-vice-president-education-support-and-innovation/>)

Preface[3]

In my tenth year at the Institute, I dedicate this book to the BCIT community.

The primary purpose of writing a book and distributing it free-of-charge is to extend my gratitude to BCIT[4] . I am particularly thrilled to do it with this textbook because it is a product of many learning opportunities BCIT has offered me over a period of several years. What follows is a brief background on how this book came to be.

My post-secondary teaching career began on 22 January 2001 at the Pacific Marine Training Campus of BCIT when I logged on to a Unix workstation to instruct in the Propulsion Plant Simulator. That has been a major milestone in many ways in my professional life. While learning inner workings of Unix operating system (OS), I also made a discovery and that discovery profoundly changed my view on how I thought the world operated. The discovery was the GNU/Linux OS and open source software (OSS) movement through several books, most notably Just for Fun: The Story of an Accidental Revolutionary[5] and The Cathedral and the Bazaar[6]. I was convinced that the collective power of connected individuals around the world and the global infrastructure of the Internet had the potential to change the ways the world functioned.

In the last 10 years, BCIT has allowed me to study various subjects through its Professional Development (PD) programs for which I am very grateful. I learned a great deal in PD courses and in one of the recent ones, I had two déjà vu moments similar to my discovery of OSS movement. The first one occurred when I began reading The Wealth of Networks[7] and the second one when I found about Connexions[8] . The former was a confirmation of my 10-year old discovery and the latter is what I am using to write this book. Connexions is a web-based curricular content authoring and publishing technology that I believe has a growing potential for writing and distributing free-of-charge learning materials.

Thus, motivation for this book stems from the notions that were generated by the OSS movement. The book was written to pay a small token of appreciation to BCIT and I hope it will be a contribution to the open educational resources repository.

Serhat Beyenir
North Vancouver, B. C.
25 October 2011

[3] This content is available online at <http://cnx.org/content/m41458/1.6/>.
[4] http://www.bcit.ca/
[5] *Just for Fun: The Story of an Accidental Revolutionary* by L. Torvalds and D. Diamond, New York: HarperCollins Publishers. ©2001
[6] *The Cathedral and the Bazaar* by E. S. Raymond, Sebastopol: O'Reilly Media. ©1999
[7] *The Wealth of Networks* by Y. Benkler, New Haven: Yale University Press. ©2006
[8] http://cnx.org/

Study Guide[9]

MATLAB, a sub-course of Computer Technology 1 and this text are specifically designed for students with no programming experience. However, students are expected to be proficient in First Year Mathematics and Sciences and access to good reference books are highly recommended. I also assume that students have a working knowledge of the Mac OS X or Microsoft Windows operating systems.

The strategic goal of the course and book is to provide learners with an appreciation for the role computation plays in solving engineering problems. The MATLAB specific skills that I would like students to acquire are as follows:

- Write scripts to solve engineering problems including interpolation, numerical integration and regression analysis,
- Plot graphs to visualize, analyze and present numerical data,
- Publish reports.

The best way to learn about engineering computation is to actually do it. We will therefore solve many engineering problems mainly using a recent version of MATLAB in this book. Since the primary focus is engineering computation, we will concentrate on the mathematical solutions and, to a limited extent, the graphical user interface (GUI) features of MATLAB.

Learning a new skill, especially a computer program, can be an overwhelming experience. To make the best of this process, students are encouraged to observe the following guidelines that have proven to work well:

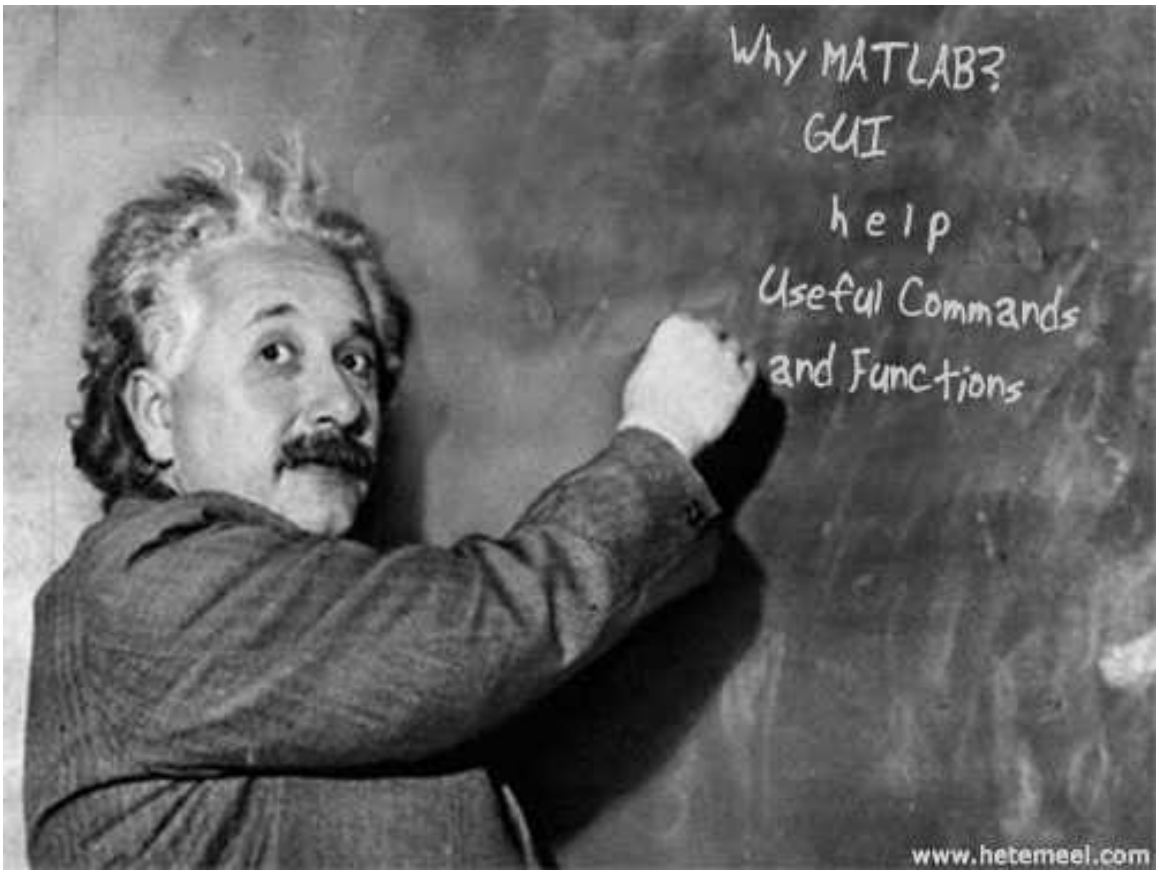
- Plan to study 2 hours outside of class for every hour inside of class,
- Practice, practice, practice: As the old saying goes, practice makes one perfect or perhaps we should modify that statement: Good practice makes one perfect,
- Buddy system: Study with a classmate. Helping one another drastically improves your understanding of the material. Particularly, students are advised to work the problem sets in this fashion,
- Muddy points: Make a note of muddy points as they may occur during lectures and email your notes to me. I will address those issues at the beginning of the next class,
- Open book exam: Do not try to memorize commands, functions or their syntax but learn where and how to find that information. Through many exercises and problem sets you will have solved by the end of the course, most computational routines will become second nature to you. The exam is open book, so keep your learning materials and m-files well organized.

[9] This content is available online at <http://cnx.org/content/m41459/1.2/>.

Chapter 1

Introduction

1.1 What is MATLAB?[1]



MATLAB stands for MATrix LABoratory (see wikipedia[2]) and is a commercial software application written by The MathWorks, Inc.[3] When you first use MATLAB, you can think of it as a glorified calculator allowing you to perform engineering calculations and plot data. However, MATLAB is more than an advanced scientific calculator, for example MATLAB's sophisticated numerical computation environment also allows

[1] This content is available online at <http://cnx.org/content/m41403/1.5/>.
[2] http://en.wikipedia.org/wiki/MATLAB
[3] http://www.mathworks.com/

us to analyze data, simulate engineering systems, document and share our code with others.

1.1.1 Why Use MATLAB?

MATLAB has become a defacto standard in many fields of engineering and science. Even a casual exploration of MATLAB should unveil its computational power however a closer look at MATLAB's graphics and data analysis tools as well as interaction with other applications and programing languages prove why MATLAB is a very strong application for technical computing.

The standard MATLAB installation includes graphics features to visualize engineering and scientific data in 2-D and 3-D plots. We can interactivity build graphs and generate MATLAB command output that can be saved for use in the future. The saved-instructions can be called again with different data set to build new plots. The plots created with MATLAB can be exported in various file formats (e.g. .jpg, .png) to embed in Microsoft Word documents or PowerPoint slideshows.

MATLAB also contains interactive tools to explore and analyze data. For example, we can visualize data with one of the many plotting routines, zoom in to plots to take measurements, perform statistical calculations, fit curves to data and evaluate the obtained expression for a desired value.

MATLAB interacts with other applications (e.g. Microsoft Excel) and can be called from C code, C++ or Fortran programming language.

1.1.2 Running MATLAB

To use MATLAB, it must be installed on your computer and you can start it just like you start any application on your system or you must have access to a network where it is available.

In POWR 3307, we will use MATLAB by accessing the BCIT network. The network access is platform independent, that is, we can run MATLAB under Mac OS X or Microsoft Windows operating systems through a web browser. The following links provide instructions on how to access and use BCIT's AppsAnywhere service:

Configuring AppsAnywhere on an Apple Macintosh[4]
Configuring AppsAnywhere in Windows 7[5]
How to open and save files in AppsAnywhere when logging in from a Macintosh[6]
How to open and save files in AppsAnywhere when logging in from Windows[7]

1.1.3 The MATLAB Desktop

When you start the MATLAB program, it displays the MATLAB desktop. The desktop is a set of tools (graphical user interfaces or GUIs) for managing files, variables, and applications associated with MATLAB. The first time you start MATLAB, the desktop appears with the default layout, as shown in the following illustration.

[4] http://kb.bcit.ca/sr/AppsAnywhere/1346.html
[5] http://kb.bcit.ca/sr/AppsAnywhere/1345.html
[6] http://kb.bcit.ca/sr/AppsAnywhere/807.html
[7] http://kb.bcit.ca/sr/AppsAnywhere/806.html

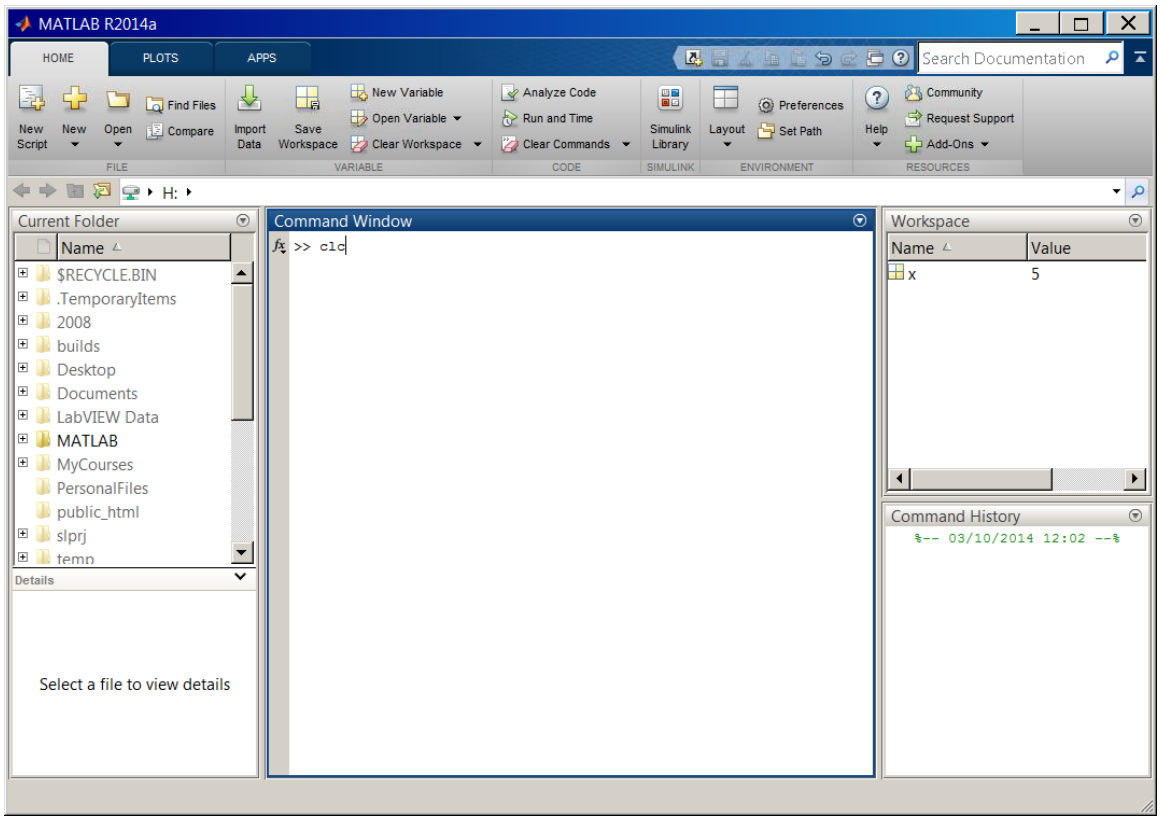


Figure 1.1: The MATLAB Desktop.

1.1.3.1 Command Window

The Command Window is where we execute MATLAB commands. We enter statements at the Command Window prompt. The prompt can be any one of the following:

- `Trial`≫ indicates that the Command Window is in normal mode and the MATLAB license will expire after the trial period ends.
- `EDU`≫ indicates that the Command Window is in normal mode, in MATLAB Student Version.
- ≫ indicates that the Command Window is in normal mode.

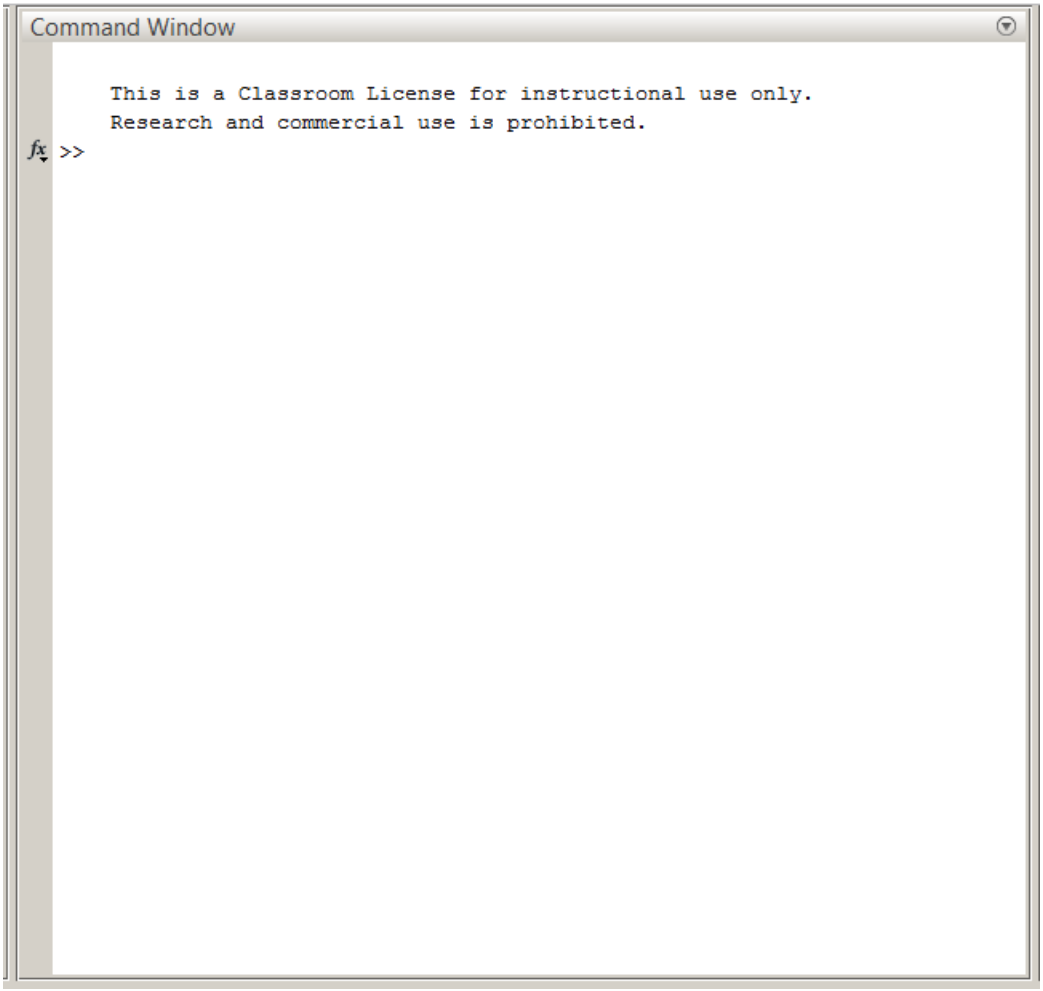


Figure 1.2: The Command Window.

1.1.3.2 Command History

The Command History is a log of the commands we have executed in the command window.

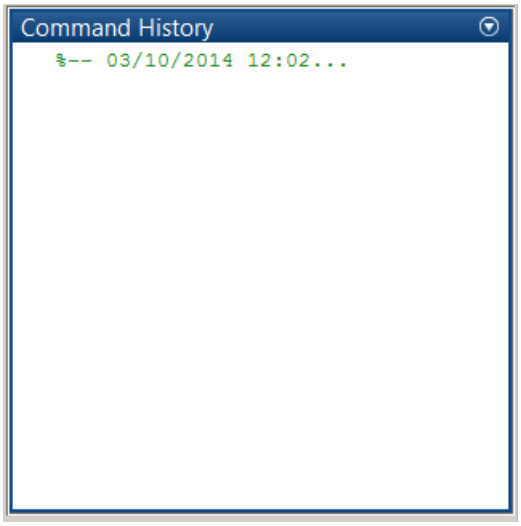


Figure 1.3: The Command History.

1.1.3.3 Workspace

The workspace consists of a set of variables stored in memory during a MATLAB session. To open the Workspace browser, select Desktop > Workspace in the MATLAB desktop, or type

```
≫ workspace
```

at the Command Window prompt.

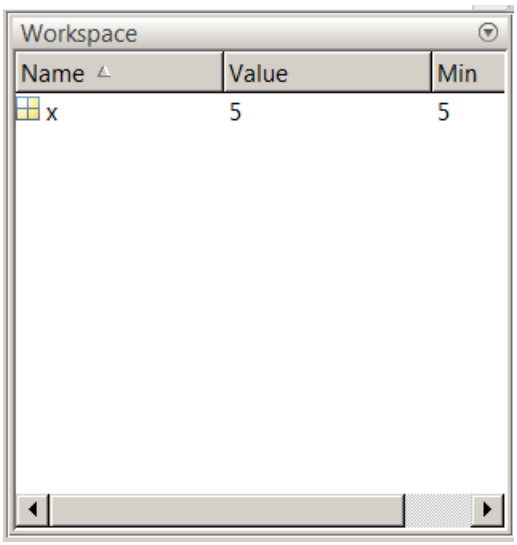


Figure 1.4: Workspace.

1.1.3.4 Current Folder

The Current Folder is like the Finder in Mac OS X or Windows Explorer in Windows operating systems and allows us to browse through the files and folders. The Current Folder also displays details about files in your current directory and within the hierarchy of the folders it contains.

Figure 1.5: Current Folder.

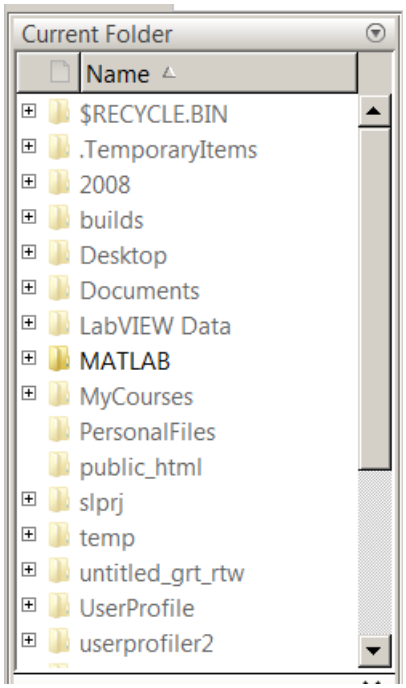


Figure 1.6: Current Folder docked on the desktop.

1.1.3.5 Tool Strip

The tool strip contains global tabs, Home, Plots and Apps. Contextual tabs become available when you need them.

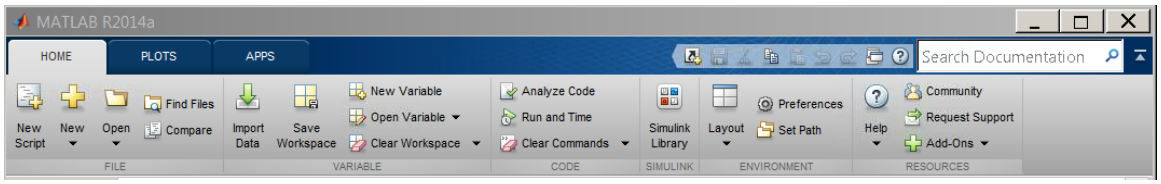


Figure 1.7: Tool Strip.

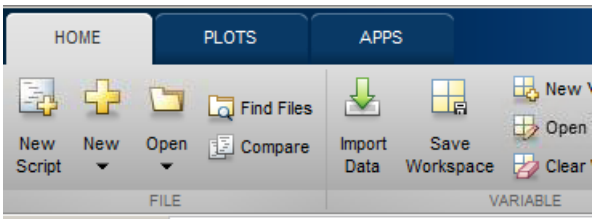


Figure 1.8: Tabs.

The plots tab allows us to plot various types of graphs quickly and easily.

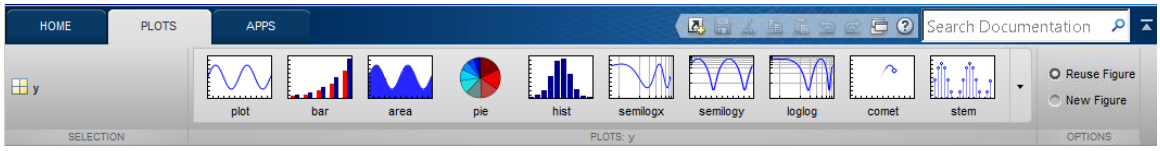


Figure 1.9: The plots tab.

The apps tab gives quick access to interactive applications within MATLAB environment.

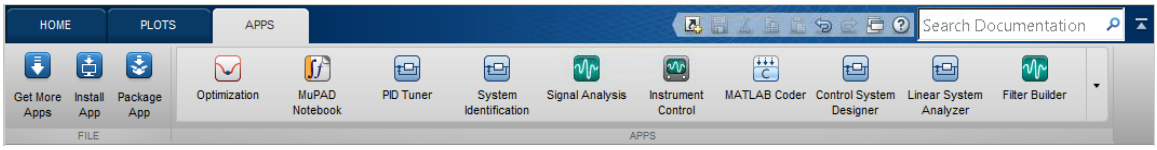


Figure 1.10: The apps tab.

Layout button allows us to change the desktop layout or go back to the default configuration.

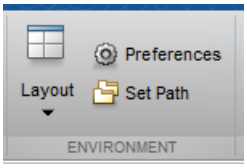


Figure 1.11: Layout button.

1.1.3.6 Toolbar

The MATLAB toolbar provides on-screen buttons to access frequently used features such as, copy, paste, undo and redo.

Figure 1.12: Toolbar.

1.1.3.7 Keyboard shortcuts

MATLAB provides keyboard shortcuts for viewing a history of commands and listing contextual help.

1. The up arrow key,
2. The tab key,
3. The semicolon symbol.

1.1.3.7.1 The Up Arrow Key

Suppose we want to enter the following equation:

≫ `y=sin(45)`

But we mistakenly entered

≫ `y=sine(45)`

MATLAB returns the following prompt:

```
??? Undefined function or method 'sine' for input arguments of type 'double'.
```

Instead of retyping the equation, press the up arrow key, the mistakenly entered line is displayed. Using the left arrow key, move the cursor to the misspelled letter. Make the correction and press Return or Enter to execute the command.

Pressing the up arrow key repeatedly recalls the previously entered commands. Likewise, typing the first characters of previously entered line and pressing the up arrow key displays the full command line. To execute that line, simply press the Return or Enter key.

1.1.3.7.2 The Tab Key

Suppose you forgot how to enter the square root command. Begin typing `y=sq` in the command prompt:

≫ `y=sq`

Then press the tab key and scroll down to `sqrt`. Select it and press Return or Enter key.

≫ `y=sqrt`

1.1.3.7.3 The Semicolon Symbol

The semicolon symbol at the end of a line suppresses the screen output. This is useful when you want to keep your command window clean.

Type the following entry and press the Return key:

```
≫ y=2+2
```

The following output is displayed:

```
y =

 4
```

Now, press the up arrow key to recall our initial entry

```
≫ y=2+2
```

And insert a semicolon as follows:

```
≫ y=2+2;
```

No numerical result is displayed however MATLAB stores the value of y in the memory. We can recall the value y by simply typing y and pressing Return.

1.1.4 MATLAB Help

MATLAB comes with three forms of online help: help, doc and demos.

1.1.4.1 Help

Typing help in the Command Window lists all primary help topics. You can display a topic by clicking on the link.

```
≫ help
```

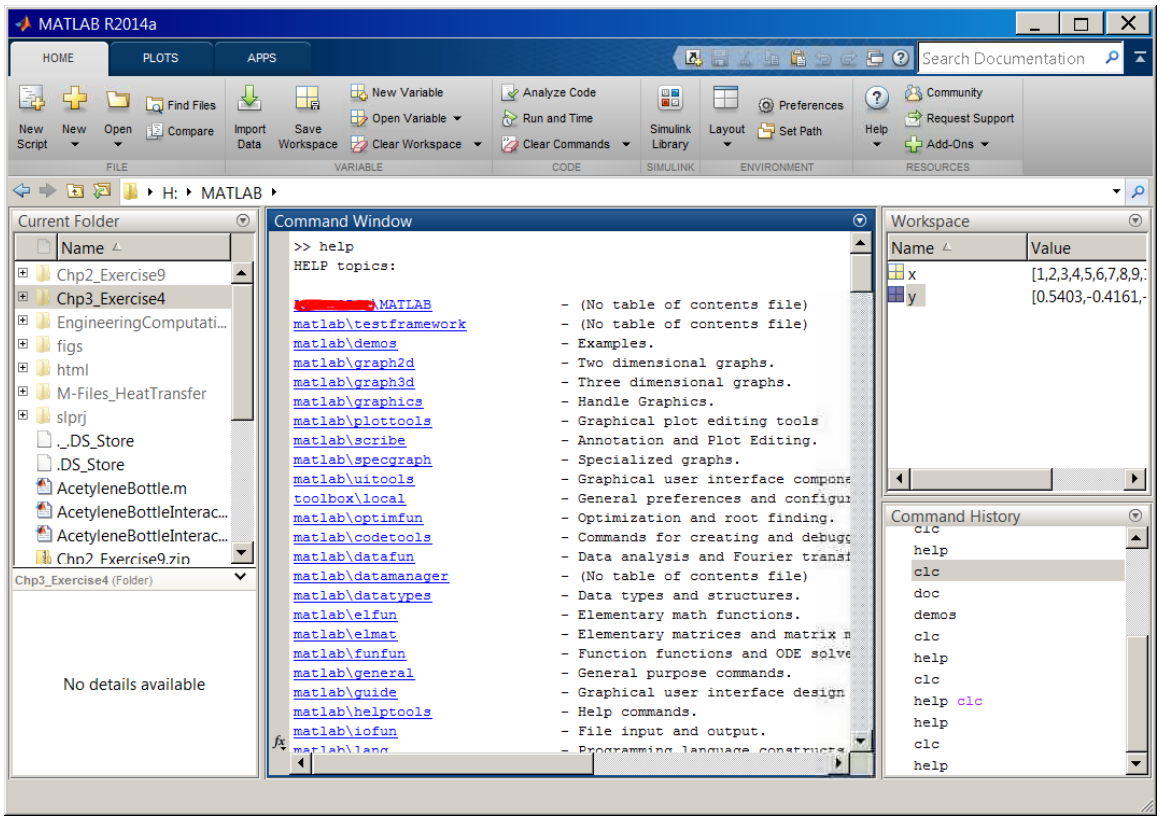


Figure 1.13: Help.

Or if you know the command or function you need help with, you can type help followed by the command or function. For example to learn about `clc` command, type `help clc` at the command prompt:

```
≫ help clc
```

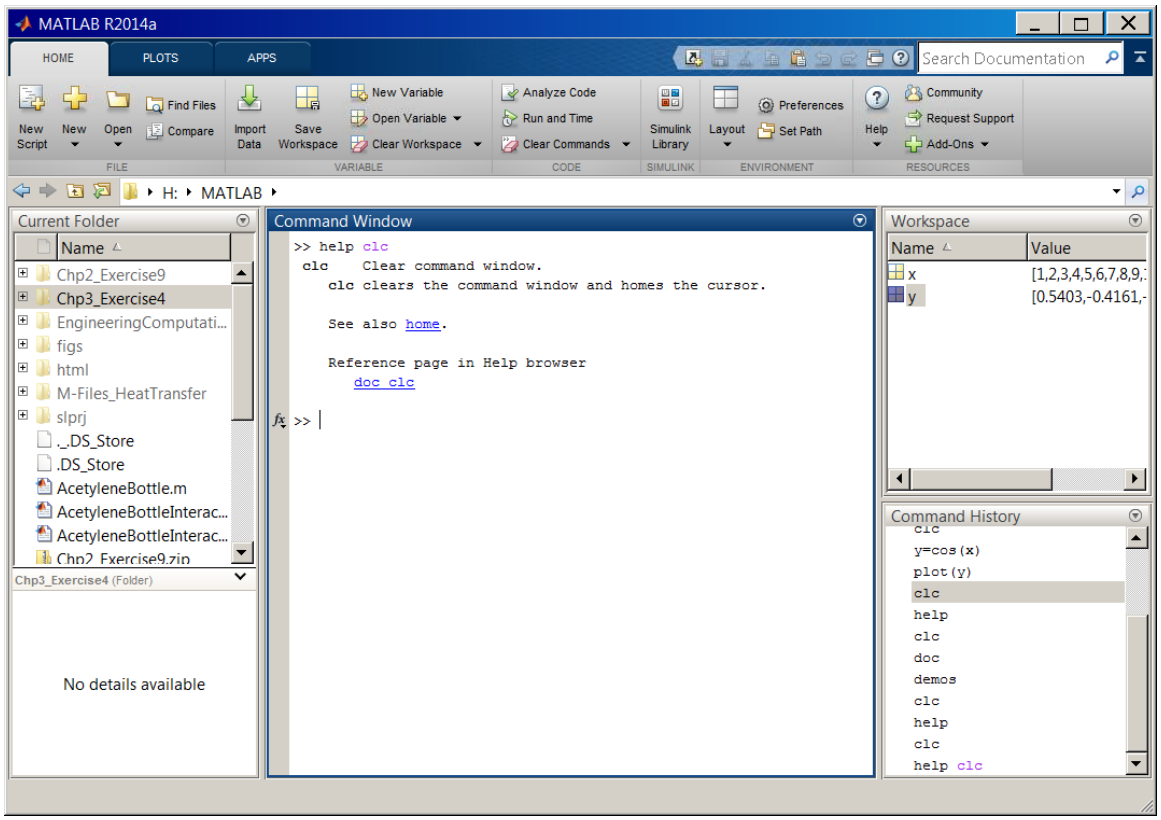


Figure 1.14: The output of `≫ help clc` command.

Also try the following command: `≫ help clear`

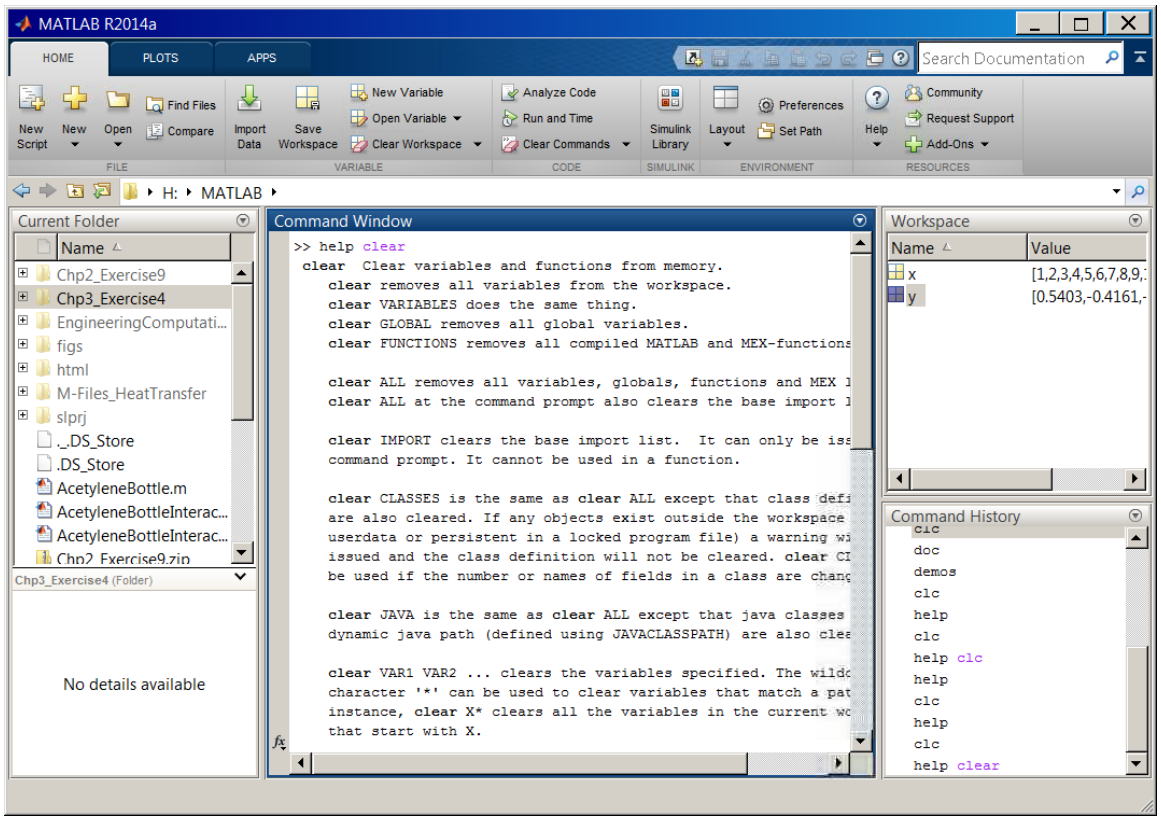


Figure 1.15: The output of ≫ `help clear` command.

To learn about sine function, type `help sin` at the command prompt:

≫ `help sin`

1.1.4.2 Doc

Obviously, to use `help` effectively, you need to know what you are looking for. Often times, especially when you first start learning an application, it is usually difficult to ask the right questions. In the case of MATLAB, `doc` command is generally better than `help`. If you type `doc` in the command prompt, MATLAB opens a browser from where you can obtain help easier:

≫ `doc`

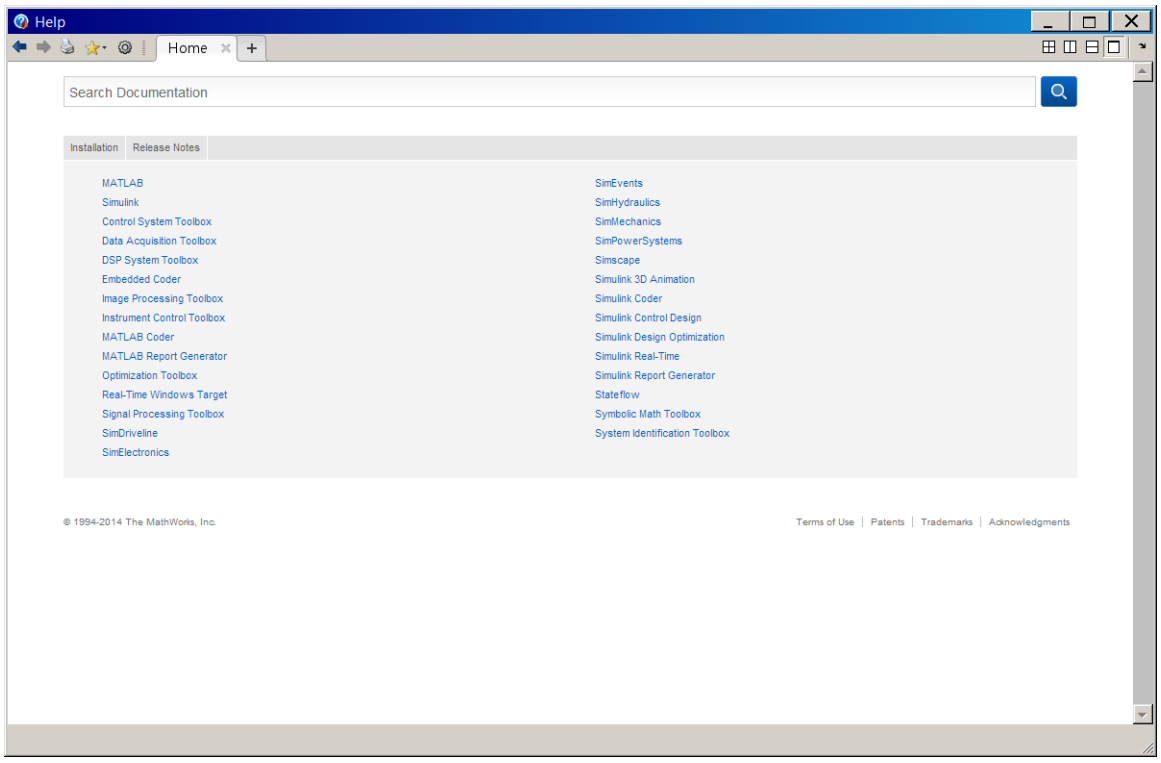


Figure 1.16: Built-in MATLAB Documentation.

Like using `help sin`, try typing `doc sin` in the command prompt:

```
≫ doc sin
```

1.1.4.3 Demos

You can learn more about MATLAB through demos by typing `demo` in the command prompt, a list of links to demos will open in Help Browser. Demos and online seminars are available at product demos and online seminars[8] .

```
≫ demo
```

[8] http://www.mathworks.com/products/matlab/demos.html

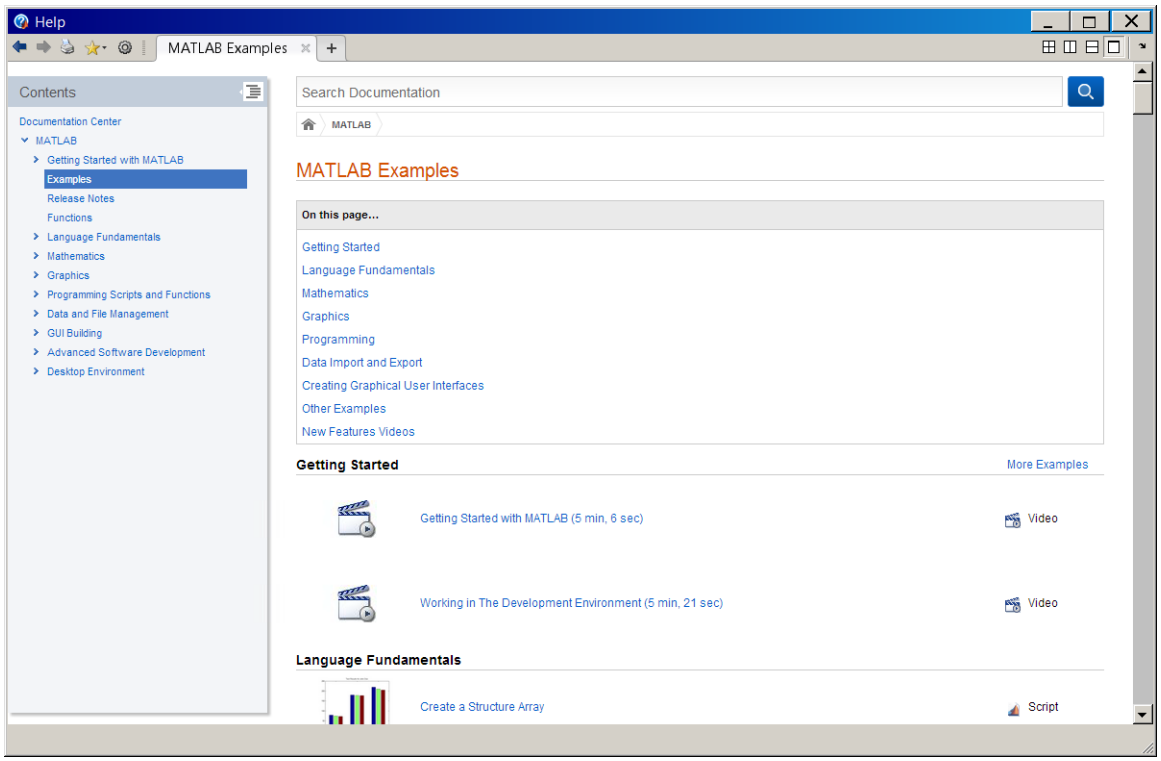


Figure 1.17: Built-in MATLAB Demos.

1.1.5 Useful Commands and Functions

For a detailed explanation and examples for each of the following type 'help function' (without quotes) at the MATLAB prompt.

Command/Function	Meaning
clc	Clear Command Window
clear	Remove items from workspace
who, whos	List variables in workspace
workspace	Display Workspace browser
cd	Change working directory
pwd	Display current directory
computer	Identify information about computer on which MATLAB is running
ver	Display version information for MathWorks products
quit	Terminate MATLAB
exit	Terminate MATLAB (same as quit)

Table 1.1: Useful commands and functions

1.1.6 Summary of Key Points

1. MATLAB is a popular technical computing application and MathWorks offers a trial version of MATLAB on their website,
2. The MATLAB Desktop consists of Command Window, Command History, Workspace, Current Folder and Start Button,
3. The up/down arrow keys, the tab key and the semicolon are convenient tools to use the Command Window,
4. MATLAB features an online help, doc and demo,
5. Various commands and functions make MATLAB experience easier, for example, `clc, clear and exit`.

1.2 Problem Set[9]

Exercise 1.2.1 *(Solution on p. 23.)*

Learn about the following terms using `help` command:

1. workspace
2. plot
3. clear
4. format
5. roots

[9] This content is available online at <http://cnx.org/content/m41463/1.3/>.

Solutions to Exercises in Chapter 1

Solution to Exercise 1.2.1 (p. 22)

1.

```
   ≫ help workspace
WORKSPACE Open Workspace browser to manage workspace
   WORKSPACE Opens the Workspace browser with a view of the variables
   in the current Workspace.  Displayed variables may be viewed,
   manipulated, saved, and cleared.

   See also whos, openvar, save.

   Reference page in Help browser
      doc workspace

≫
```

2.

```
   ≫ help plot
PLOT   Linear plot.
   PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
   then the vector is plotted versus the rows or columns of the matrix,
   whichever line up.  If X is a scalar and Y is a vector, disconnected
   line objects are created and plotted as discrete points vertically at
   X. ..........
```

3.

```
   ≫ help clear
CLEAR  Clear variables and functions from memory.
   CLEAR removes all variables from the workspace.
   CLEAR VARIABLES does the same thing.
   CLEAR GLOBAL removes all global variables.
   CLEAR FUNCTIONS removes all compiled M- and MEX-functions.

   CLEAR ALL removes all variables, globals, functions and MEX links.
   CLEAR ALL at the command prompt also removes the Java packages import
   list.
......
```

4.

```
   ≫ help format
FORMAT Set output format.
   FORMAT with no inputs sets the output format to the default appropriate
   for the class of the variable. For float variables, the default is
```

```
    FORMAT SHORT.
.....
```

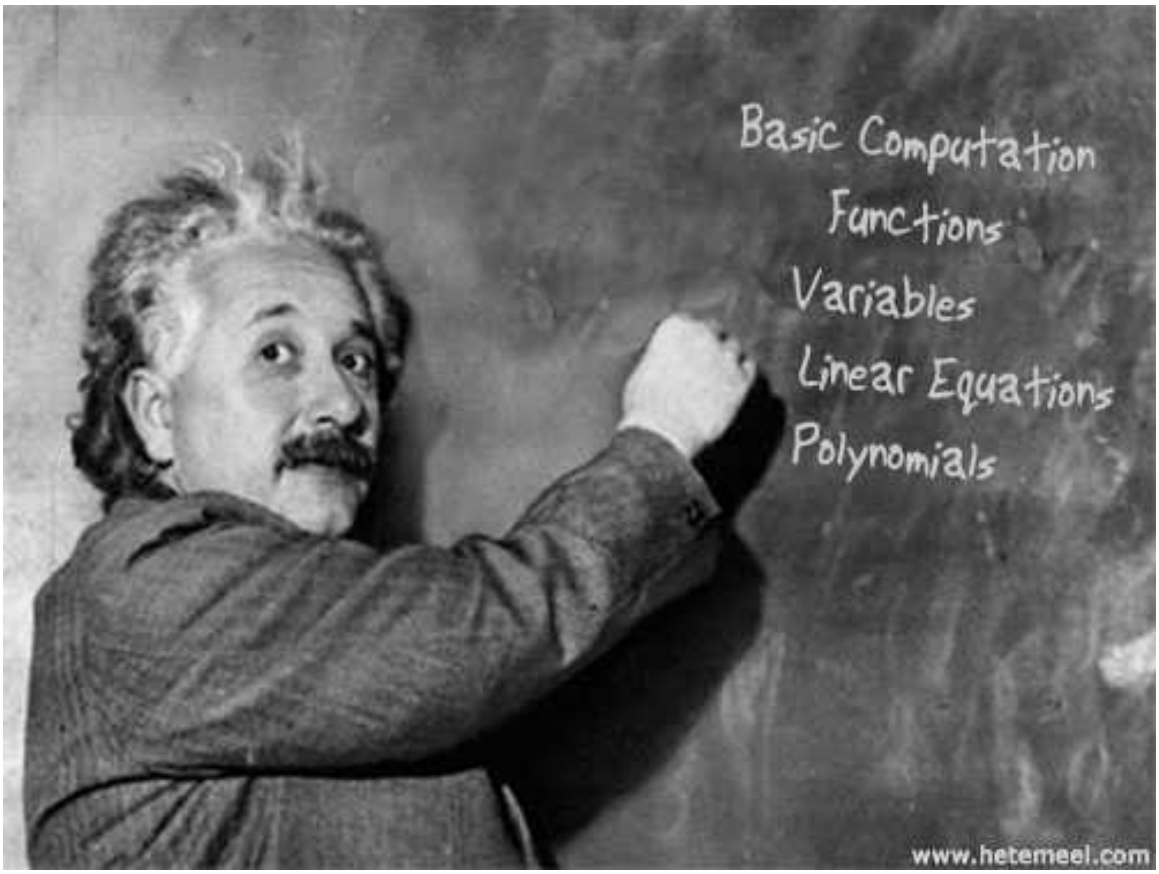
5.

```
    ≫ help roots
 ROOTS  Find polynomial roots.
    ROOTS(C) computes the roots of the polynomial whose coefficients
    are the elements of the vector C. If C has N+1 components,
    the polynomial is C(1)*X^N + ... + C(N)*X + C(N+1).
.....
```

Chapter 2

Getting Started

2.1 Essentials[1]



Learning a new skill, especially a computer program in this case, can be overwhelming. However, if we build on what we already know, the process can be handled rather effectively. In the preceding chapter we learned about MATLAB Graphical User Interface (GUI) and how to get help. Knowing the GUI, we will use basic math skills in MATLAB to solve linear equations and find roots of polynomials in this chapter.

[1] This content is available online at <http://cnx.org/content/m41409/1.3/>.

2.1.1 Basic Computation

2.1.1.1 Mathematical Operators

The evaluation of expressions is accomplished with arithmetic operators as we use them in scientific calculators. Note the addtional operators shown in the table below:

Operator	Name	Description
+	Plus	Addition
-	Minus	Subtraction
*	Asterisk	Multiplication
/	Forward Slash	Division
\	Back Slash	Left Matrix Division
^	Caret	Power
.*	Dot Asterisk	Array multiplication (element-wise)
./	Dot Slash	Right array divide (element-wise)
.\	Dot Back Slash	Left array divide (element-wise)
.^	Dot Caret	Array power (element-wise)

Table 2.1: Operators

NOTE: The backslash operator is used to solve linear systems of equations, see Section 2.1.5 (Linear Equations).

IMPORTANT: Matrix is a rectangular array of numbers and formed by rows and columns. For example $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$. In this example A consists of 4 rows and 4 columns and therefore is a 4x4 matrix. (see Wikipedia[2]).

IMPORTANT: Row vector is a special matrix that contains only one row. In other words, a row vector is a 1xn matrix where n is the number of elements in the row vector. $B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \end{pmatrix}$

IMPORTANT: Column vector is also a special matrix. As the term implies, it contains only one column. A column vector is an nx1 matrix where n is the number of elements in the column vector.

$$C = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

[2] http://en.wikipedia.org/wiki/Matrix_%28mathematics%29

NOTE: Array operations refer to element-wise calculations on the arrays, for example if x is an a by b matrix and y is a c by d matrix then x.*y can be performed only if a=c and b=d. Consider the following example, x consists of 2 rows and 3 columns and therefore it is a 2x3 matrix. Likewise, y has 2 rows and 3 columns and an array operation is possible. $x = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ and $y = \begin{pmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \end{pmatrix}$ then $x.*y = \begin{pmatrix} 10 & 40 & 90 \\ 160 & 250 & 360 \end{pmatrix}$

Example 2.1
The following figure illustrates a typical calculation in the Command Window.

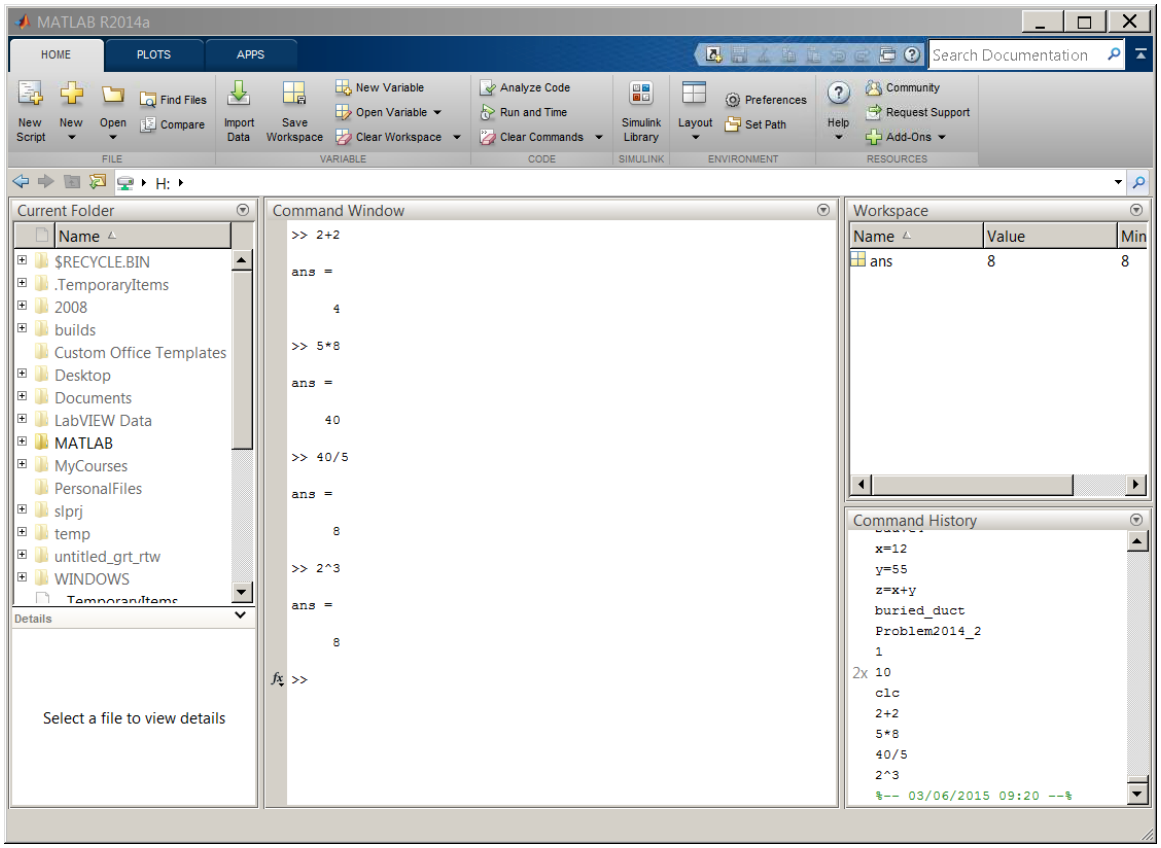


Figure 2.1: Basic arithmetic in the command window.

2.1.1.2 Operator Precedence

MATLAB allows us to build mathematical expressions with any combination of arithmetic operators. The order of operations are set by precedence levels in which MATLAB evaluates an expression from left to right.

The precedence rules for MATLAB operators are shown in the list below from the highest precedence level to the lowest.

1. Parentheses ()
2. Power (^)
3. Multiplication (*), right division (/), left division (\)
4. Addition (+), subtraction (-)

2.1.2 Mathematical Functions

MATLAB has all of the usual mathematical functions found on a scientific calculator including square root, logarithm, and sine.

IMPORTANT: Typing `pi` returns the number 3.1416. To find the sine of pi, type in `sin(pi)` and press enter.

IMPORTANT: The arguments in trigonometric functions are in radians. Multiply degrees by pi/180 to get radians. For example, to calculate sin(90), type in `sin(90*pi/180)`.

WARNING: In MATLAB `log` returns the natural logarithm of the value. To find the ln of 10, type in log(10) and press enter, (ans = 2.3026).

WARNING: MATLAB accepts `log10` for common (base 10) logarithm. To find the log of 10, type in log10(10) and press enter, (ans = 1).

Practice the following examples to familiarize yourself with the common mathematical functions. Be sure to read the relevant `help` and `doc` pages for functions that are not self explanatory.

Example 2.2
Calculate the following quantities:

1. $\frac{2^3}{3^2-1}$,
2. $5^{0.5}-1$
3. $\frac{\pi}{4}d^2$ for d=2

MATLAB inputs and outputs are as follows:

1. $\frac{2^3}{3^2-1}$ is entered by typing `2^3/(3^2-1)` (ans = 1)
2. $5^{0.5}-1$ is entered by typing `sqrt(5)-1` (ans = 1.2361)
3. $\frac{\pi}{4}d^2$ for d=2 is entered by typing `pi/4*2^2` (ans = 3.1416)

Example 2.3
Calculate the following exponential and logarithmic quantities:

1. e^2
2. $\ln\left(5^{10}\right)$
3. $\log 10^5$

MATLAB inputs and outputs are as follows:

1. `exp(2)` (ans = 7.3891)

2. `log((5^10))` (ans = 16.0944)
3. `log10(10^5)` (ans = 5)

Example 2.4
Calculate the following trigonometric quantities:

1. $\cos\left(\frac{\pi}{6}\right)$
2. $\tan(45)$
3. $\sin(\pi) + \cos(45)$

MATLAB inputs and outputs are as follows:

1. `cos(pi/6)` (ans = 0.8660)
2. `tan(45*pi/180)` (ans = 1.0000)
3. `sin(pi)+cos(45*pi/180)` (ans = 0.7071)

2.1.3 The `format` Function

The `format` function is used to control how the numeric values are displayed in the Command Window. The `short` format is set by default and the numerical results are displayed with 4 digits after the decimal point (see the examples above). The `long` format produces 15 digits after the decimal point.

Example 2.5
Calculate $\theta = \tan\left(\frac{\pi}{3}\right)$ and display results in `short` and `long` formats.
The `short` format is set by default:

```
≫ theta=tan(pi/3)

theta =

    1.7321

≫
```

And the `long` format is turned on by typing `format long`:

```
≫ theta=tan(pi/3)

theta =

    1.7321

≫ format long
≫ theta

theta =

   1.732050807568877
```

2.1.4 Variables

In MATLAB, a named value is called a variable. MATLAB comes with several predefined variables. For example, the name pi refers to the mathematical quantity π, which is approximately `pi ans = 3.1416`

> WARNING: MATLAB is case-sensitive, which means it distinguishes between upper- and lowercase letters (e.g. data, DATA and DaTa are three different variables). Command and function names are also case-sensitive. Please note that when you use the command-line help, function names are given in upper-case letters (e.g., CLEAR) only to emphasize them. Do not use upper-case letters when running functions and commands.

2.1.4.1 Declaring Variables

Variables in MATLAB are generally represented as matrix quantities. Scalars and vectors are special cases of matrices having size 1x1 (scalar), 1xn (row vector) or nx1 (column vector).

2.1.4.1.1 Declaration of a Scalar

The term scalar as used in linear algebra refers to a real number. Assignment of scalars in MATLAB is easy, type in the variable name followed by = symbol and a number:

Example 2.6

```
a = 1
```

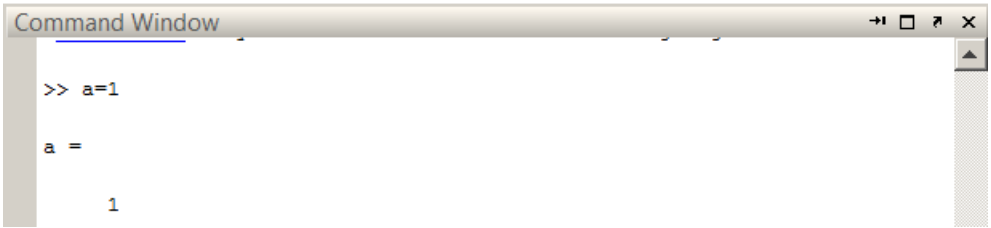


Figure 2.2: Assignment of a scalar quantity.

2.1.4.1.2 Declaration of a Row Vector

Elements of a row vector are separated with blanks or commas.

Example 2.7

Let's type the following at the command prompt:

```
b = [1 2 3 4 5]
```

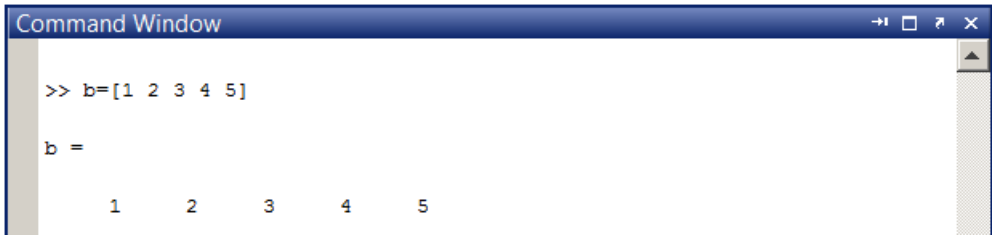


Figure 2.3: Assignment of a row vector quantity.

We can also use the New Variable button to assign a row vector. In the tool strip, select Home > New Variable. This action will create a variable called unnamed which is displayed in the workspace. By clicking on the title unnamed, we can rename it to something more descriptive. By double-clicking on the variable, we can open the Variable Editor and type in the values into spreadsheet looking table.

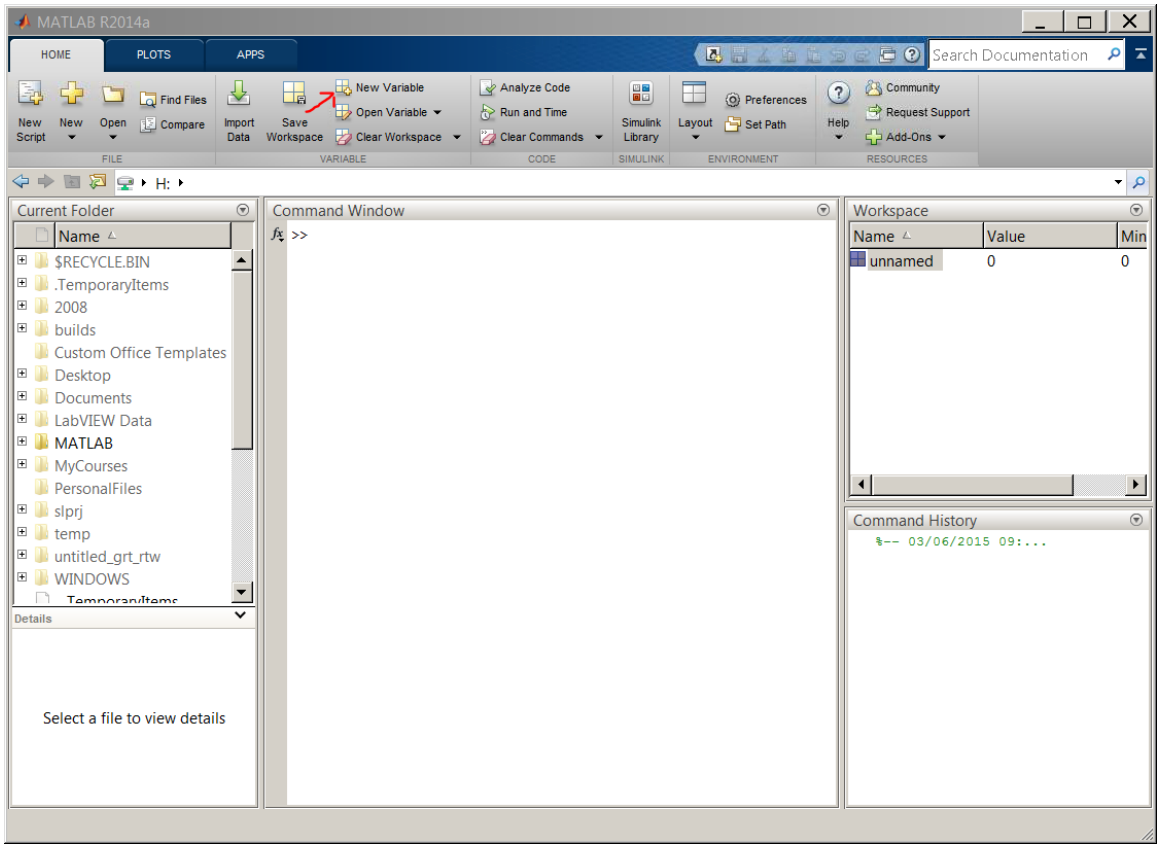


Figure 2.4: Using the New Variable button in the tool strip.

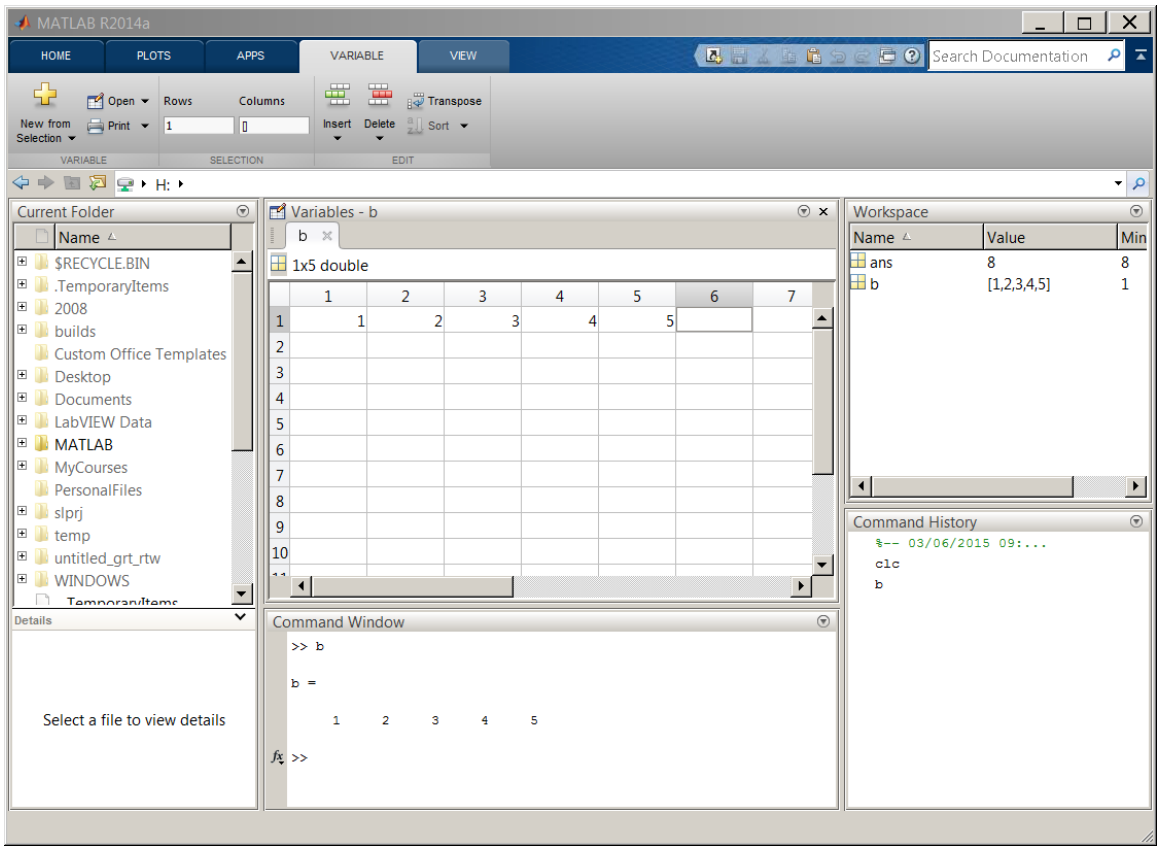


Figure 2.5: Assignment of a row vector by using the Variable Editor.

2.1.4.1.3 Declaration of a Column Vector

Elements of a column vector is ended by a semicolon:

Example 2.8

```
c = [1;2;3;4;5;]
```

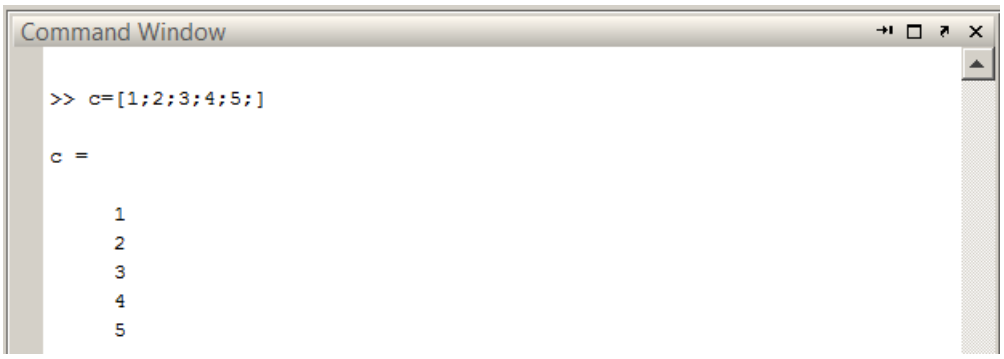


Figure 2.6: Assignment of a column vector quantity.

Or by transposing a row vector with the ' operator:
`c = [1 2 3 4 5]'`

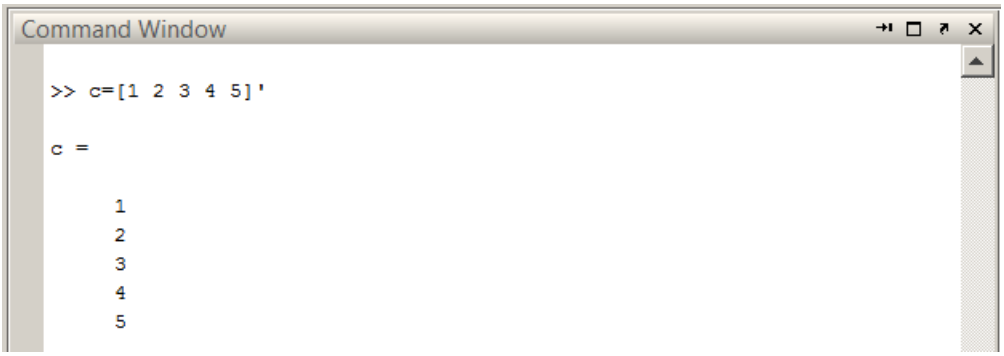


Figure 2.7: Assignment of a column vector quantity by transposing a row vector with the ' operator.

Or by using the Variable Editor:

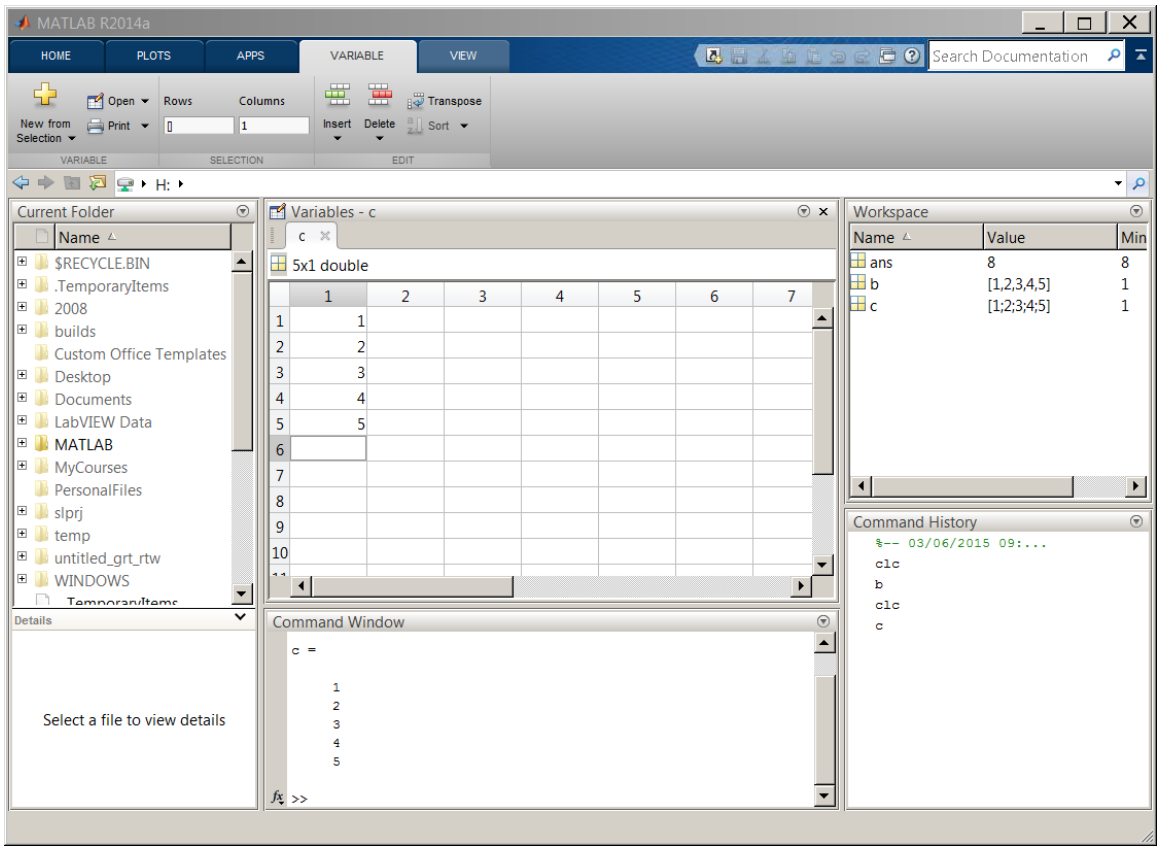


Figure 2.8: Assignment of a column vector quantity by using the Variable Editor.

2.1.4.1.4 Declaration of a Matrix

Matrices are typed in rows first and separated by semicolons to create columns. Consider the examples below:

Example 2.9
Let us type in a 2x5 matrix:

```
d = [2 4 6 8 10; 1 3 5 7 9]
```

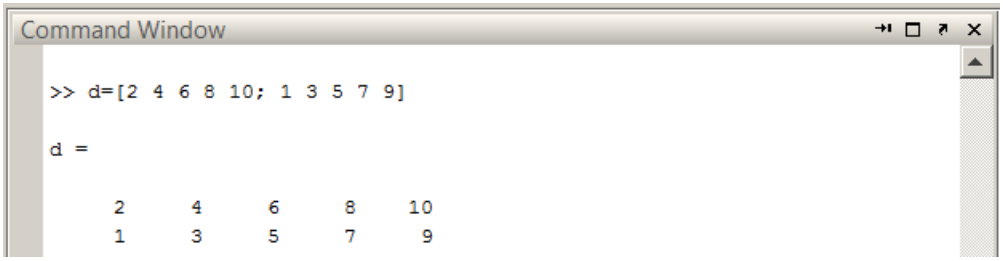


Figure 2.9: Assignment of a 2x5 matrix.

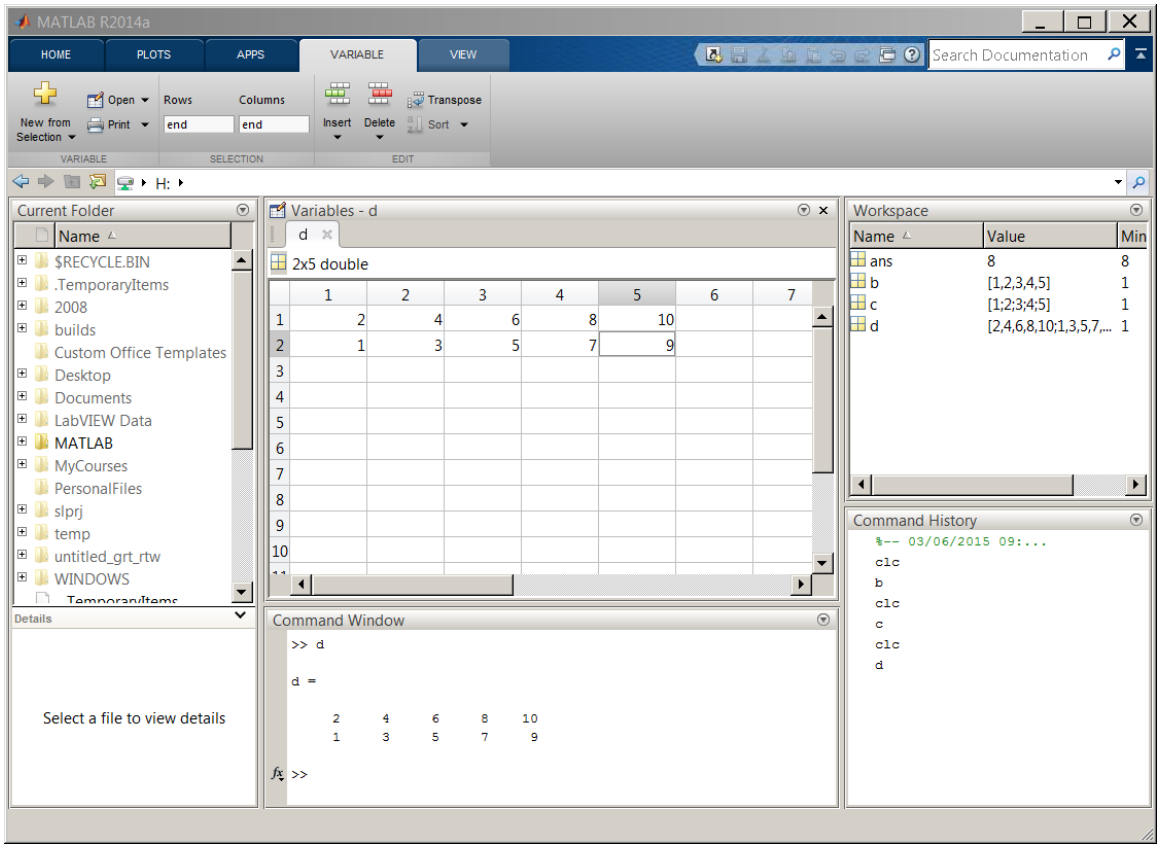


Figure 2.10: Assignment of a matrix by using the Variable Editor.

Example 2.10
This example is a 5x2 matrix:

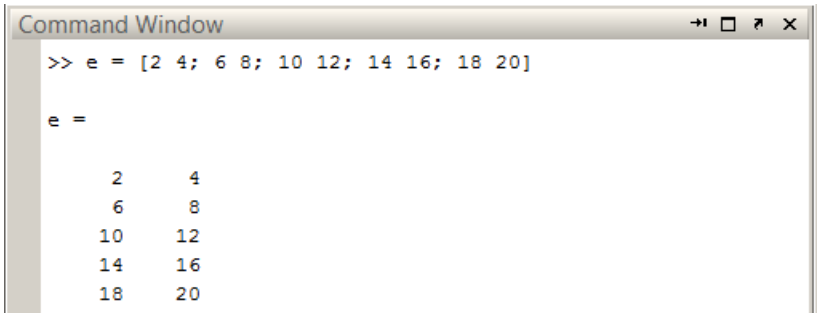


Figure 2.11: Assignment of a 5x2 matrix.

2.1.5 Linear Equations

Systems of linear equations are very important in engineering studies. In the course of solving a problem, we often reduce the problem to simultaneous equations from which the results are obtained. As you learned earlier, MATLAB stands for Matrix Laboratory and has features to handle matrices. Using the coefficients of simultaneous linear equations, a matrix can be formed to solve a set of simultaneous equations.

Example 2.11
Let's solve the following simultaneous equations:

$$x + y = 1 \tag{2.1}$$

$$2x - 5y = 9 \tag{2.2}$$

First, we will create a matrix for the left-hand side of the equation using the coefficients, namely 1 and 1 for the first and 2 and -5 for the second. The matrix looks like this:

$$\begin{pmatrix} 1 & 1 \\ 2 & -5 \end{pmatrix} \tag{2.3}$$

The above matrix can be entered in the command window by typing `A=[1 1; 2 -5]`.

Second, we create a column vector to represent the right-hand side of the equation as follows:

$$\begin{pmatrix} 1 \\ 9 \end{pmatrix} \tag{2.4}$$

The above column vector can be entered in the command window by typing `B= [1;9]`.

To solve the simultaneous equation, we will use the left division operator and issue the following command: `C=A\B`. These three steps are illustrated below:

```
≫ A=[1 1; 2 -5]

A =

     1     1
```

```
     2    -5

≫ B= [1;9]

B =

     1
     9

≫ C=A\B

C =

     2
    -1

≫
```

The result `C` indicating 2 and 1 are the values for `x` and `y`, respectively.

2.1.6 Polynomials

In the preceding section, we briefly learned about how to use MATLAB to solve linear equations. Equally important in engineering problem solving is the application of polynomials. Polynomials are functions that are built by simply adding together (or subtracting) some power functions. (see Wikipedia[3]).

$$ax^2 + bx + c = 0 \tag{2.5}$$

$$\mathrm{f}\,(\mathrm{x}) = ax^2 + bx + c \tag{2.6}$$

The coeffcients of a polynominal are entered as a row vector beginning with the highest power and including the ones that are equal to 0.

Example 2.12
Create a row vector for the following function: $y = 2x^4 + 3x^3 + 5x^2 + x + 10$

Notice that in this example we have 5 terms in the function and therefore the row vector will contain 5 elements. `p=[2 3 5 1 10]`

Example 2.13
Create a row vector for the following function: $y = 3x^4 + 4x^2 - 5$

In this example, coefficients for the terms involving power of 3 and 1 are 0. The row vector still contains 5 elements as in the previous example but this time we will enter two zeros for the coefficients with power of 3 and 1: `p=[3 0 4 0 -5]`.

2.1.6.1 The `polyval` Function

We can evaluate a polynomial `p` for a given value of `x` using the syntax `polyval(p,x)` where p contains the coefficients of polynomial and x is the given number.

[3] http://en.wikipedia.org/wiki/Polynomial

Example 2.14
Evaluate f(x) at 5.

$$\mathrm{f}\,(\mathrm{x}) = 3x^2 + 2x + 1 \tag{2.7}$$

The row vector representing f(x) above is `p=[3 2 1]`. To evaluate f(x) at 5, we type in: `polyval(p,5)`. The following shows the Command Window output:

```
≫ p=[3 2 1]

p =

     3     2     1

≫ polyval(p,5)

ans =

    86

≫
```

2.1.6.2 The `roots` Function

Consider the following equation:

$$ax^2 + bx + c = 0 \tag{2.8}$$

Probably you have solved this type of equations numerous times. In MATLAB, we can use the `roots` function to find the roots very easily.

Example 2.15
Find the roots for the following:

$$0.6x^2 + 0.3x - 0.9 = 0 \tag{2.9}$$

To find the roots, first we enter the coefficients of polynomial in to a row vector p with `p=[0.6 0.3 -0.9]` and issue the `r=roots(p)` command. The following shows the command window output:

```
≫ p=[0.6 0.3 -0.9]

p =

    0.6000    0.3000   -0.9000

≫ r=roots(p)

r =

   -1.5000
    1.0000

≫
```

2.1.7 Splitting a Statement

You will soon find out that typing long statements in the Command Window or in the the Text Editor makes it very hard to read and maintain your code. To split a long statement over multiple lines simply enter three periods "..." at the end of the line and carry on with your statement on the next line.

Example 2.16
The following command window output illustrates the use of three periods:

```
≫ sin(pi)+cos(45*pi/180)-sin(pi/2)+cos(45*pi/180)+tan(pi/3)

ans =

    2.1463

≫ sin(pi)+cos(45*pi/180)-sin(pi/2)...
+cos(45*pi/180)+tan(pi/3)

ans =

    2.1463

≫
```

2.1.8 Comments

Comments are used to make scripts more "readable". The percent symbol % separates the comments from the code. Examine the following examples:

Example 2.17
The long statements are split to make it easier to read. However, despite the use of descriptive variable names, it is hard to understand what this script does, see the following Command Window output:

```
    t_water=80;
t_outside=15;
inner_dia=0.05;
thickness=0.006;
Lambda_steel=48;
AlfaInside=2800;
AlfaOutside=17;
thickness_insulation=0.012;
Lambda_insulation=0.03;

r_i=inner_dia/2
r_o=r_i+thickness
r_i_insulation=r_o
r_o_insulation=r_i_insulation+thickness_insulation
AreaInside=2*pi*r_i
AreaOutside=2*pi*r_o
AreaOutside_insulated=2*pi*r_o_insulation
AreaM_pipe=(2*pi*(r_o-r_i))/log(r_o/r_i)
```

```
AreaM_insulation=(2*pi*(r_o_insulation-r_i_insulation)) ...
    /log(r_o_insulation/r_i_insulation)
TotalResistance=(1/(AlfaInside*AreaInside))+ ...
    (thickness/(Lambda_steel*AreaM_pipe))+(1/(AlfaOutside*AreaOutside))
TotalResistance_insulated=(1/(AlfaInside*AreaInside))+ ...
    (thickness/(Lambda_steel*AreaM_pipe))+(thickness_insulation ...
    /(Lambda_insulation*AreaM_insulation))+(1/(AlfaOutside*AreaOutside_insulated))
Q_dot=(t_water-t_outside)/(TotalResistance*1000)
Q_dot_insulated=(t_water-t_outside)/(TotalResistance_insulated*1000)
PercentageReducttion=((Q_dot-Q_dot_insulated)/Q_dot)*100
```

Example 2.18

The following is an edited version of the above including numerous comments:

```
    % Problem 16.06
% Problem Statement
% Calculate the percentage reduction in heat loss when a layer of hair felt
% is wrapped around the outside surface (see problem 16.05)

format short

% Input Values
t_water=80;         % Water temperature [C]
t_outside=15;       % Atmospheric temperature [C]
inner_dia=0.05;     % Inner diameter [m]
thickness=0.006;    % [m]
Lambda_steel=48;    % Thermal conductivity of steel [W/mK]
AlfaInside=2800;    % Heat transfer coefficient of inside [W/m2K]
AlfaOutside=17;     % Heat transfer coefficient of outside [W/m2K]
% Neglect radiation
% Additional layer
thickness_insulation=0.012;     % [m]
Lambda_insulation=0.03;         % Thermal conductivity of insulation [W/mK]


% Output Values
% Q_dot=(t_water-t_outside)/TotalResistance
% TotalResistance=(1/(AlfaInside*AreaInside))+(thickness/(Lambda_steel*AreaM))+ ...
(1/(AlfaOutside*AreaOutside)
% Calculating the unknown terms
r_i=inner_dia/2                                         % Inner radius of pipe [m]
r_o=r_i+thickness                                       % Outer radius of pipe [m]
r_i_insulation=r_o                                      % Inner radius of insulation [m]
r_o_insulation=r_i_insulation+thickness_insulation      % Outer radius of pipe [m]
AreaInside=2*pi*r_i
AreaOutside=2*pi*r_o
AreaOutside_insulated=2*pi*r_o_insulation
AreaM_pipe=(2*pi*(r_o-r_i))/log(r_o/r_i)                % Logarithmic mean area for pipe
AreaM_insulation=(2*pi*(r_o_insulation-r_i_insulation)) ...
    /log(r_o_insulation/r_i_insulation)    % Logarithmic mean area for insulation
TotalResistance=(1/(AlfaInside*AreaInside))+(thickness/ ...
    (Lambda_steel*AreaM_pipe))+(1/(AlfaOutside*AreaOutside))
```

```
TotalResistance_insulated=(1/(AlfaInside*AreaInside))+(thickness/ ...
    (Lambda_steel*AreaM_pipe))+(thickness_insulation/(Lambda_insulation*AreaM_insulation)) ...
    +(1/(AlfaOutside*AreaOutside_insulated))
Q_dot=(t_water-t_outside)/(TotalResistance*1000) % converting into kW
Q_dot_insulated=(t_water-t_outside)/(TotalResistance_insulated*1000) % converting into kW
PercentageReducttion=((Q_dot-Q_dot_insulated)/Q_dot)*100
```

2.1.9 Basic Operations

Command	Meaning
sum	Sum of array elements
prod	Product of array elements
sqrt	Square root
log10	Common logarithm (base 10)
log	Natural logarithm
max	Maximum elements of array
min	Minimum elements of array
mean	Average or mean value of arrays
std	Standard deviation

Table 2.2: Basic operations.

2.1.10 Special Characters

Character	Meaning
=	Assignment
( )	Prioritize operations
[ ]	Construct array
:	Specify range of array elements
,	Row element separator in an array
;	Column element separator in an array
...	Continue statement to next line
.	Decimal point, or structure field separator
%	Insert comment line into code

Table 2.3: Special Characters

2.1.11 Summary of Key Points

1. MATLAB has the common functions found on a scientific calculator and can be operated in a similar way,
2. MATLAB can store values in variables. Variables are case sensitive and some variables are reserved by MATLAB (e.g. `pi` stores 3.1416),
3. Variable Editor can be used to enter or manipulate matrices,
4. The coefficients of simultaneous linear equations and polynomials are used to form a row vector. MATLAB then can be used to solve the equations,
5. The `format` function is used to control the number of digits displayed,
6. Three periods "..." at the end of the line is used to split a long statement over multiple lines,
7. The percent symbol % separates the comments from the code, anything following % symbol is ignored by MATLAB.

2.2 Problem Set[4]

Determine the value of each of the following.

Exercise 2.2.1 *(Solution on p. 44.)*
$6 \times 7 + 4^2 - 2^4$

Exercise 2.2.2 *(Solution on p. 44.)*
$\frac{3^2+2^3}{4^5-5^4} + \frac{64^{0.5}-5^2}{4^5+5^6+7^8}$

Exercise 2.2.3 *(Solution on p. 44.)*
$\log 10^2 + 10^5$

Exercise 2.2.4 *(Solution on p. 44.)*
$e^2 + 2^3 - \ln\left(e^2\right)$

Exercise 2.2.5 *(Solution on p. 44.)*
$\sin\left(2\pi\right) + \cos\left(\frac{\pi}{4}\right)$

Exercise 2.2.6 *(Solution on p. 44.)*
$\tan\left(\frac{\pi}{3}\right) + \cos\left(270\right) + \sin\left(270\right) + \cos\left(\frac{\pi}{3}\right)$

Exercise 2.2.7 *(Solution on p. 44.)*
Solve the following system of equations:
$2x + 4y = 1$
$x + 5y = 2$

Exercise 2.2.8 *(Solution on p. 44.)*
Evaluate y at 5.
$y = 4x^4 + 3x^2 - x$

Exercise 2.2.9 *(Solution on p. 45.)*
Given below is Load-Gage Length data for a type 304 stainless steel that underwent a tensile test. Original specimen diameter is 12.7 mm. [5]

[4] This content is available online at <http://cnx.org/content/m41464/1.7/>.
[5] *Introduction to Materials Science for Engineers* by J. F. Shackelford, Macmillan Publishing Company. ©1985, (p.304)

Load [N]	Gage Length [mm]
0.000	50.8000
4890	50.8102
9779	50.8203
14670	50.8305
19560	50.8406
24450	50.8508
27620	50.8610
29390	50.8711
32680	50.9016
33950	50.9270
34580	50.9524
35220	50.9778
35720	51.0032
40540	51.816
48390	53.340
59030	55.880
65870	58.420
69420	60.960
69670 (maximum)	61.468
68150	63.500
60810 (fracture)	66.040 (after fracture)

Table 2.4

$\sigma = \frac{P}{A}$, where P is the load [N] on the sample with an original cross-sectional area A [m^2] and the engineering strain is defined as $\epsilon = \frac{\Delta l}{l}$, where Δl is the change in length and l is the initial length.

Compute the stress and strain values for each of the measurements obtained in the tensile test. Data available for download.[6]

[6]See the file at <http://cnx.org/content/m41464/latest/Chp2_Exercise9.zip>

Solutions to Exercises in Chapter 2

Solution to Exercise 2.2.1 (p. 42)
≫ `(6*7)+4^2-2^4` (ans = 42)
Solution to Exercise 2.2.2 (p. 42)
≫ `((3^2+2^3)/(4^5-5^4))+((sqrt(64)-5^2)/(4^5+5^6+7^8))` (ans = 0.0426)
Solution to Exercise 2.2.3 (p. 42)
≫ `log10(10^2)+10^5` (ans = 100002)
Solution to Exercise 2.2.4 (p. 42)
≫ `exp(2)+2^3-log(exp(2))` (ans = 13.3891)
Solution to Exercise 2.2.5 (p. 42)
≫ `sin(2*pi)+cos(pi/4)` (ans = 0.7071)
Solution to Exercise 2.2.6 (p. 42)
≫ `tan(pi/3)+cos(270*pi/180)+sin(270*pi/180)+cos(pi/3)` (ans = 1.2321)
Solution to Exercise 2.2.7 (p. 42)

```
≫ A=[2 4; 1 5]

A =

     2     4
     1     5

≫ B=[1; 2]

B =

     1
     2

≫ Solution=A\B

Solution =

   -0.5000
    0.5000
```

Solution to Exercise 2.2.8 (p. 42)

```
≫ p=[4 0 3 -1 0]

p =

     4     0     3    -1     0

≫ polyval(p,5)

ans =
```

2570

≫

Solution to Exercise 2.2.9 (p. 42)

First, we need to enter the data sets. Because it is rather a large table, using Variable Editor is more convenient. See the figures below:

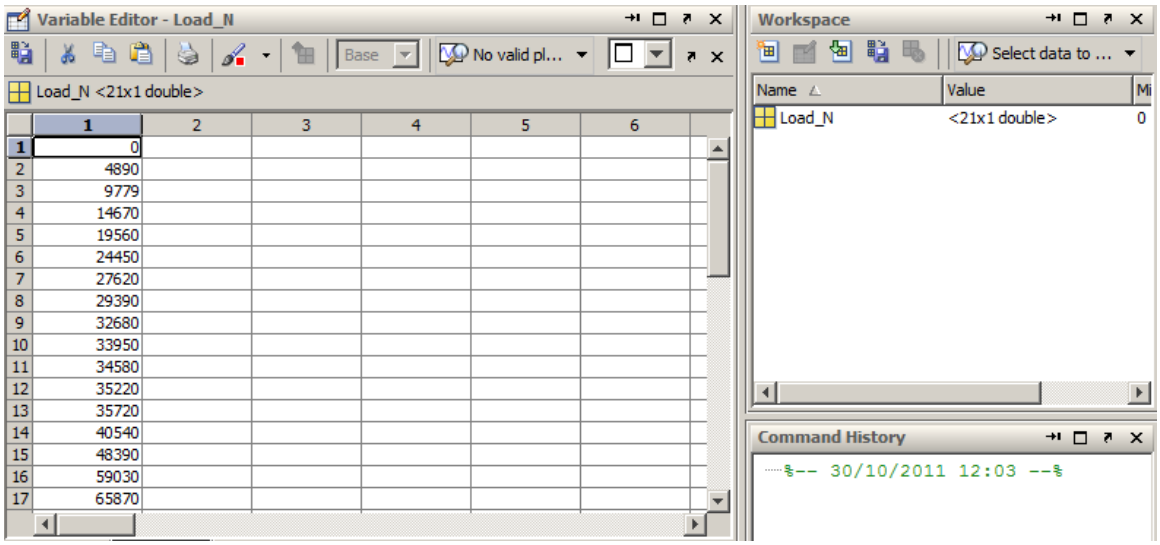
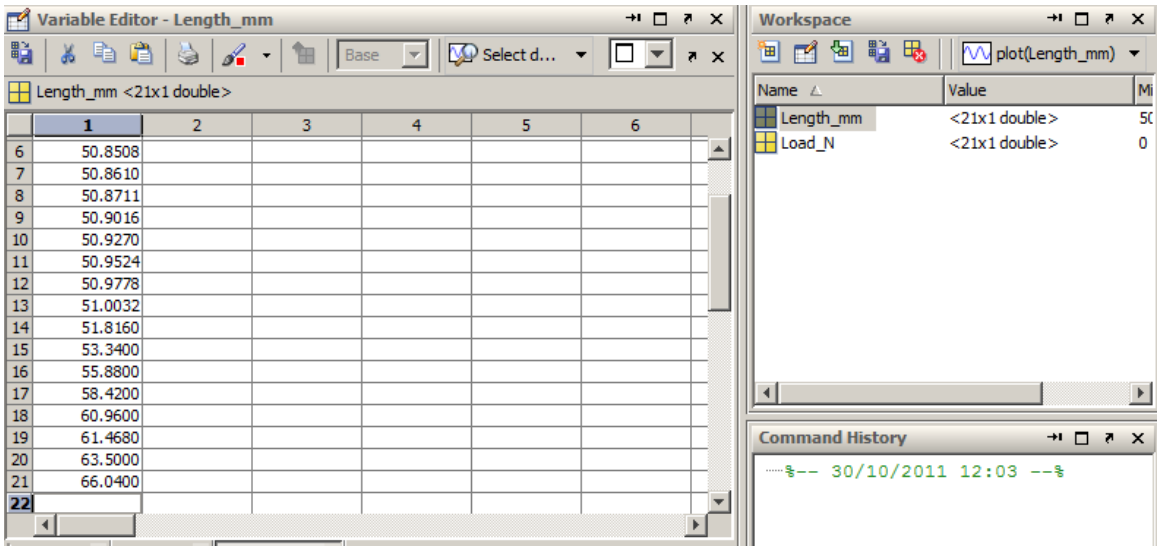


Figure 2.12: Load in Newtons



	1	2	3	4	5	6
6	50.8508					
7	50.8610					
8	50.8711					
9	50.9016					
10	50.9270					
11	50.9524					
12	50.9778					
13	51.0032					
14	51.8160					
15	53.3400					
16	55.8800					
17	58.4200					
18	60.9600					
19	61.4680					
20	63.5000					
21	66.0400					
22						

Figure 2.13: Extension length in mm.

Next, we will calculate the cross-sectional area.

```
Area=pi/4*(0.0127^2)

Area =

  1.2668e-004
```

Now, we can find the Stress values with the following, note that we are obtaining results in MPa:

```
Sigma=(Load_N./Area)*10^(-6)

Sigma =

         0
   38.6022
   77.1964
  115.8065
  154.4086
  193.0108
  218.0351
  232.0076
  257.9792
  268.0047
  272.9780
  278.0302
  281.9773
  320.0269
  381.9955
```

```
  465.9888
  519.9844
  548.0085
  549.9820
  537.9830
  480.0403
```

For strain calculation, we will first find the change in length:

```
Delta_L=Length_mm-50.800

Delta_L =

         0
    0.0102
    0.0203
    0.0305
    0.0406
    0.0508
    0.0610
    0.0711
    0.1016
    0.1270
    0.1524
    0.1778
    0.2032
    1.0160
    2.5400
    5.0800
    7.6200
   10.1600
   10.6680
   12.7000
   15.2400
```

Now we can determine Strain with the following:

```
Epsilon=Delta_L./50.800

Epsilon =

         0
    0.0002
    0.0004
    0.0006
    0.0008
    0.0010
    0.0012
    0.0014
    0.0020
    0.0025
```

```
    0.0030
    0.0035
    0.0040
    0.0200
    0.0500
    0.1000
    0.1500
    0.2000
    0.2100
    0.2500
    0.3000
```

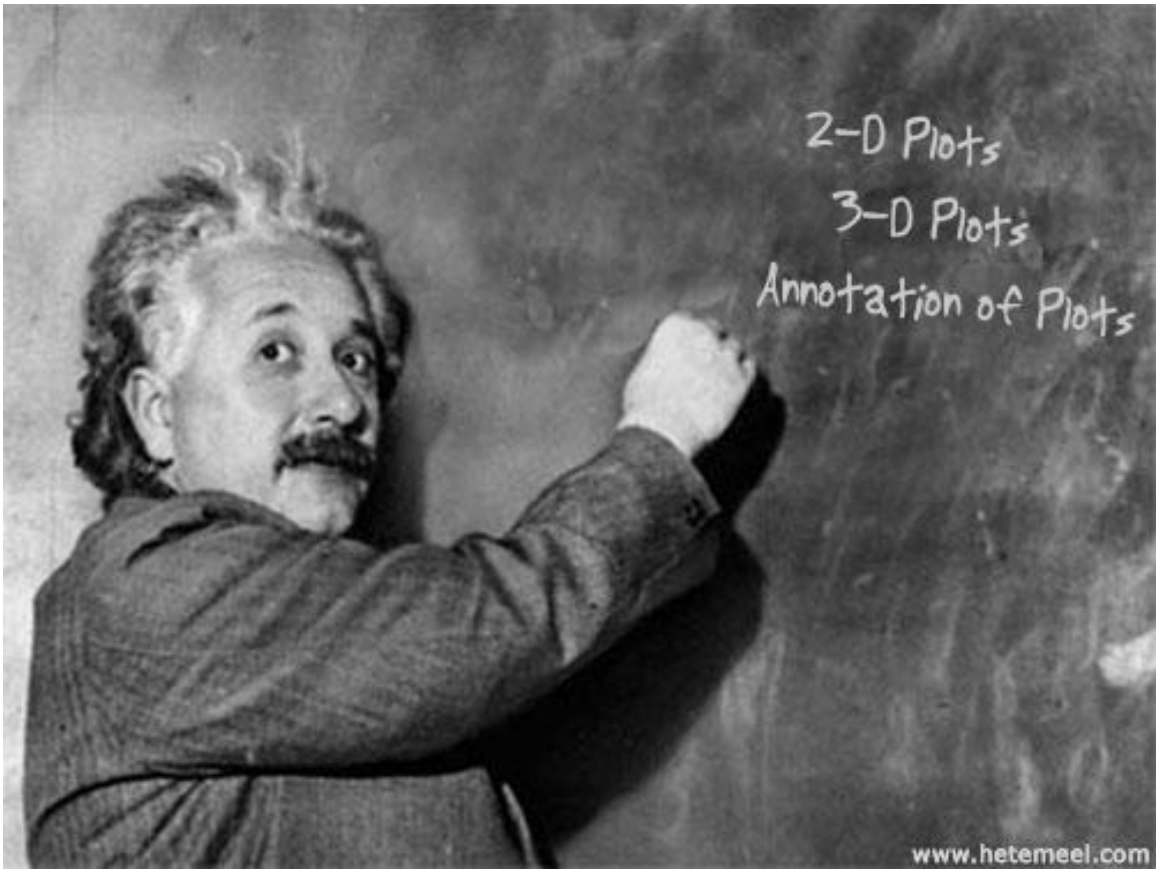
The final results can be tabulated as foolows:

```
[Sigma Epsilon]

ans =

         0         0
   38.6022    0.0002
   77.1964    0.0004
  115.8065    0.0006
  154.4086    0.0008
  193.0108    0.0010
  218.0351    0.0012
  232.0076    0.0014
  257.9792    0.0020
  268.0047    0.0025
  272.9780    0.0030
  278.0302    0.0035
  281.9773    0.0040
  320.0269    0.0200
  381.9955    0.0500
  465.9888    0.1000
  519.9844    0.1500
  548.0085    0.2000
  549.9820    0.2100
  537.9830    0.2500
  480.0403    0.3000
```

Chapter 3

Graphics

3.1 Plotting in MATLAB[1]



A picture is worth a thousand words, particularly visual representation of data in engineering is very useful. MATLAB has powerful graphics tools and there is a very helpful section devoted to graphics in MATLAB Help: Graphics. Students are encouraged to study that section; what follows is a brief summary of the main plotting features.

[1] This content is available online at <http://cnx.org/content/m41442/1.3/>.

3.1.1 Two-Dimensional Plots

3.1.1.1 The `plot` Statement

Probably the most common method for creating a plot is by issuing `plot(x, y)` statement where function y is plotted against x.

Example 3.1
Type in the following statement at the MATLAB prompt:

```
x=[-pi:.1:pi]; y=sin(x); plot(x,y);
```

After we executed the statement above, a plot named Figure1 is generated:

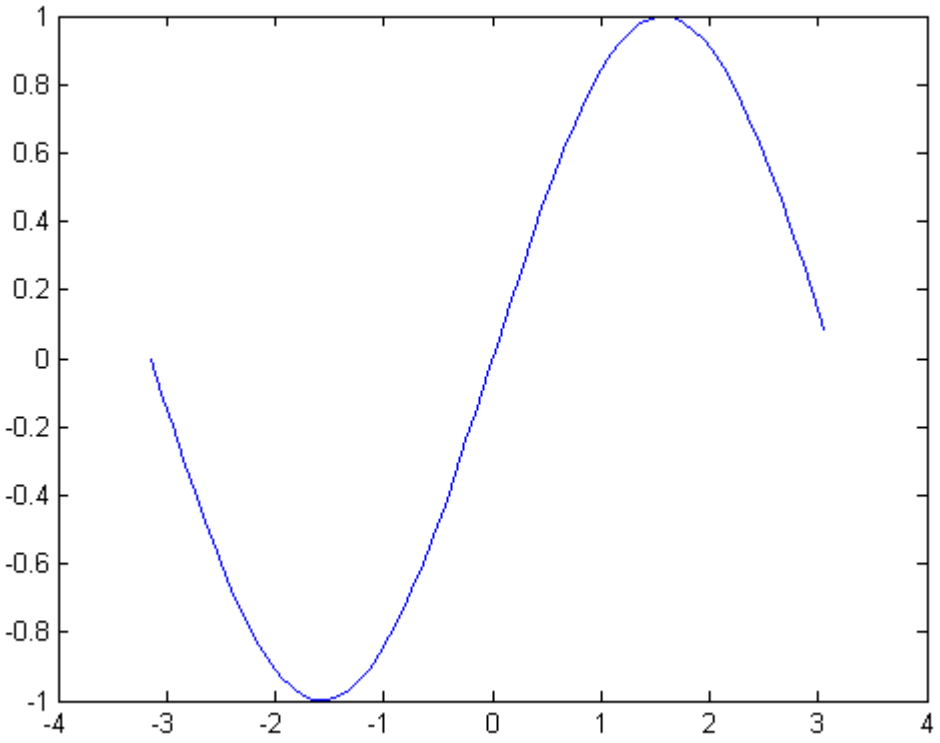


Figure 3.1: Graph of sin(x)

Having variables assigned in the Workspace, x and y=sin(x) in our case, we can also select x and y, and right click on the selected variables. This opens a menu from which we choose plot(x,y). See the figure below.

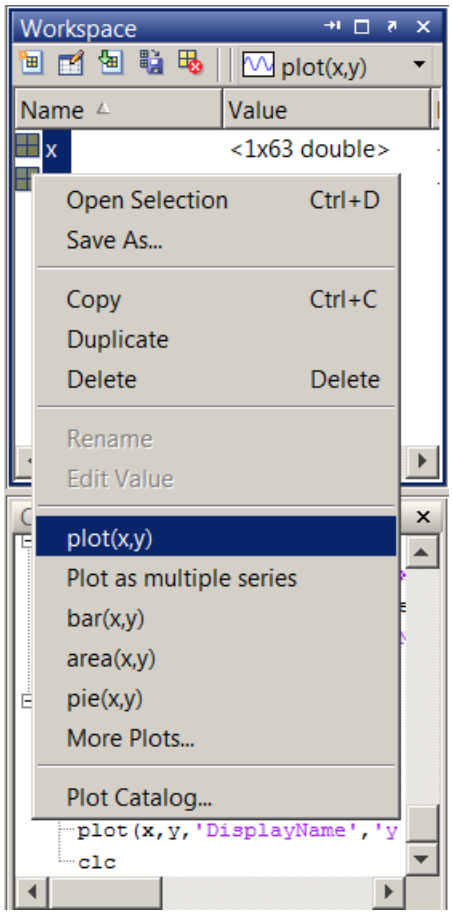


Figure 3.2: Creating a plot from Workspace.

3.1.1.2 Annotating Plots

Graphs without labels are incomplete and labeling elements such as plot title, labels for x and y axes, and legend should be included. Using up arrow, recall the statement above and add the annotation commands as shown below.

```
x=[-pi:.1:pi];y=sin(x);plot(x,y);title('Graph of y=sin(x)');xlabel('x');ylabel('sin(x)');grid on
```

Run the file and compare your result with the first one.

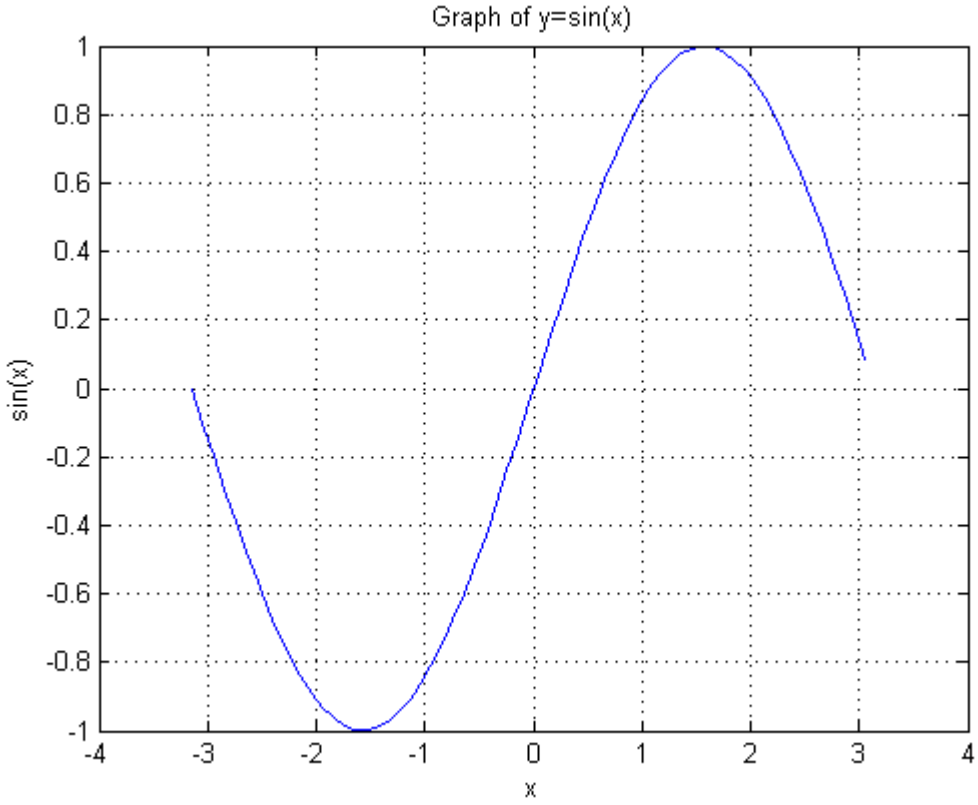


Figure 3.3: Graph of sin(x) with Labels.

ASIDE: Type in the following at the MATLAB prompt and learn additional commands to annotate plots:

```
help gtext
help legend
help zlabel
```

3.1.1.3 Superimposed Plots

If you want to merge data from two graphs, rather than create a new graph from scratch, you can superimpose the two using a simple trick:

```
    % This script generates sin(x) and cos(x) plot on the same graph
% initialize variables
x=[-pi:.1:pi];      %create a row vector from -pi to +pi with .1 increments
y0=sin(x);          %calculate sine value for each x
y1=cos(x);          %calculate cosine value for each x
```

```
% Plot sin(x) and cos(x) on the same graph
plot(x,y0,x,y1);
title('Graph of sin(x) and cos(x)'); %Title of graph
xlabel('x');                 %Label of x axis
ylabel('sin(x), cos(x)');    %Label of y axis
legend('sin(x)','cos(x)');  %Insert legend in the same order as y0 and y1 calculated
grid on                      %Graph grid is turned
```

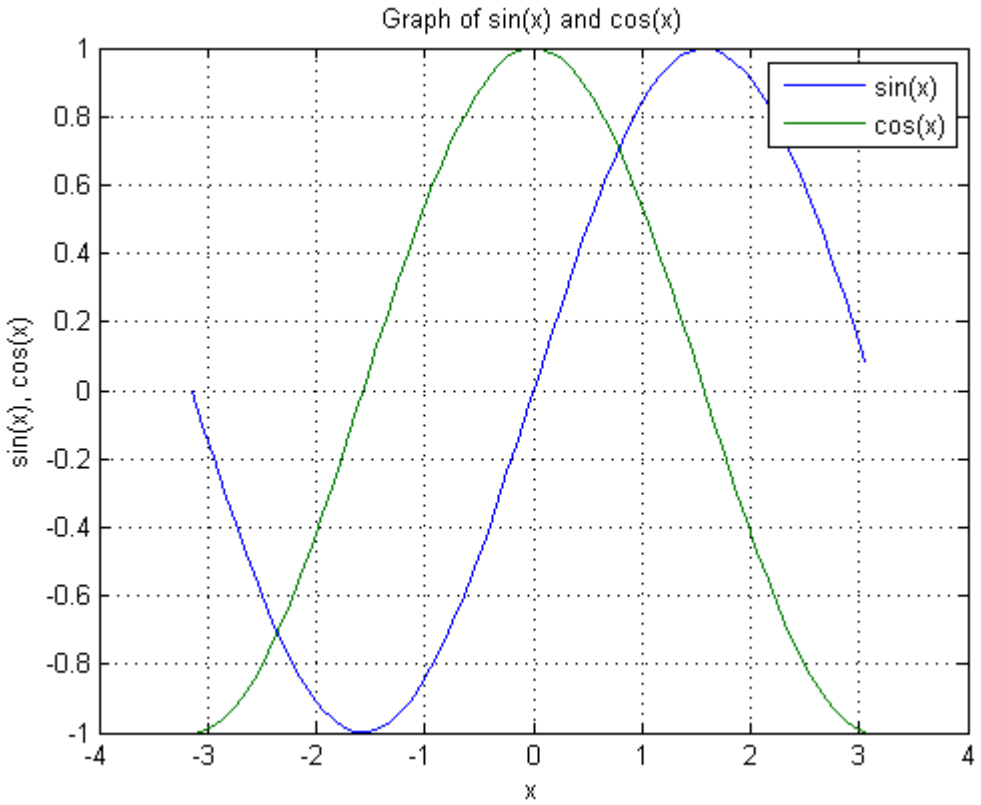


Figure 3.4: Graph of sin(x) and cos(x) in the same plot with labels and legend.

3.1.1.4 Multiple Plots in a Figure

Multiple plots in a single figure can be generated with `subplot` in the Command Window. However, this time we will use the built-in Plot Tools. Before we initialize that tool set, let us create the necessary variables using the following script:

```
    % This script generates sin(x) and cos(x) variables
clc                 %Clears command window
clear all           %Clears the variable space
close all           %Closes all figures
X1=[-2*pi:.1:2*pi];  %Creates a row vector from -2*pi to 2*pi with .1 increments
```

```
Y1=sin(X1);          %Calculates sine value for each x
Y2=cos(X1);          %Calculates cosine value for each x
Y3=Y1+Y2;            %Calculates sin(x)+cos(x)
Y4=Y1-Y2;            %Calculates sin(x)-cos(x)
```

Note that the above script clears the command window and variable workspace. It also closes any open Figures. After running the script, we will have X1, Y1, Y2, Y3 and Y4 loaded in the workspace. Next, select File > New > Figure, a new Figure window will open. Click "Show Plot Tools and Dock Figure" on the tool bar.

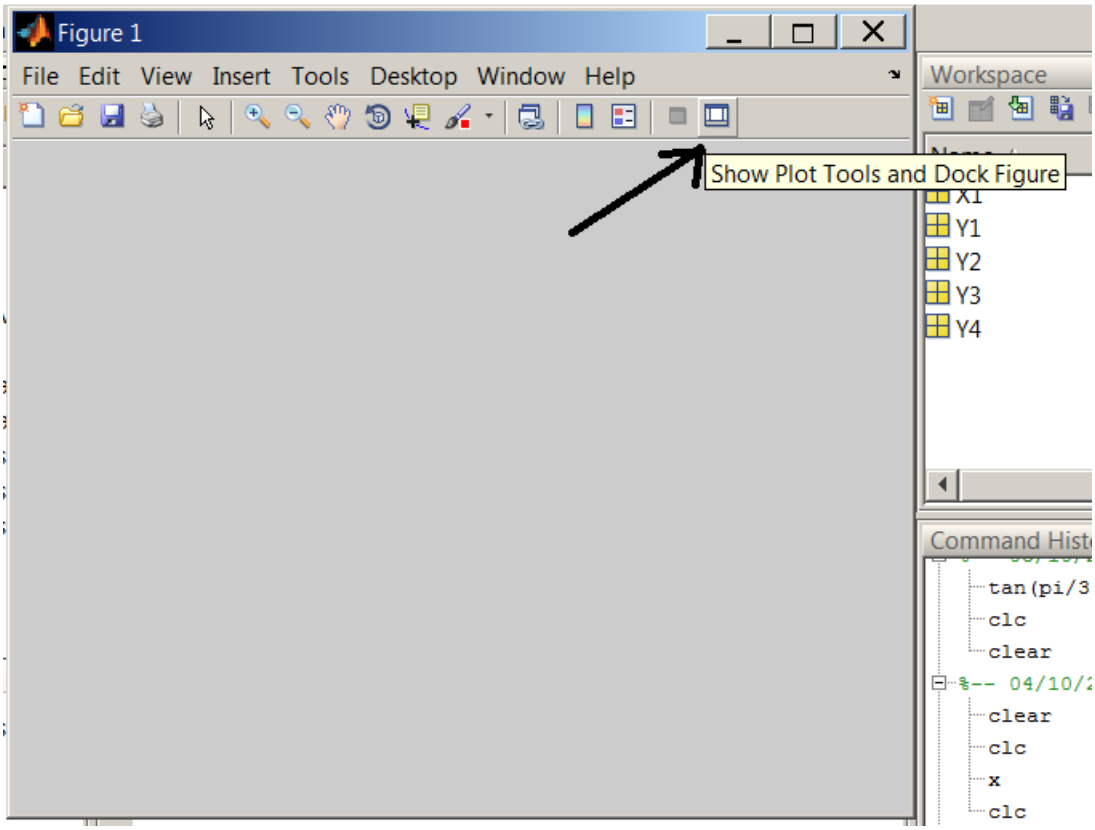


Figure 3.5: Plot Tools

Under New Subplots > 2D Axes, select four vertical boxes that will create four subplots in one figure. Also notice, the five variables we created earlier are listed under Variables.

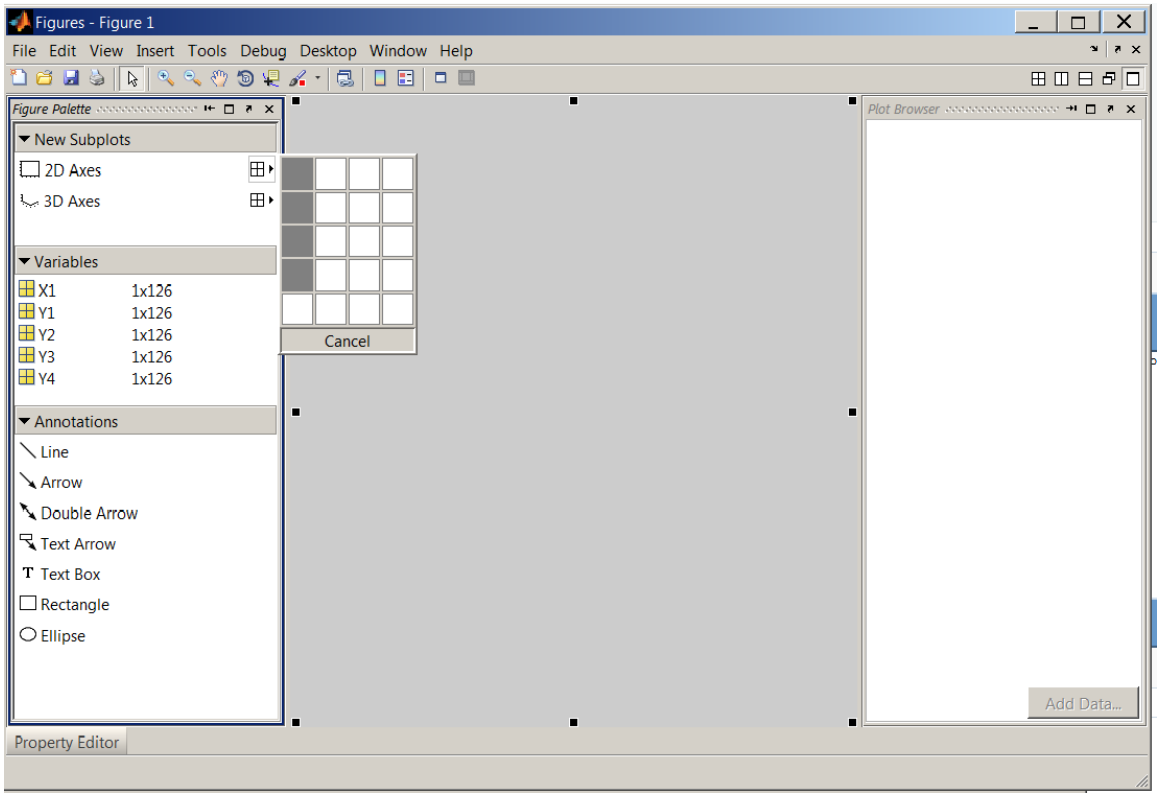


Figure 3.6: Creating four sub plots.

After the subplots have been created, select the first supblot and click on "Add Data". In the dialog box, set X Data Source to X1 and Y Data Source to Y1. Repeat this step for the remaining subplots paying attention to Y Data Source (Y2, Y3 and Y4 need to be selected in the subsequent steps while X1 is always the X Data Source).

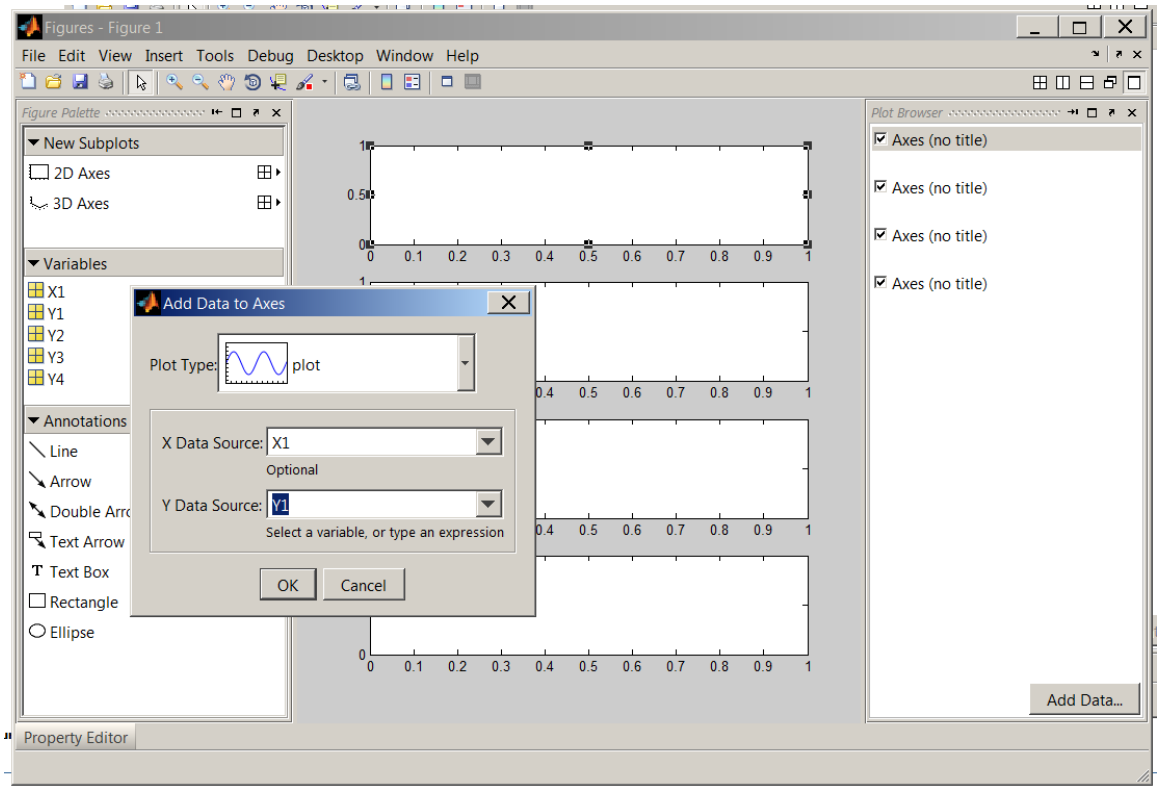


Figure 3.7: Adding data to axes.

Next, select the first item in "Plot Browser" and activate the "Property Editor". Fill out the fields as shown in the figure below. Repeat this step for all subplots.

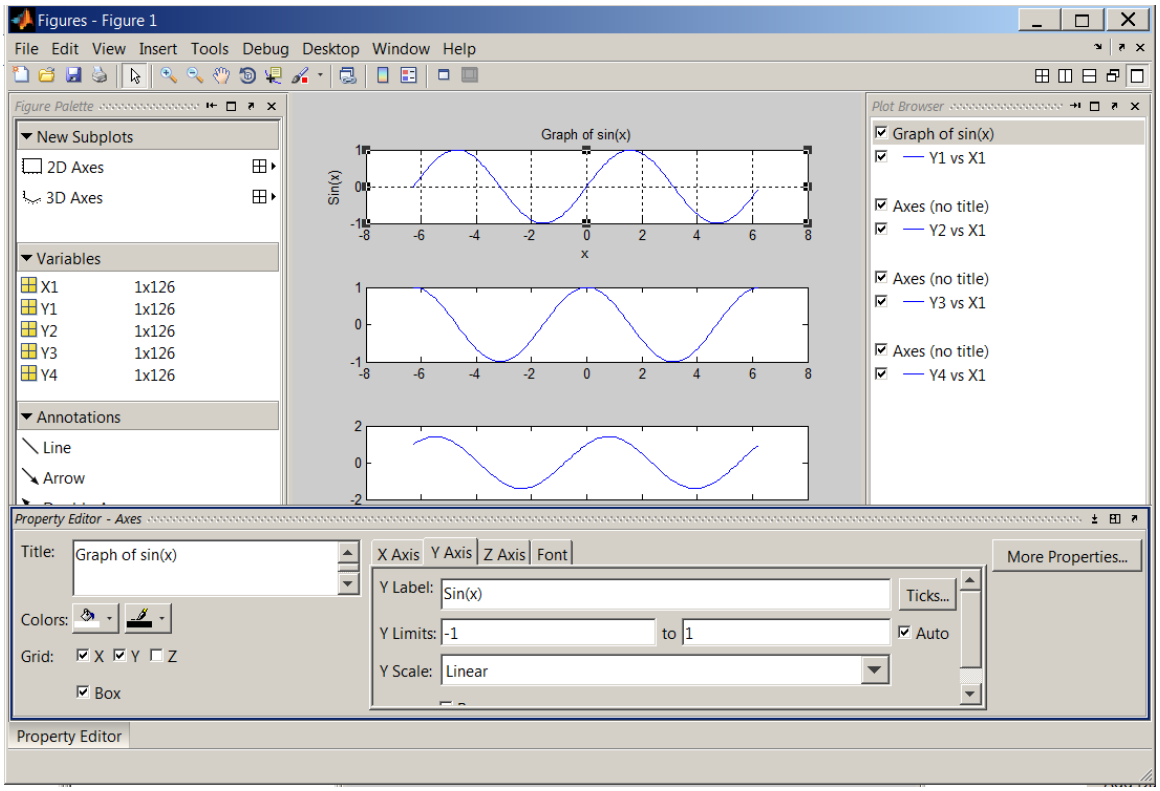


Figure 3.8: Using "Property Editor".

Save the figure as `sinxcosx.fig` in the current directory.

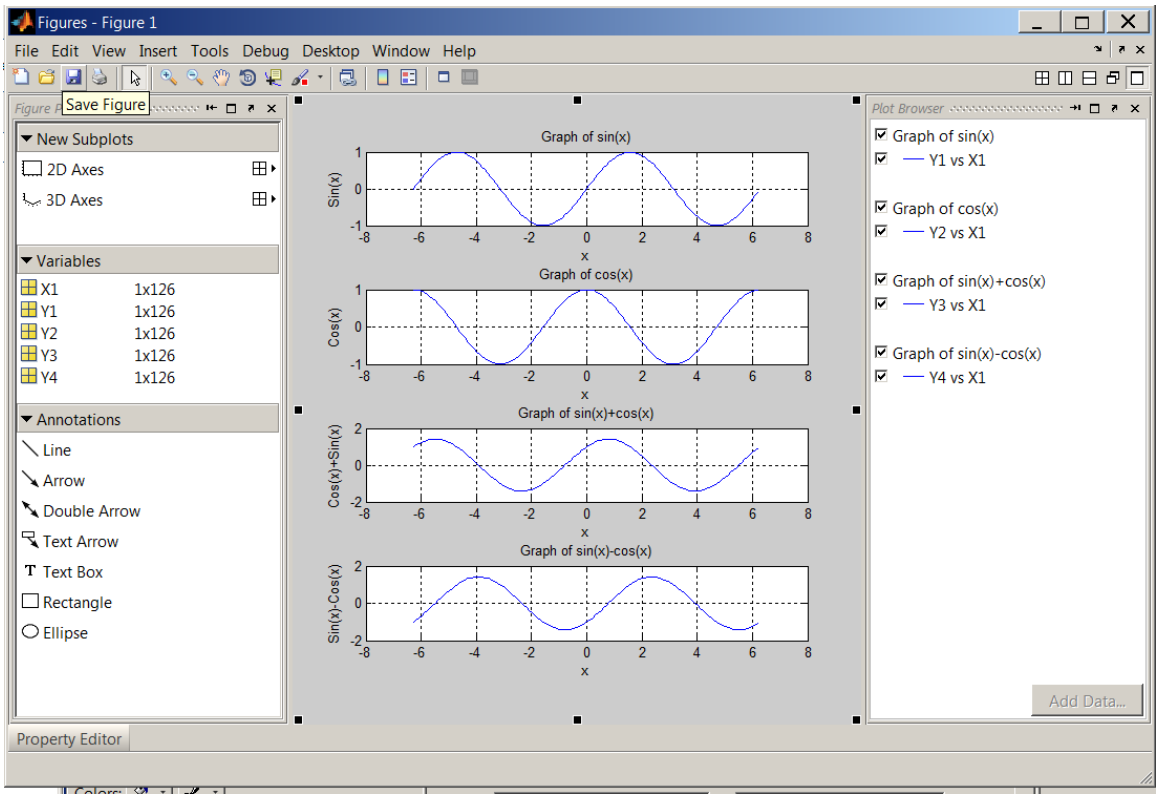


Figure 3.9: The four subplots generated with "Plot Tools".

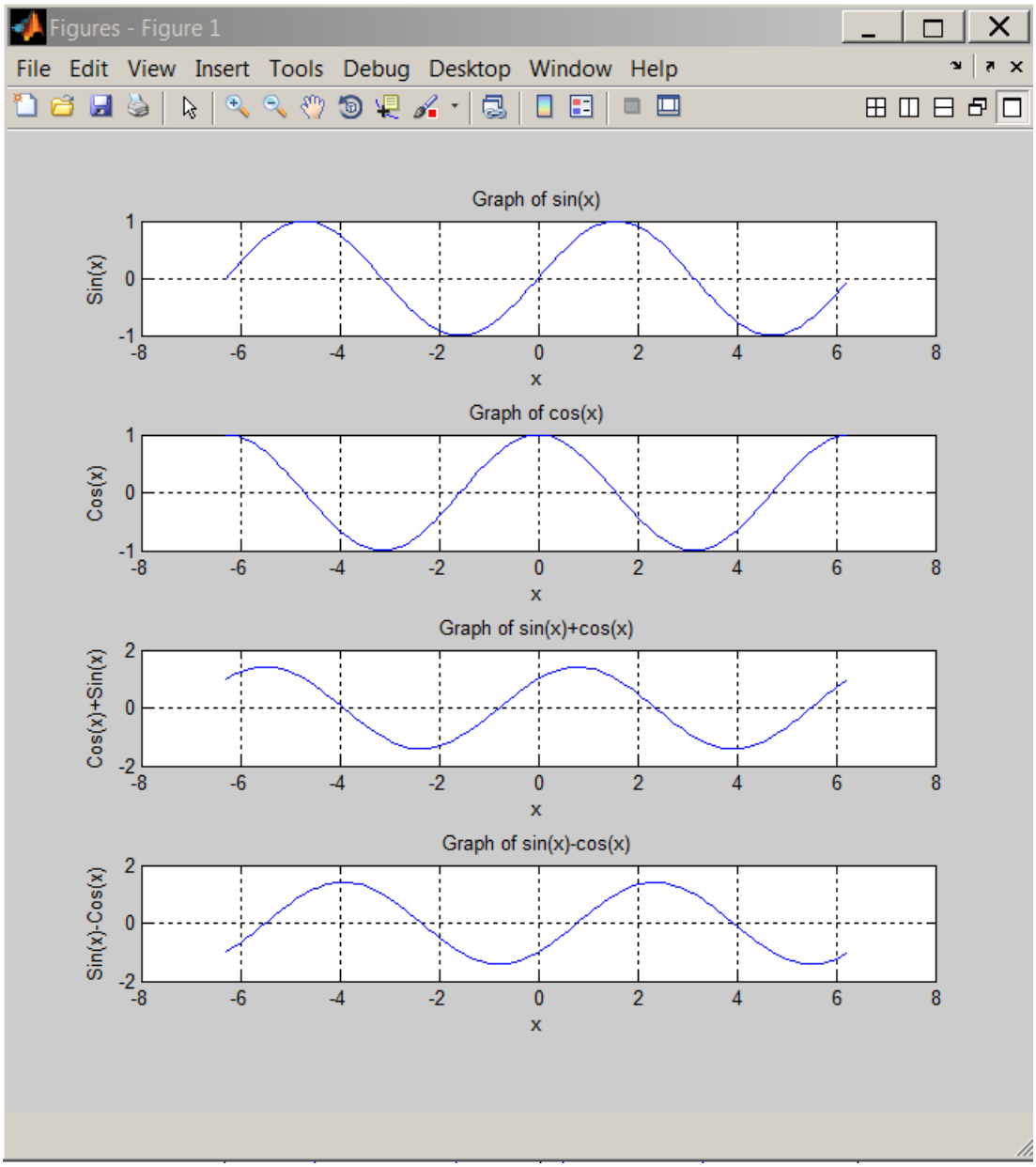


Figure 3.10: The four subplots in a single figure.

3.1.2 Three-Dimensional Plots

3D plots can be generated from the Command Window as well as by GUI alternatives. This time, we will go back to the Command Window.

3.1.2.1 The `plot3` Statement

With the X1,Y1,Y2 and Y2 variables still in the workspace, type in `plot3(X1,Y1,Y2)` at the MATLAB prompt. A figure will be generated, click "Show Plot Tools and Dock Figure".

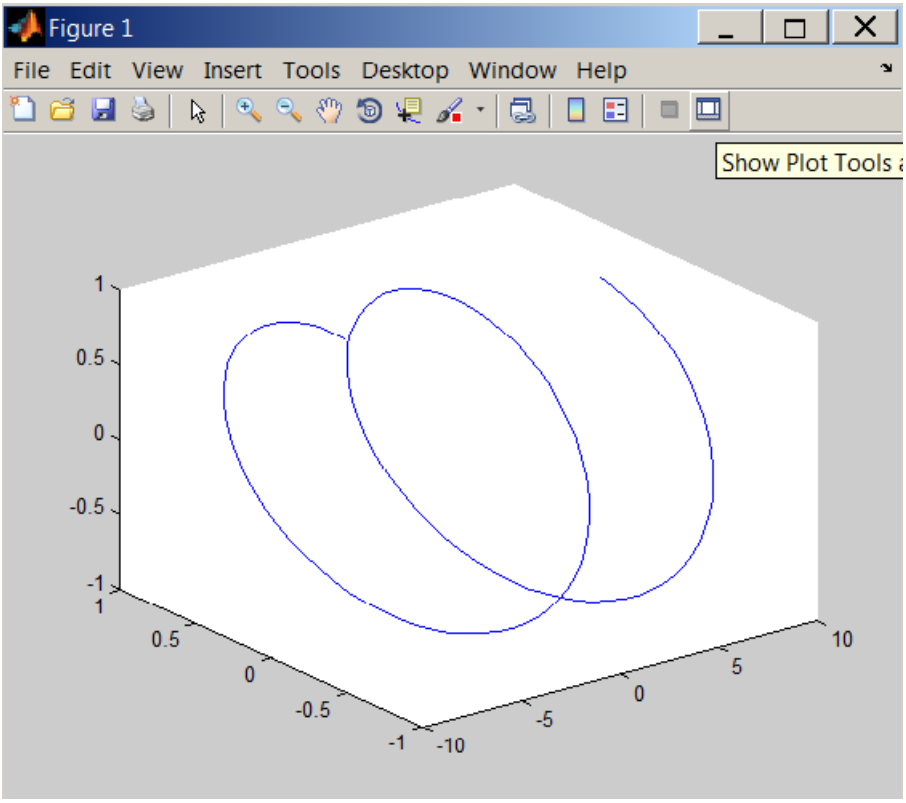


Figure 3.11: A raw 3D figure is generated with `plot3`.

Use the property editor to make the following changes.

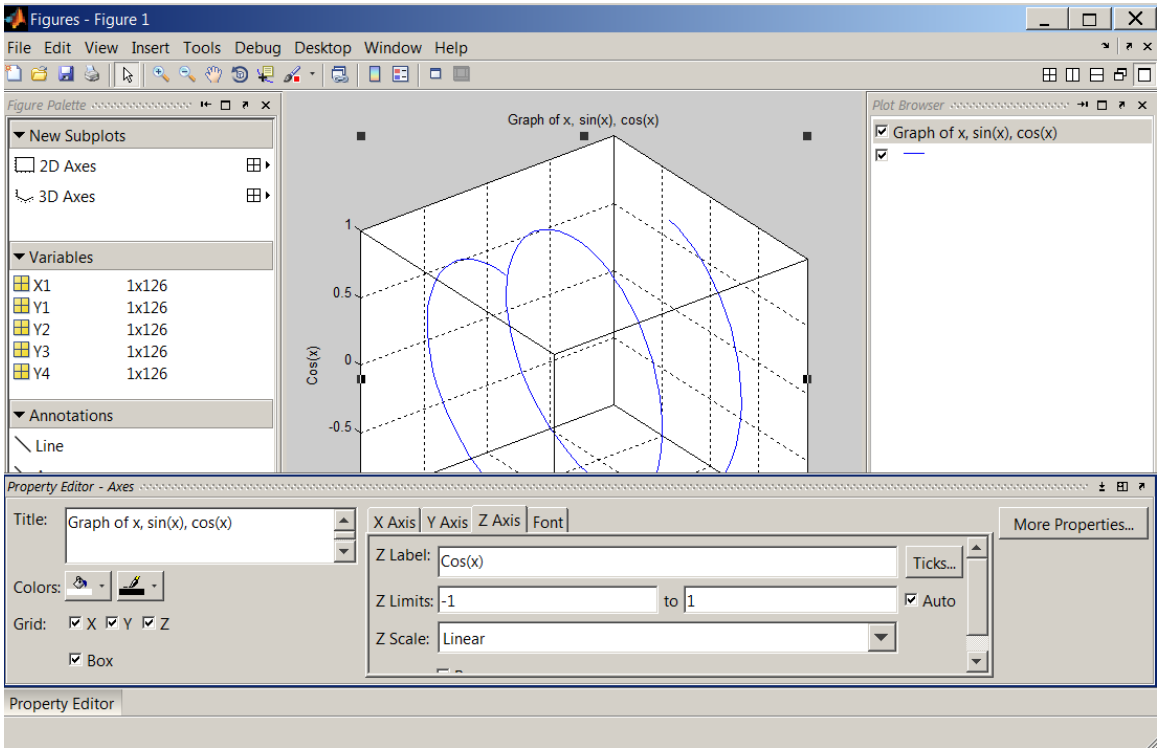


Figure 3.12: 3D Property Editor.

The final result should look like this:

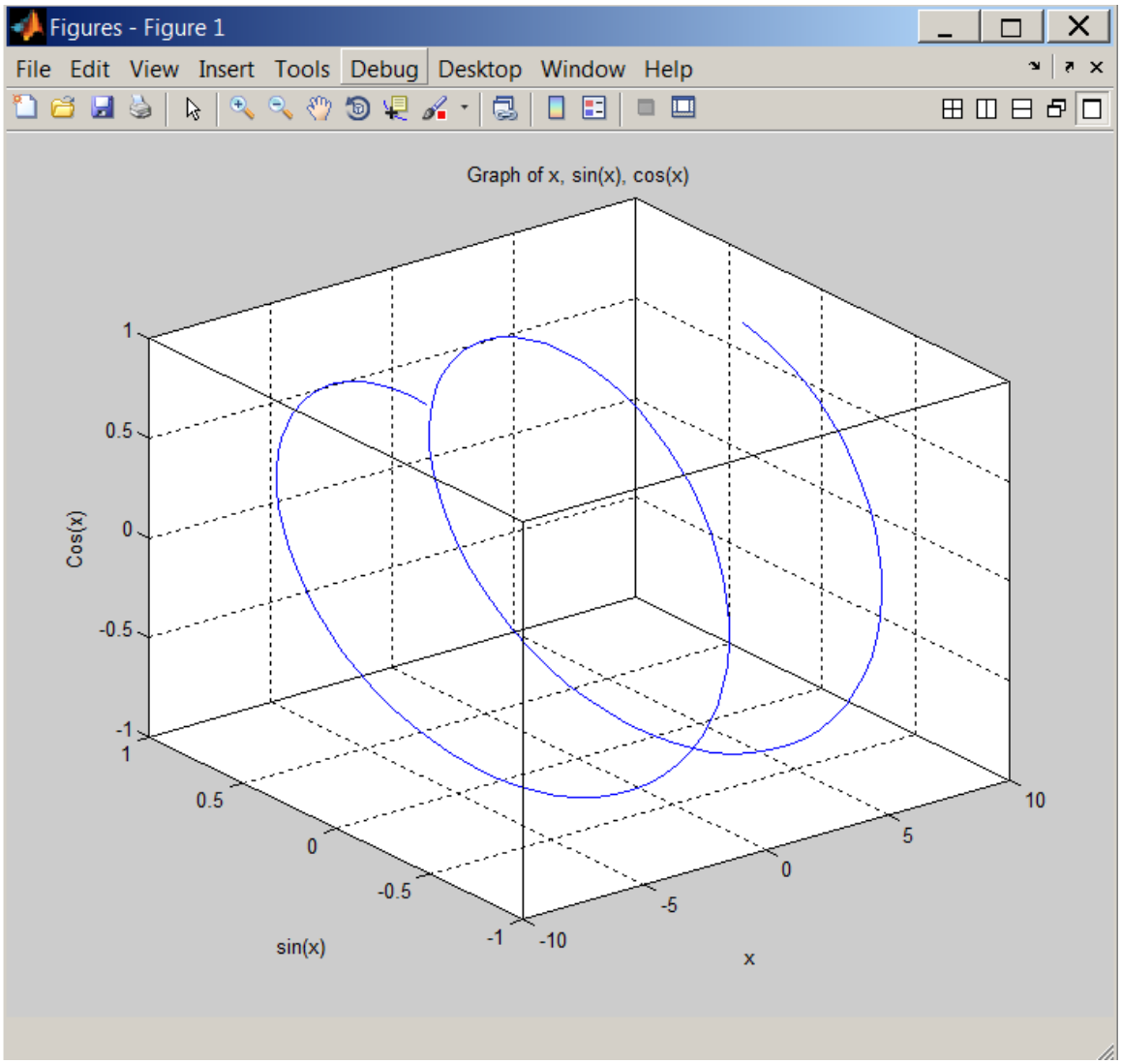


Figure 3.13: 3D graph of x, sin(x), cos(x)

Use `help` or `doc` commands to learn more about 3D plots, for example, `image(x)`, `surf(x)` and `mesh(x)`.

3.1.3 Quiver or Velocity Plots

To plot vectors, it is useful to draw arrows so that the direction of the arrow points the direction of the vector and the length of the arrow is vector's magnitude. However the standard plot function is not suitable for this purpose. Fortunately, MATLAB has `quiver` function appropriately named to plot arrows. `quiver(x,y,u,v)` plots vectors as arrows at the coordinates (x,y) with components (u,v). The matrices x, y, u, and v must all be the same size and contain corresponding position and velocity components.

Example 3.2
Calculate the magnitude of forces OA, OB and the resultant R of OA and OB shown below. Plot

all three forces on x-y Cartesian coordinate system[2].

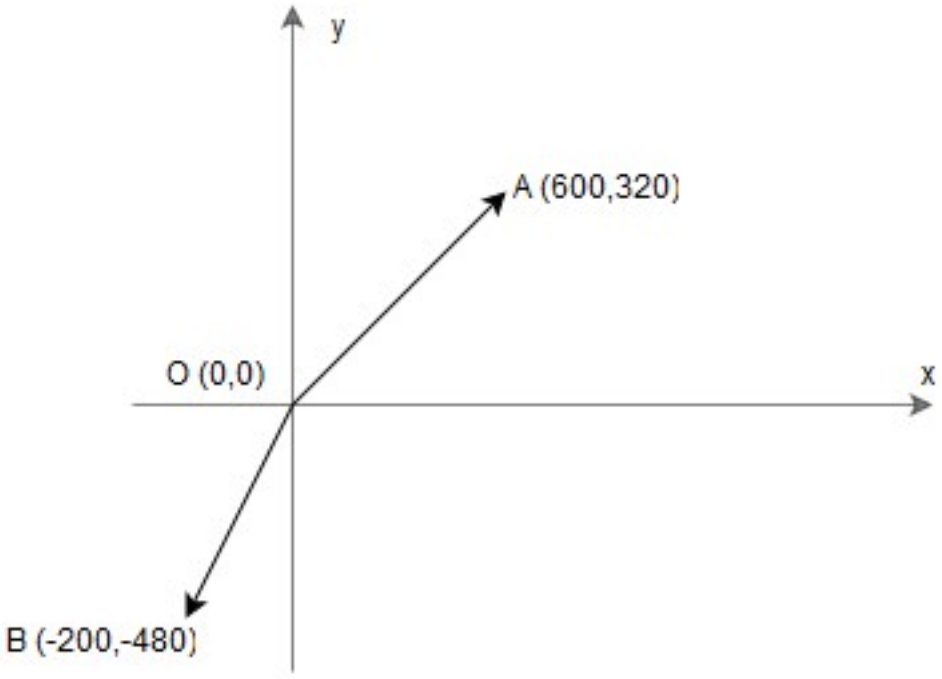


Figure 3.14: Quiver Plot.

```
    % Preparation
clear   % removes all variables from the current workspace,
        % releasing them from system memory.
clc     % clears all input and output from the Command Window display,
        % giving you a "clean screen."
% Input and Computation
OA=[600 320];    % Force 1
magOA=sqrt(sum(OA.^2));
OB=[-200 -480];  % Force 2
magOB=sqrt(sum(OB.^2));
OC=OA+OB;        % The resultant of OA and OB
magOC=sqrt(sum(OC.^2)); % The magnitude of resultant force OC
angleMag=atan(OC(2)/OC(1))*180/pi; % angle of OC in degrees
% Output
disp(' ')                    % Display blank line
str1= ['The magnitude of the resultant force is ', num2str(magOC), ' N.'];
disp(str1);
str2= ['The angle of the resultant force is ', num2str(angleMag), ' degrees.'];
disp(str2);
```

[2] *Applied Engineering Mechanics* by A. Jensen, H. Chenoweth McGraw-Hill Ryerson Limited ©1972, (p. 15)

```
% Plot Preparation
starts = zeros(3,2); % Origin for all 3 forces, 3x2 "zero" matrix
ends = [OA;OB;OC]; % End point for all 3 forces
vectors = horzcat(starts,ends); % Concatenate arrays horizontally
% Plot Forces on x-y Cartesian Coordinate System
% The following MATLAB function plots vectors as arrows
% at the coordinates specified in each corresponding
% pair of elements in x and y.
quiver( vectors( :,1 ), vectors( :,2 ), vectors( :,3 ), vectors( :,4 ));
axis equal
grid
title('Forces on x-y Cartesian Coordinate System')
xlabel('x') % x-axis label
ylabel('y') % y-axis label
view(2) % setting view to 2-D
```

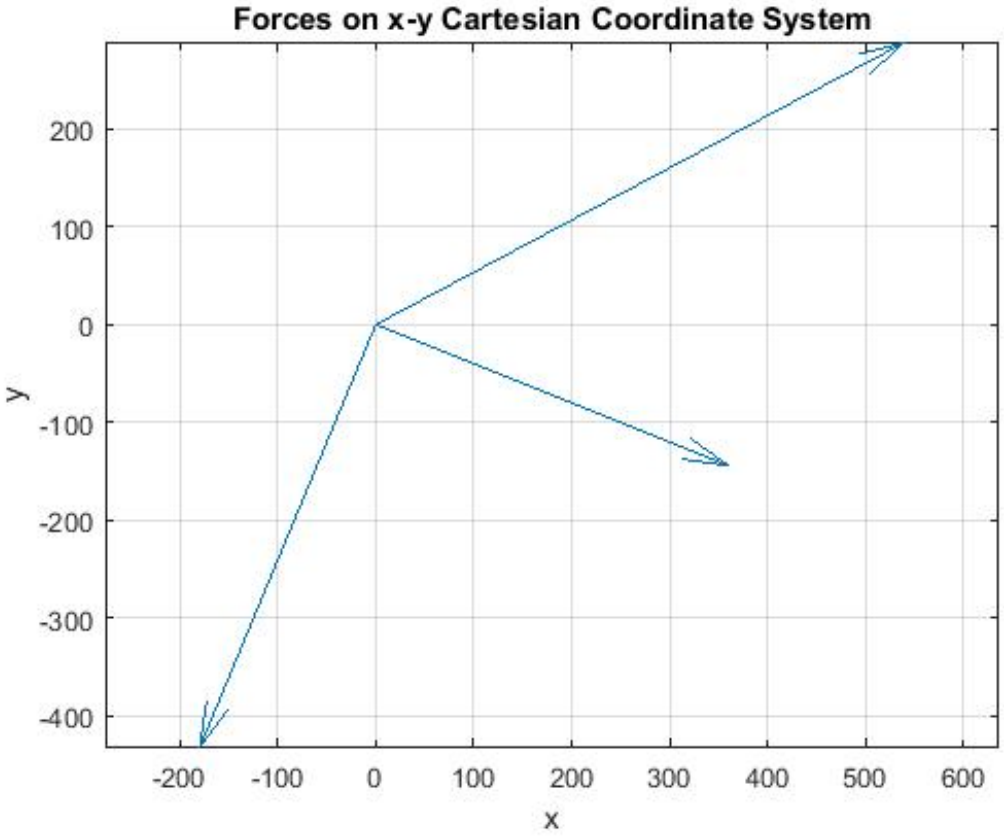


Figure 3.15: Output of `quiver` function.

Example 3.3
Write an interactive script to calculate the resultant R of forces F1, F2 and F3 shown below and

plot all four forces on x-y Cartesian coordinate system[3].

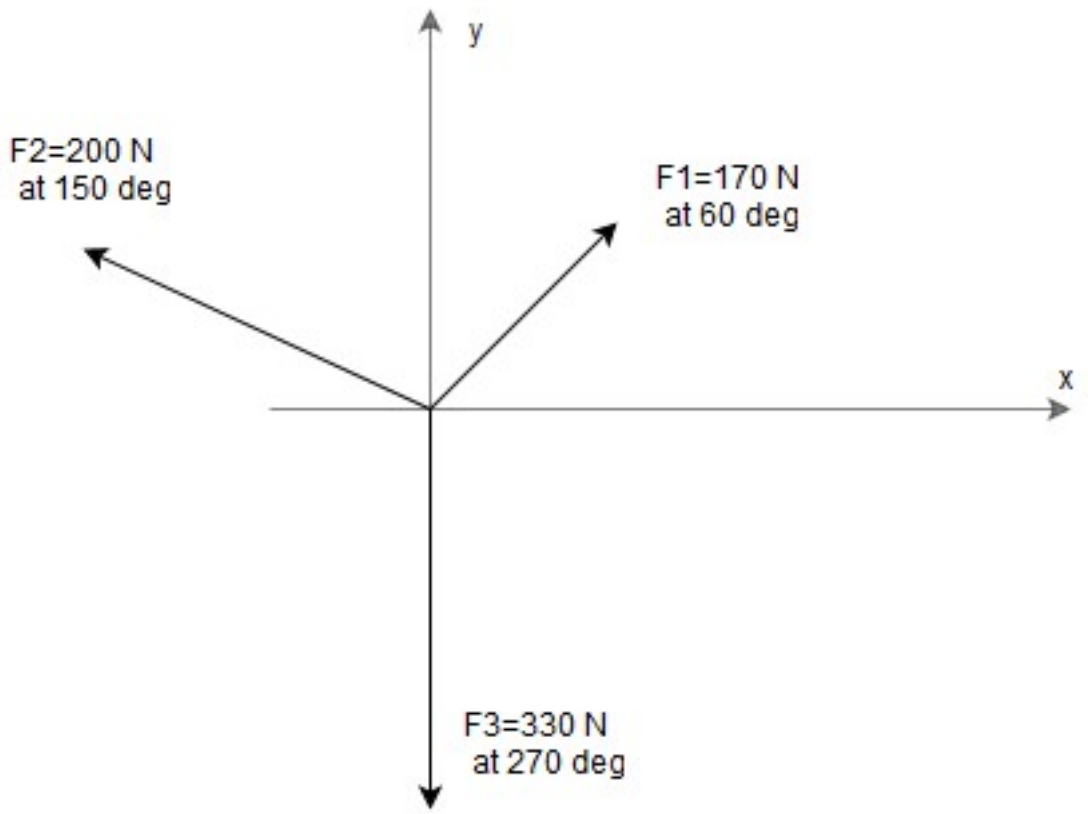


Figure 3.16: An example for quiver3 plot.

```
    clear
clc
disp('This script computes the resultant of three forces on x-y Cartesian coordinate system.')
f1=input('Enter the magnitude of first force in N: ');
theta1=input('Enter the angle of first force in deg: ');
f2=input('Enter the magnitude of second force in N: ');
theta2=input('Enter the angle of second force in deg: ');
f3=input('Enter the magnitude of third force in N: ');
theta3=input('Enter the angle of third force in deg: ');
x1=f1*cos(theta1*pi/180); % The components of force
y1=f1*sin(theta1*pi/180); % The components of force
F1=[x1 y1];  % Force 1
x2=f2*cos(theta2*pi/180); % The components of force
y2=f2*sin(theta2*pi/180); % The components of force
F2=[x2 y2];  % Force 2
```

[3] *Applied Engineering Mechanics* by A. Jensen, H. Chenoweth McGraw-Hill Ryerson Limited ©1972, (p. 15)

```
x3=f3*cos(theta3*pi/180); % The components of force
y3=f3*sin(theta3*pi/180); % The components of force
F3=[x3 y3];  % Force 3
R=F1+F2+F3;  % The resultant of F1, F2 and F3
magR=sqrt(sum(R.^2)); % The magnitude of resultant force R
angle=atan(R(2)/R(1))*180/pi; % Angle of R in degrees
disp(' ')                     % Display blank line
str1= ['The magnitude of the resultant force is ', num2str(magR), ' N.'];
disp(str1);
str2= ['The angle of the resultant force is ', num2str(angle), ' degrees.'];
disp(str2);
starts = zeros(4,3);
ends = [F1;F2;F3;R];
ends(3,3)=0; % inputs 0s for z components, making it 3D
vectors = horzcat(starts,ends); % Concatenate arrays horizontally
quiver3( vectors( :,1 ), vectors( :,2 ), vectors( :,3 ), vectors( :,4 ), vectors( :,5 ), vectors( :,6 )
axis equal
title('Forces on x-y Cartesian coordinate system')
xlabel('x') % x-axis label
ylabel('y') % y-axis label
view(2)
```

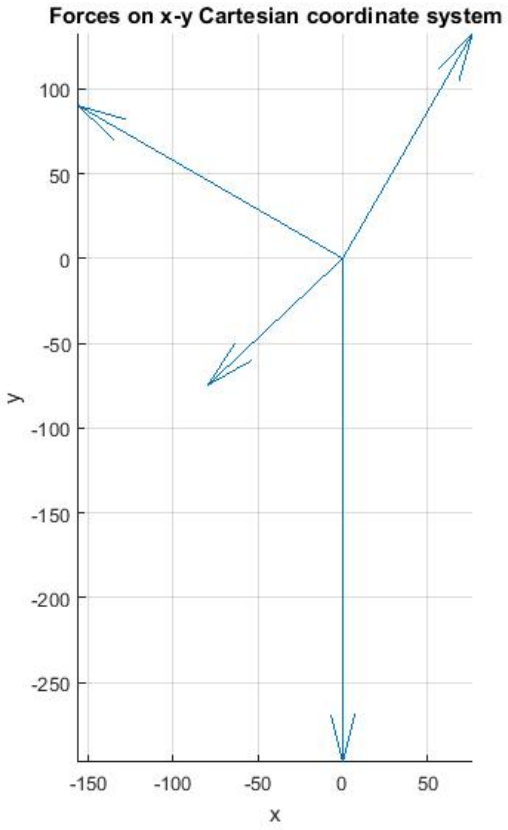


Figure 3.17: Output of `quiver3` function.

3.1.4 Summary of Key Points

1. `plot(x, y)` and `plot3(X1,Y1,Y2)` statements create 2- and 3-D graphs respectively,
2. Plots at minimum should contain the following elements: `title`, `xlabel`, `ylabel` and `legend`,
3. Annotated plots can be easily generated with GUI Plot Tools,
4. `quiver` and `quiver3` plots are useful for making vector diagrams.

3.2 Problem Set[4]

Exercise 3.2.1 ***(Solution on p. 71.)***

Plot $y = a + bx$, using the specified coefficients and ranges (use increments of 0.1):

[4] This content is available online at <http://cnx.org/content/m41466/1.7/>.

a. $a = 2$, $b = 0.3$ for $0 \leq x \leq 5$
b. $a = 3$, $b = 0$ for $0 \leq x \leq 10$
c. $a = 4$, $b = -0.3$ for $0 \leq x \leq 15$

Exercise 3.2.2 *(Solution on p. 71.)*

Plot the following functions, using increments of 0.01 and $a = 6$, $b = 0.8$, $0 \leq x \leq 5$:

a. $y = a + x^b$
b. $y = ax^b$
c. $y = a\sin(x)$

Exercise 3.2.3 *(Solution on p. 73.)*

Plot function $y = \frac{\sin(x)}{x}$ for $\frac{\pi}{100} \leq x \leq 10\pi$ using increments of $\frac{\pi}{100}$

Exercise 3.2.4 *(Solution on p. 74.)*

Data collected from Boyle's Law experiment are as follows: (Data available for download.[5])

Volume [cm^3]	Pressure [Pa]
7.34	100330
7.24	102200
7.14	103930
7.04	105270
6.89	107400
6.84	108470
6.79	109400
6.69	111140
6.64	112200

Table 3.1

Plot a graph of Pressure vs Volume, annotate your graph.

Exercise 3.2.5 *(Solution on p. 75.)*

The original data collected from Boyle's [6] experiment are as follows: (Data available for download.[7])

Volume [tube-inches]	Pressure [inches-Hg]
12	29.125
10	35.000
8	43.688
6	58.250
5	70.000
4	87.375
3	116.500

[5] See the file at <http://cnx.org/content/m41466/latest/Chp3_Exercise4.zip>
[6] *Introduction to Engineering: Modeling and Problem Solving* by J. B. Brockman, John Wiley and Sons, Inc. ©2009, (p.246)
[7] See the file at <http://cnx.org/content/m41466/latest/Chp3_Exercise5.zip>

Table 3.2

Plot a graph of Pressure vs Volume, annotate your graph.

Exercise 3.2.6 *(Solution on p. 76.)*

Display the two plots created earlier in one plot.

Exercise 3.2.7 *(Solution on p. 77.)*

A tensile test of SAE 1020 steel produced the data below (Data available for download.[8]) [9] experiment are as follows:

Extension [mm]	**Load [kN]**
0.00	0.0
0.09	1.9
0.31	6.1
0.47	9.4
2.13	11.0
5.05	11.7
10.50	12.0
16.50	11.9
23.70	10.7
27.70	9.3
34.50	8.1

Table 3.3

Plot a graph of Load vs Extension, annotate your graph.

Exercise 3.2.8 *(Solution on p. 78.)*

Given below is Stress-Strain data for a type 304 stainless steel. [10] experiment are as follows: (Data available for download.[11])

[8] See the file at <http://cnx.org/content/m41466/latest/Chp3_Exercise7.zip>

[9] *Introduction to Materials Science for Engineers | Instructor's Manual* by J. F. Shackelford, Macmillan Publishing Company. ©1992, (p.440)

[10] *Introduction to Materials Science for Engineers* by J. F. Shackelford, Macmillan Publishing Company. ©1985, (p.304)

[11] See the file at <http://cnx.org/content/m41466/latest/Chp3_Exercise8.zip>

Stress [MPa]	Strain [mm/mm]
0.0	0.0000
38.6	0.0002
77.2	0.0004
115.8	0.0006
154.4	0.0008
193.0	0.0010
218.0	0.0012
232.0	0.0014
258.0	0.0020
268.0	0.0025
273.0	0.0030
278.0	0.0035
282.0	0.0040
320.0	0.0200
382.0	0.0500
466.0	0.1000
520.0	0.1500
548.0	0.2000
550.0	0.2100
538.0	0.2500
480.0	0.3000

Table 3.4

Plot a graph of Stress vs Strain, annotate your graph.

Solutions to Exercises in Chapter 3

Solution to Exercise 3.2.1 (p. 67)

a.

```
    a=2; b=.3; x=[0:.1:5]; y=a+b*x;
plot(x,y),title('Graph of y=a+bx'),xlabel('x'),ylabel('y'),grid
```

b.

```
    a=3; b=.0; x=[0:.1:10]; y=a+b*x;
plot(x,y),title('Graph of y=a+bx'),xlabel('x'),ylabel('y'),grid
```

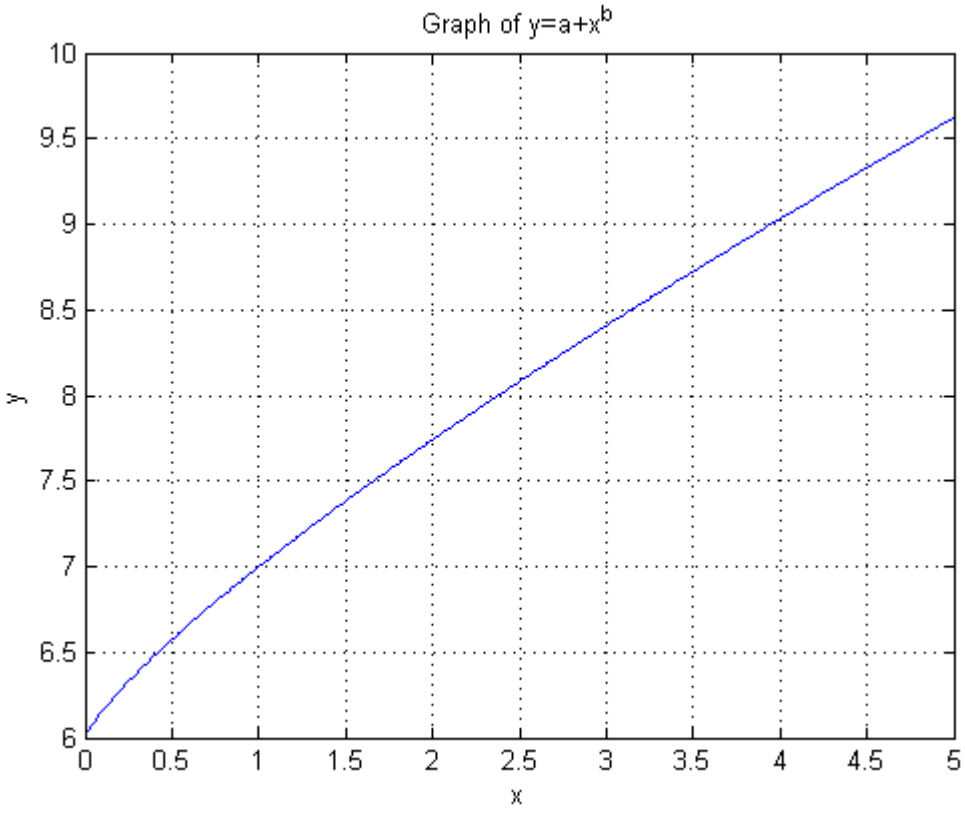
c.

```
    a=2; b=.3; x=[0:.1:5]; y=a+b*x;
plot(x,y),title('Graph of y=a+bx'),xlabel('x'),ylabel('y'),grid
```

Solution to Exercise 3.2.2 (p. 68)

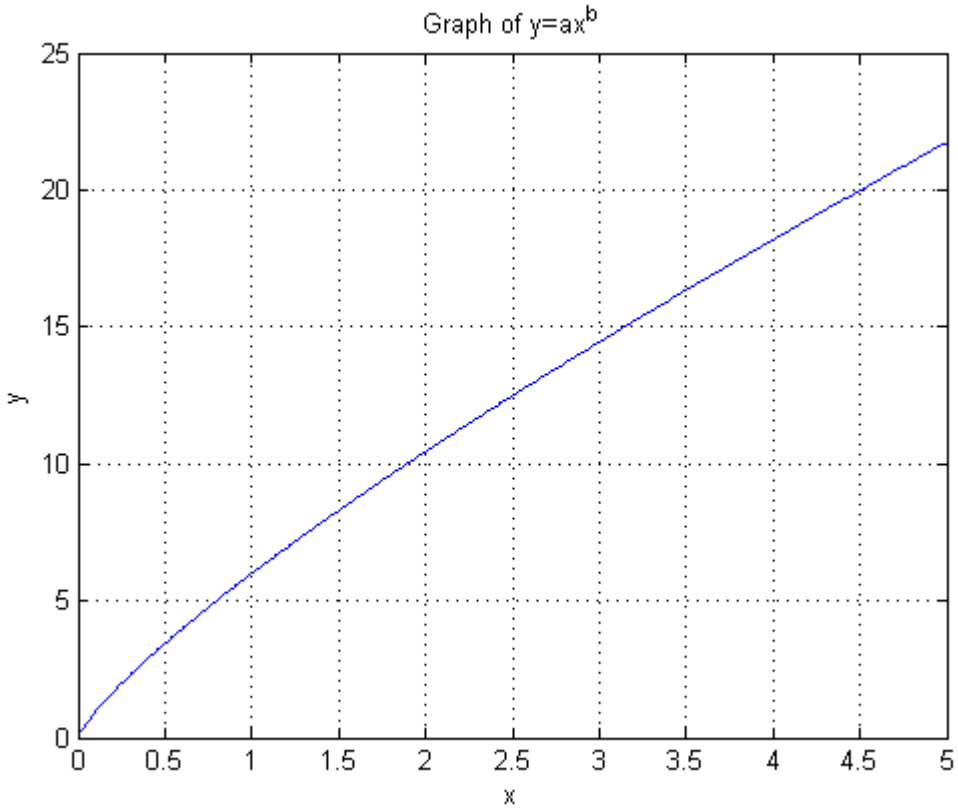
a.

```
    a=6; b=.8; x=[0:.01:5]; y=a+x.^b;
plot(x,y),title('Graph of y=a+x^b'),xlabel('x'),ylabel('y'),grid
```



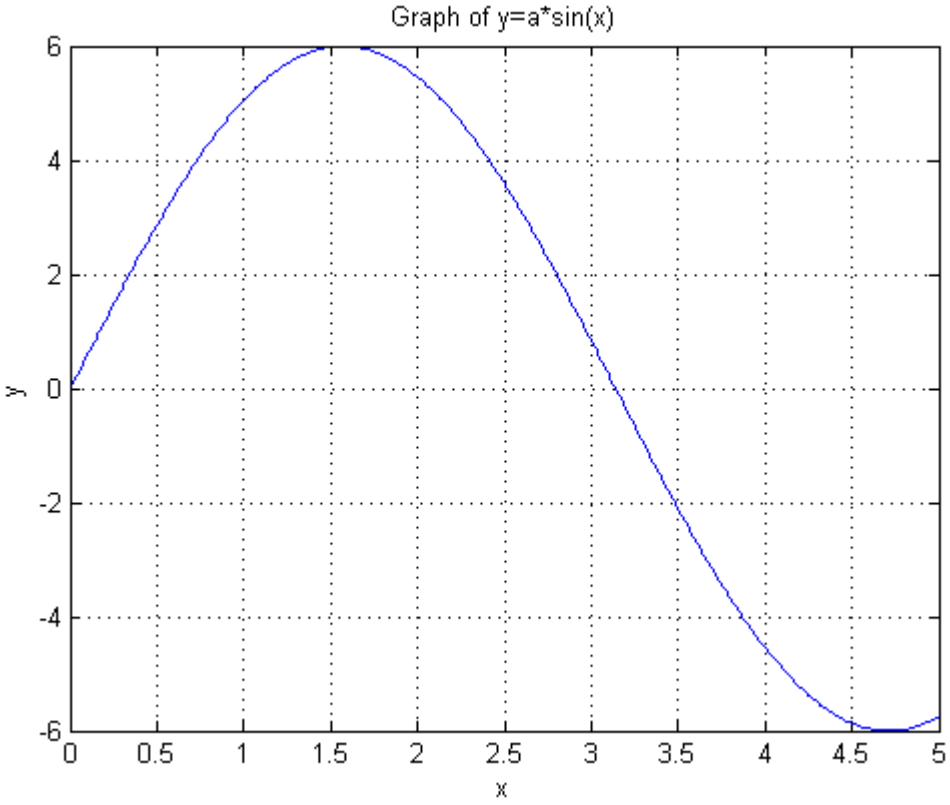
b.

```
    a=6; b=.8; x=[0:.01:5]; y=a*x.^b;
plot(x,y),title('Graph of y=ax^b'),xlabel('x'),ylabel('y'),grid
```



c.

```
    a=6; x=[0:.01:5]; y=a*sin(x);
plot(x,y),title('Graph of y=a*sin(x)'),xlabel('x'),ylabel('y'),grid
```



Solution to Exercise 3.2.3 (p. 68)

```
    x = pi/100:pi/100:10*pi;
y = sin(x)./x;
plot(x,y),title('Graph of y=sin(x)/x'),xlabel('x'),ylabel('y'),grid
```

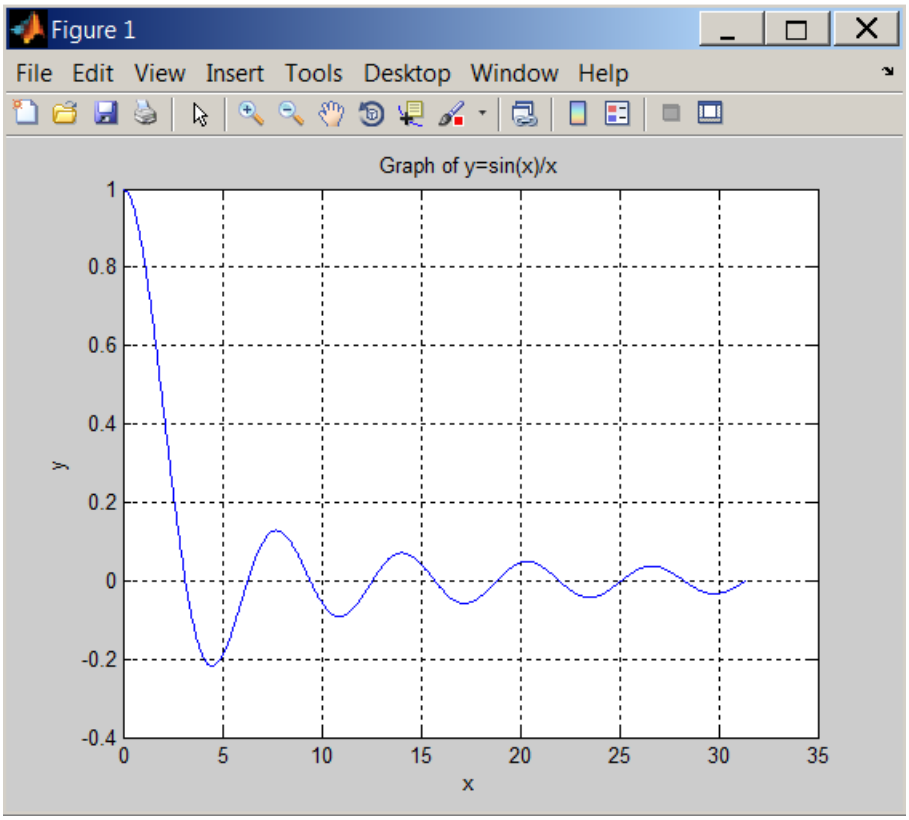
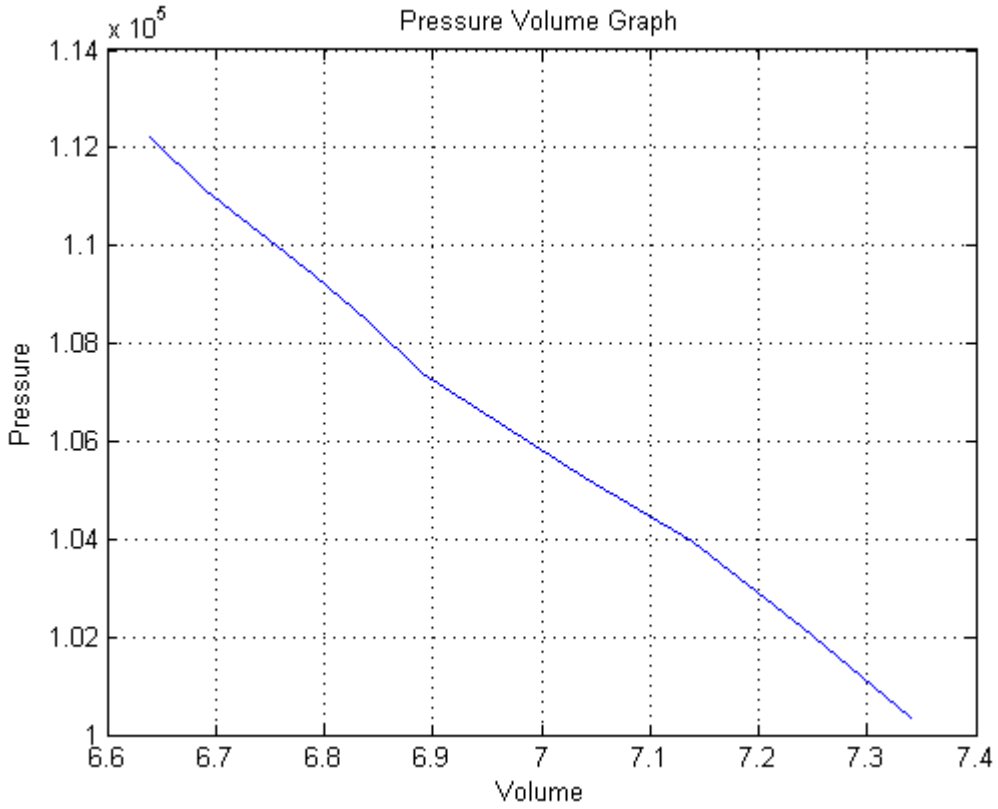


Figure 3.18: Graph of $y = \frac{\sin(x)}{x}$

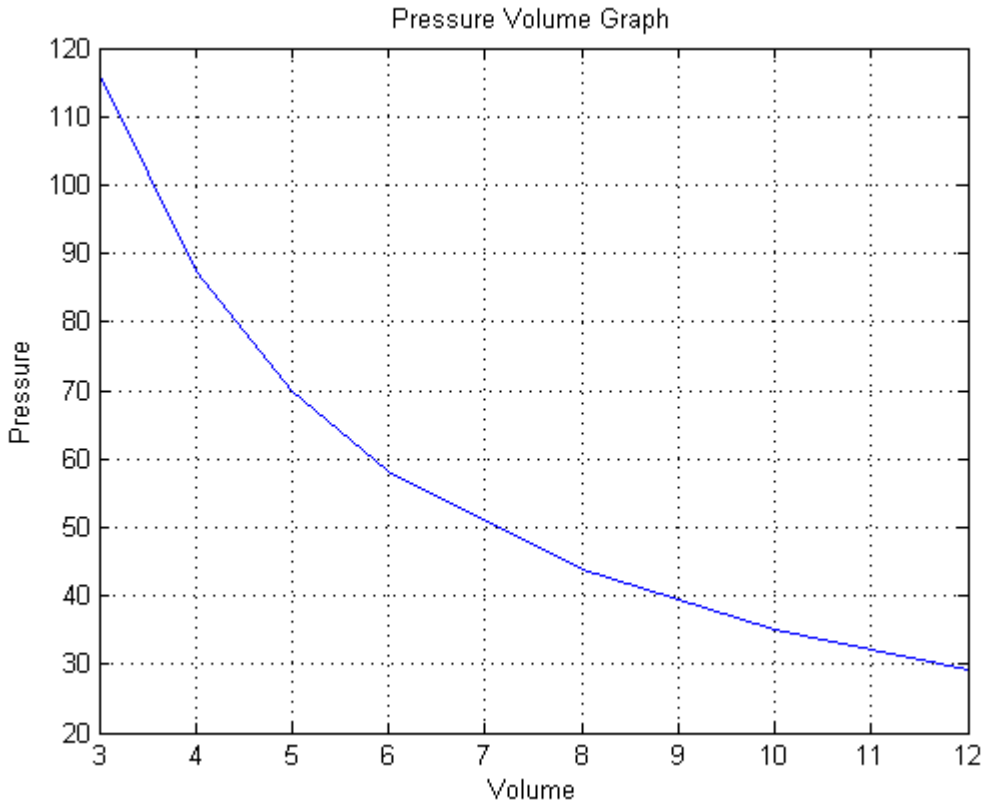
Solution to Exercise 3.2.4 (p. 68)

```
    Pressure=[100330,102200,103930,105270,107400,108470,109400,111140,112200];
Volume=[7.34,7.24,7.14,7.04,6.89,6.84,6.79,6.69,6.64];
plot(Volume, Pressure),title('Pressure Volume Graph'),xlabel('Volume'),ylabel('Pressure'),grid
```

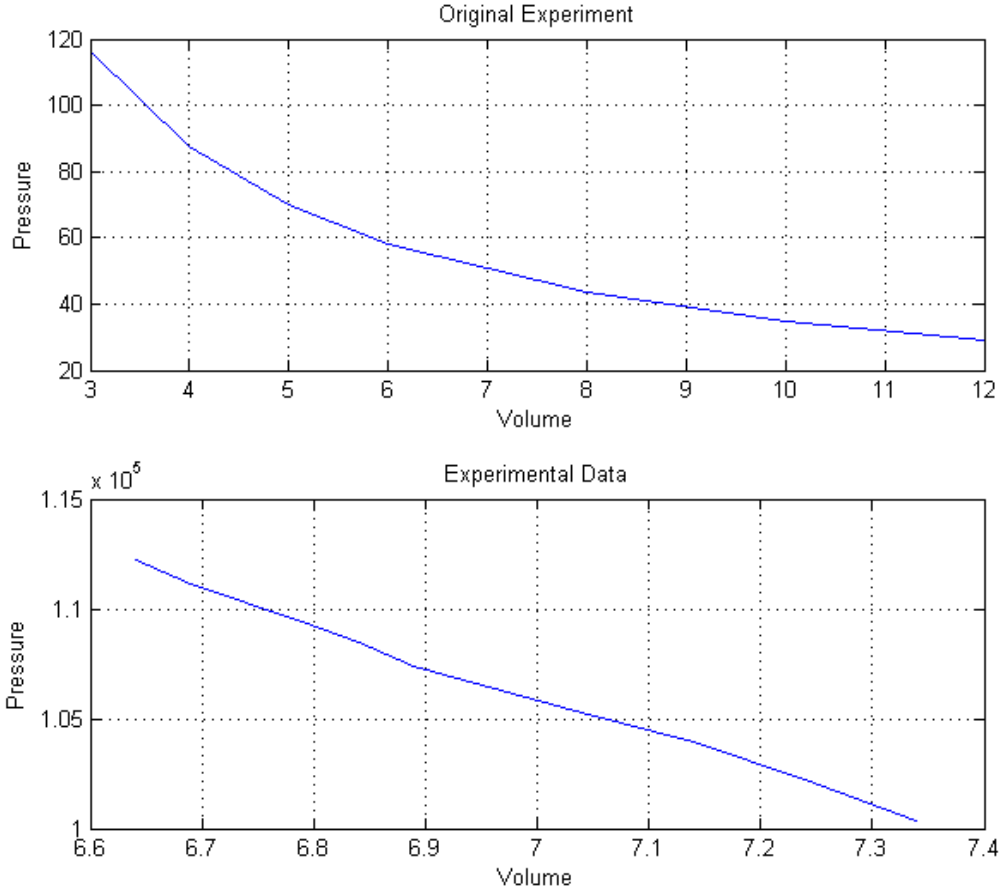


Solution to Exercise 3.2.5 (p. 68)

```
≫ P=[29.125,35,43.688,58.25,70,87.375,116.5];
≫ V=[12,10,8,6,5,4,3];
≫ plot(V,P),title('Pressure Volume Graph'),xlabel('Volume'),ylabel('Pressure'),grid
```

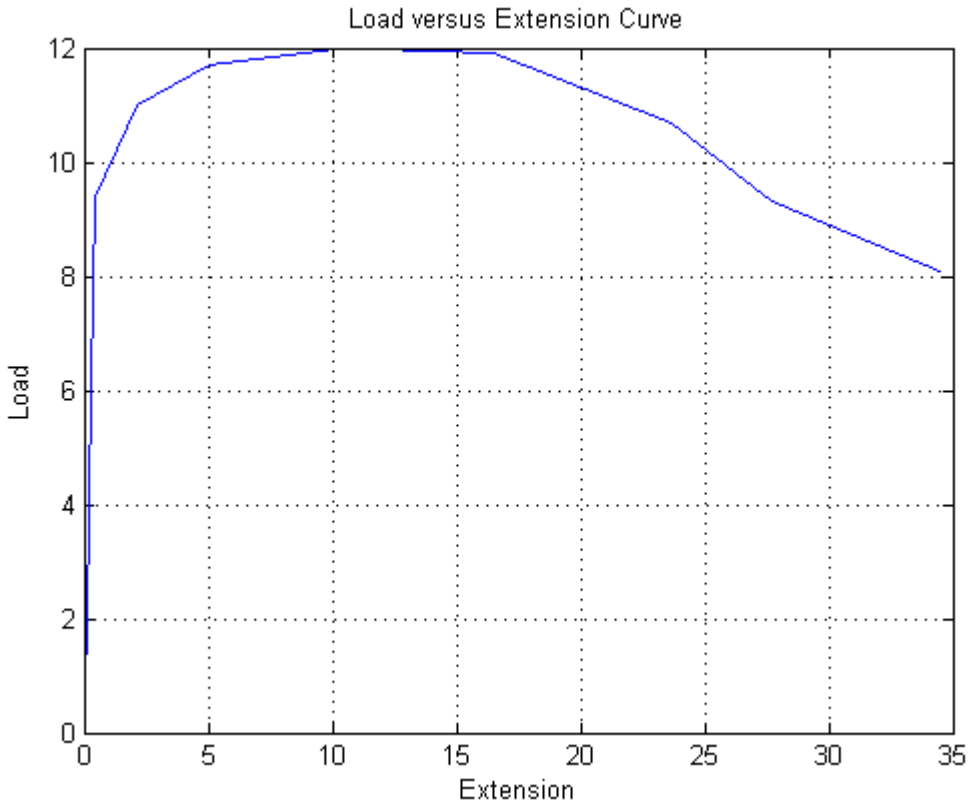

Pressure Volume Graph
Pressure
Volume
120
110
100
90
80
70
60
50
40
30
20
3
4
5
6
7
8
9
10
11
12

Solution to Exercise 3.2.6 (p. 69)

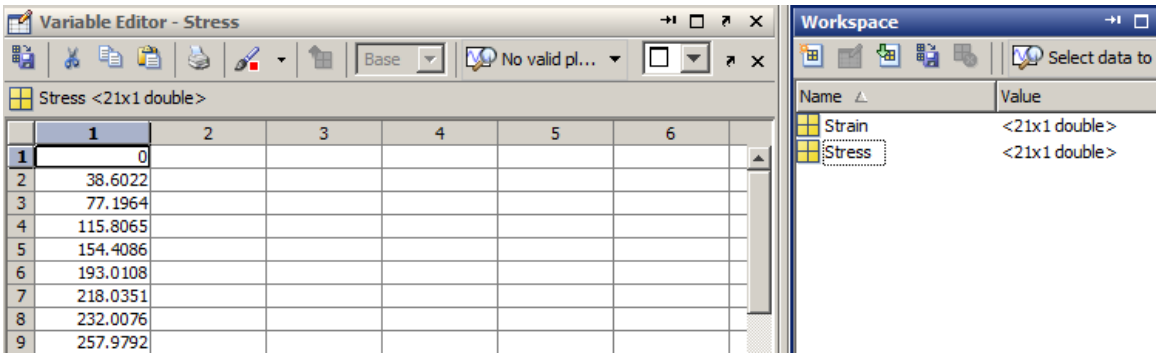


Solution to Exercise 3.2.7 (p. 69)

```
    Extension=[0.00,0.09,0.31,0.47,2.13,5.05,10.50,16.50,23.70,27.70,34.50];
Load=[0.0,1.9,6.1,9.4,11.0,11.7,12.0,11.9,10.7,9.3,8.1];

plot(Extension, Load),title('Load versus Extension Curve'),xlabel('Extension'),ylabel('Load'),grid
```

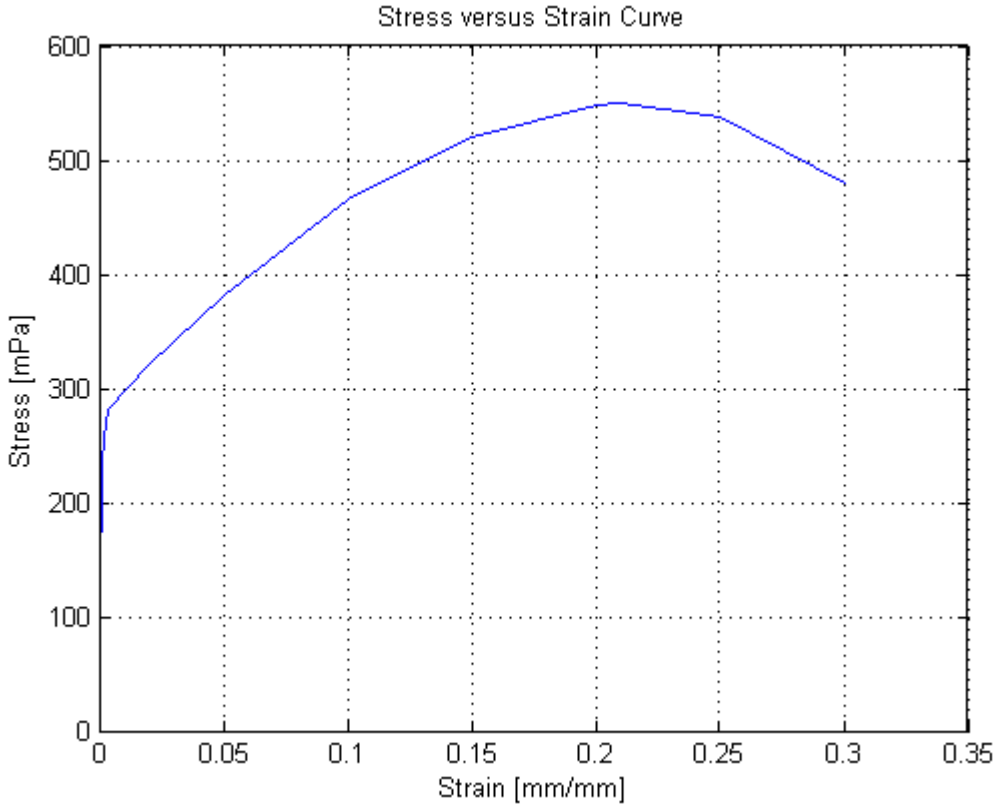


Solution to Exercise 3.2.8 (p. 69)
The data can be entered using Variable Editor:



Then execute the following:

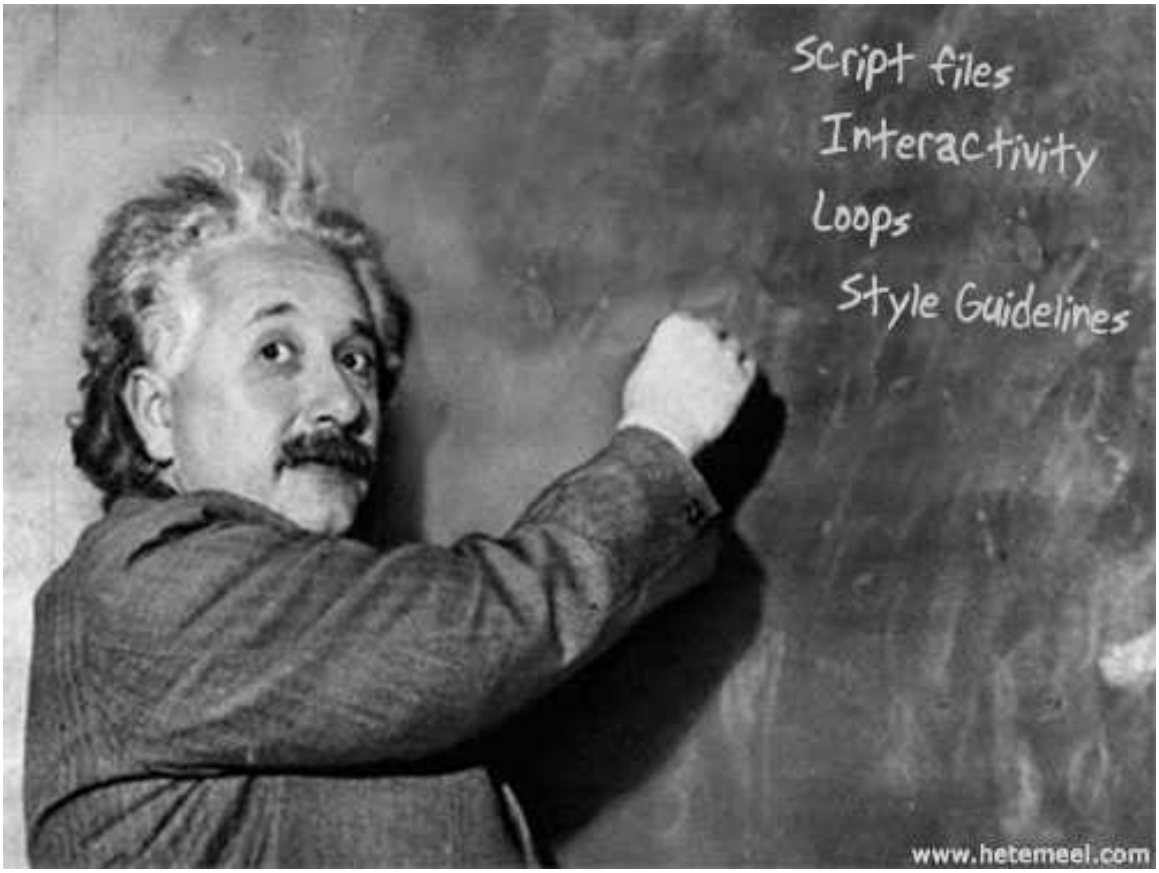
```
plot(Strain,Stress),title('Stress versus Strain Curve'),xlabel('Strain [mm/mm]'),ylabel('Stress [mP
```


Stress versus Strain Curve
Stress [mPa]
Strain [mm/mm]
0
100
200
300
400
500
600
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35

Chapter 4

Introductory Programming

4.1 Writing Scripts to Solve Problems[1]



MATLAB provides scripting and automation tools that can simplify repetitive computational tasks. For example, a series of commands executed in a MATLAB session to solve a problem can be saved in a script file called an m-file. An m-file can be executed from the command line by typing the name of the file or by pressing the run button in the built-in text editor tool bar.

[1] This content is available online at <http://cnx.org/content/m41440/1.6/>.

4.1.1 Script Files

A script is a file containing a sequence of MATLAB statements. Script files have a filename extension of .m. By typing the filename at the command prompt, we can run the script and obtain results in the command window.

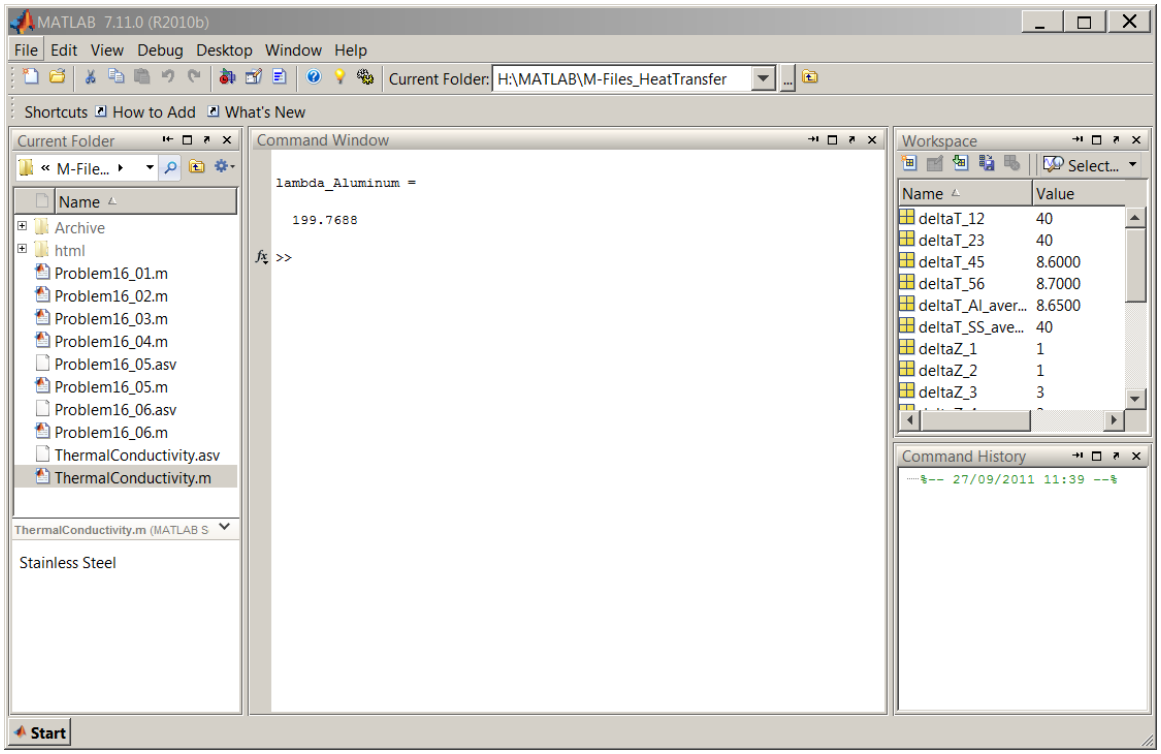


Figure 4.1: Number of m-files are displayed in the Current Folder sub-window.

A sample m-file named `ThermalConductivity.m` is displayed in Text Editor below. Note the triangle (in green) run button in the tool bar, pressing this button executes the script in the command window.

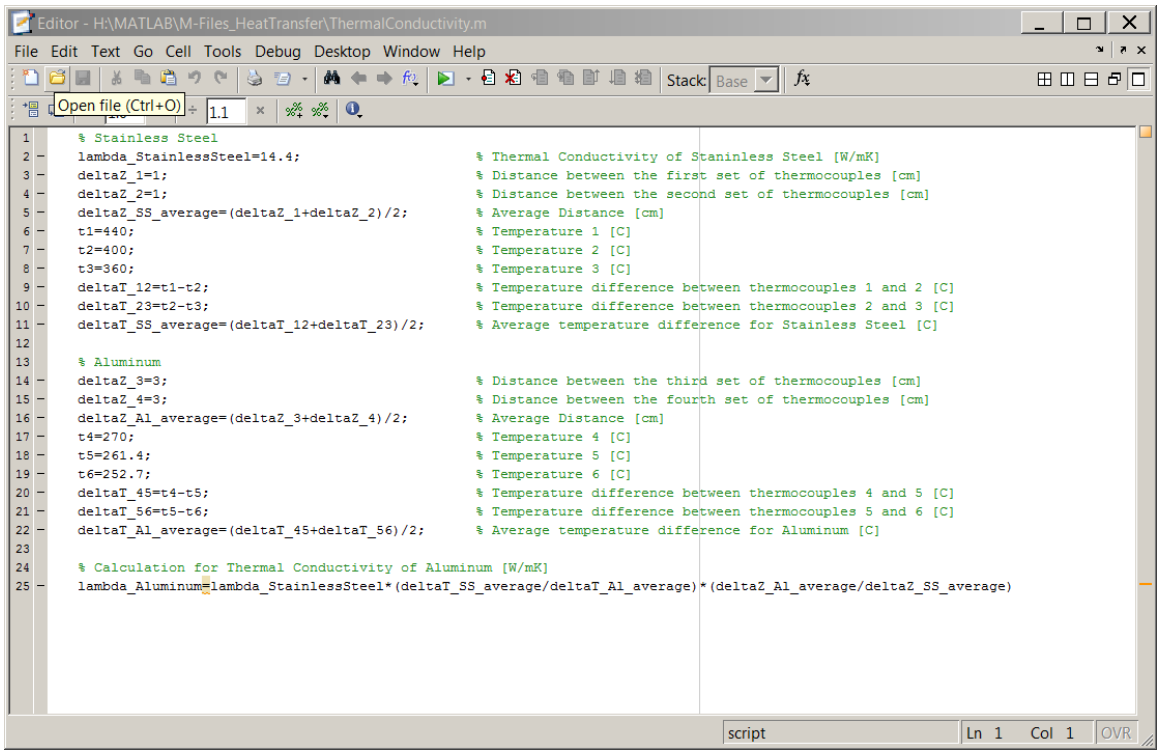


Figure 4.2: The content of `ThermalConductivity.m` file is displayed in Text Editor.

Now let us see how an m-file is created and executed.

Example 4.1

A cylindrical acetylene bottle with a radius r=0.3 m has a hemispherical top. The height of the cylindrical part is h=1.5 m. Write a simple script to calculate the volume of the acetylene bottle.

To solve this problem, we will first apply the volume of cylinder equation (4.1). Using the volume of sphere equation (4.2), we will calculate the volume of hemisphere (4.3). The total volume of the acetylene bottle is found with the sum of volumes equation (4.4).

$$V_{\text{cylinder}} = \pi r^2 h \tag{4.1}$$

$$V_{\text{sphere}} = \frac{4}{3}\pi r^3 \tag{4.2}$$

$$V_{\text{top}} = \frac{2}{3}\pi r^3 \tag{4.3}$$

$$V_{\text{acetylene bottle}} = V_{\text{cylinder}} + V_{\text{top}} \tag{4.4}$$

To write the script, we will use the built-in text editor. From the menu bar select File > New > Script. The text editor window will open in a separate window. First save this file as `AcetyleneBottle.m`. In that window type the following code paying attention to the use of percentage and semicolon symbols to comment out the lines and suppress the output, respectively.

```
    % This script computes the volume of an acetylene bottle with a radius r=0.3 m,
% a hemispherical top and a height of cylindrical part h=1.5 m.
r=0.3;                          % Radius [m]
h=1.5;                          % Height [m]
Vol_top=(2*pi*r^3)/3;           % Calculating the volume of hemispherical top [m3]
Vol_cyl=pi*r^2*h;               % Calculating the volume of cylindrical bottom [m3]
Vol_total=Vol_top+Vol_cyl       % Calculating the total volume of acetylene bottle [m3]
```

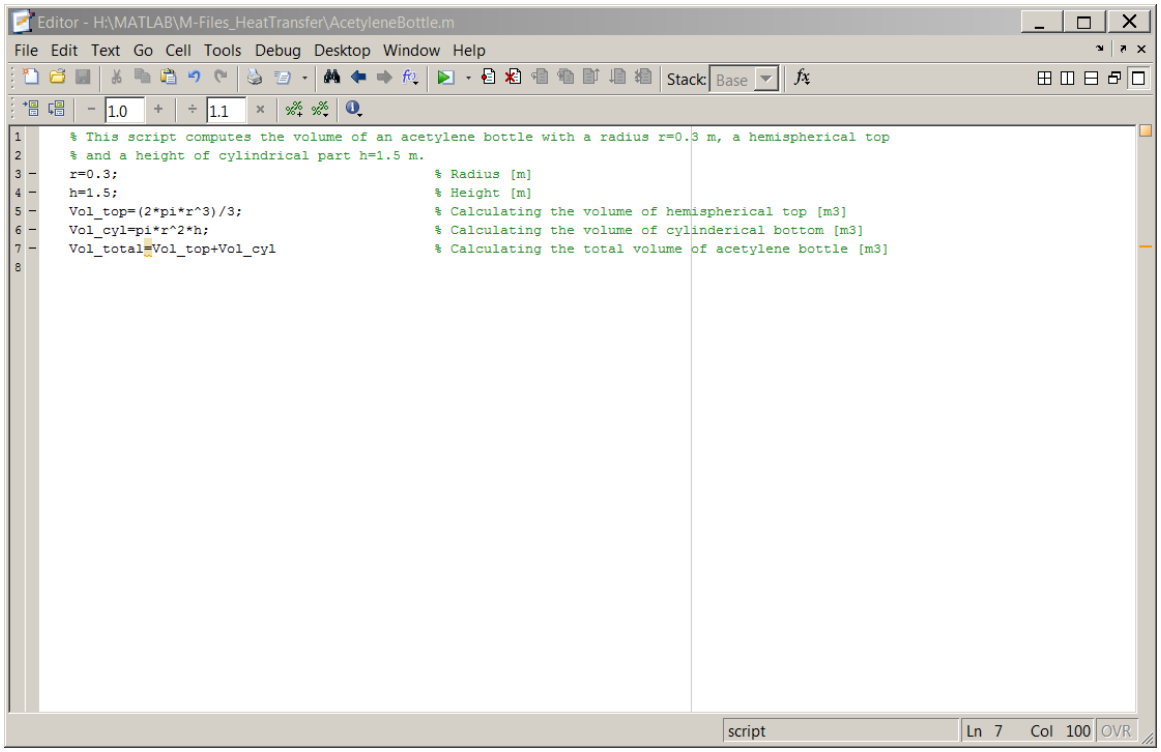


Figure 4.3: Script created with the built-in text editor.

After running the script by pressing the green button in the Text Editor tool bar, the output is displayed in the command window as shown below.

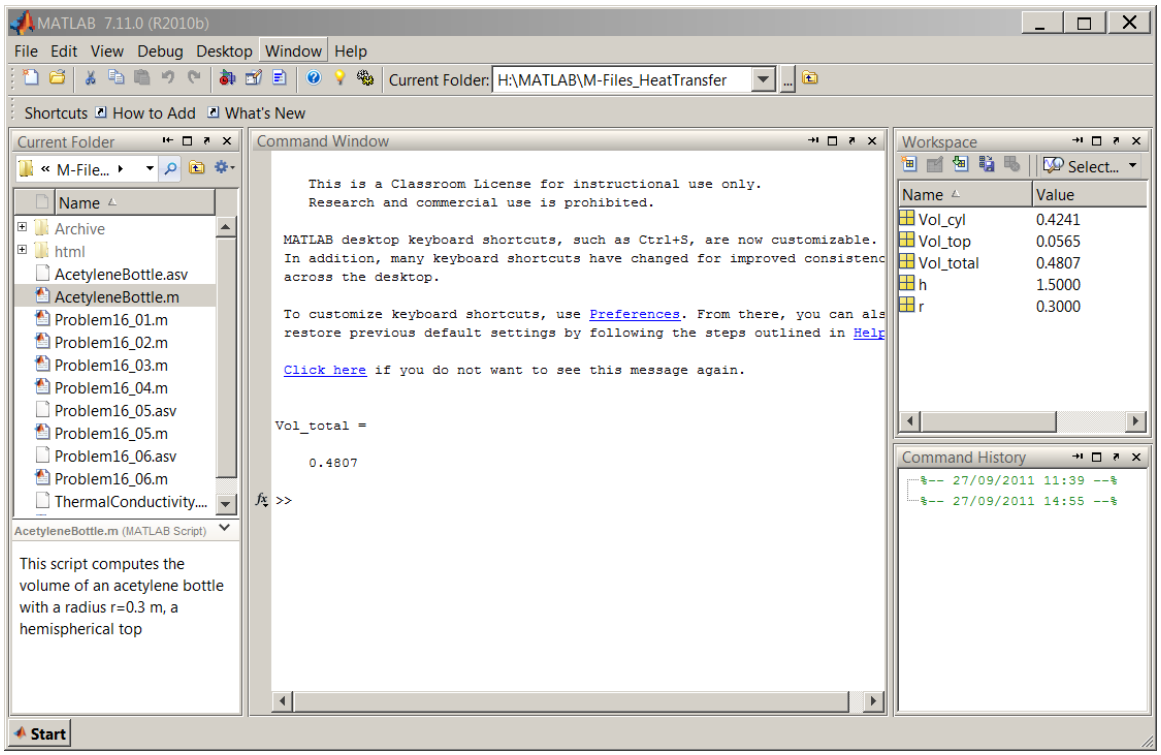


Figure 4.4: The MATLAB output in the command window.

4.1.2 The `input` Function

Notice that the script we have created above (Example 4.1) is not interactive and computes the total volume only for the variables defined in the m-file. To make this script interactive we will make some changes to the existing `AcetyleneBottle.m` by adding `input` function and save it as `AcetyleneBottleInteractive.m`.

The syntax for `input` is as follows:

```
userResponse = input('prompt')
```

Example 4.2

Now, let's incorporate the `input` command in `AcetyleneBottleInteractive.m` as shown below and the subsequent figure:

```
% This script computes the volume of an acetylene bottle
% user is prompted to enter
    % a radius r for a hemispherical top
    % a height h for a cylindrical part
r=input('Enter the radius of acetylene bottle in meters ');
h=input('Enter the height of cylindrical part of acetylene bottle in meters ');
Vol_top=(2*pi*r^3)/3;          % Calculating the volume of hemispherical top [m3]
Vol_cyl=pi*r^2*h;              % Calculating the volume of cylindrical bottom [m3]
Vol_total=Vol_top+Vol_cyl      % Calculating the total volume of acetylene bottle [m3]
```

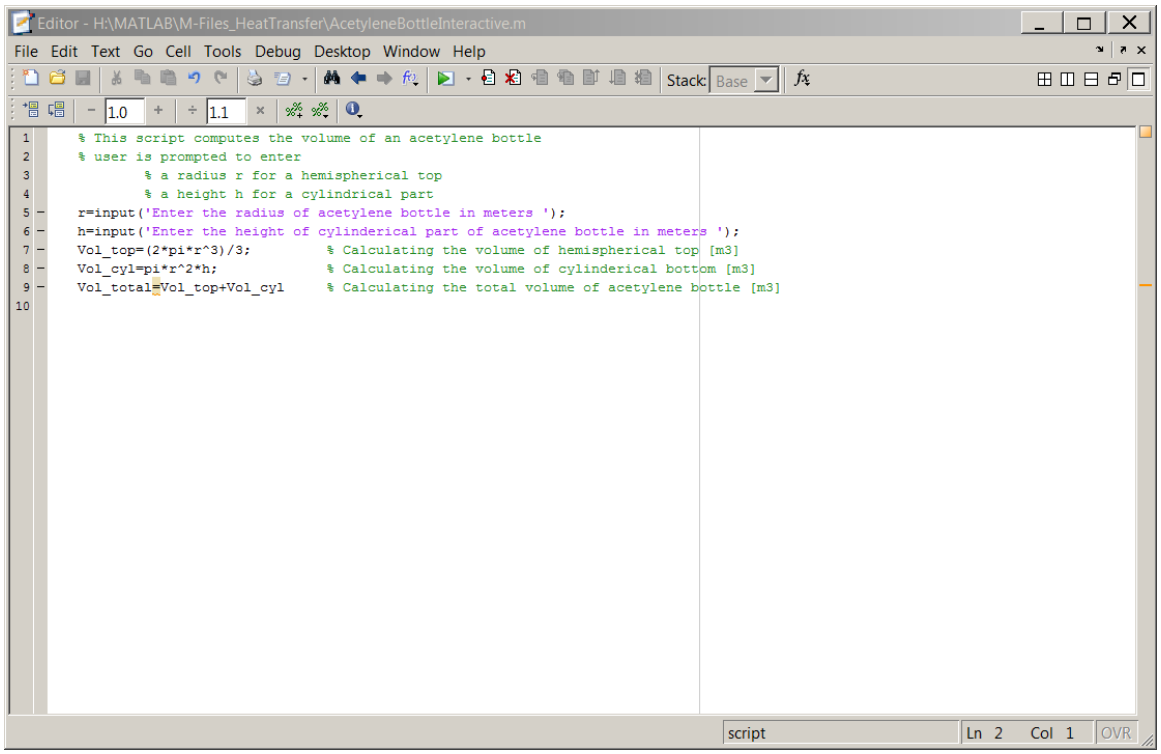


Figure 4.5: Interactive script that computes the volume of acetylene cylinder.

The command window upon run will be as follows, note that user keys in the radius and height values and the same input values result in the same numerical answer as in example (Example 4.1) which proves that the computation is correct.

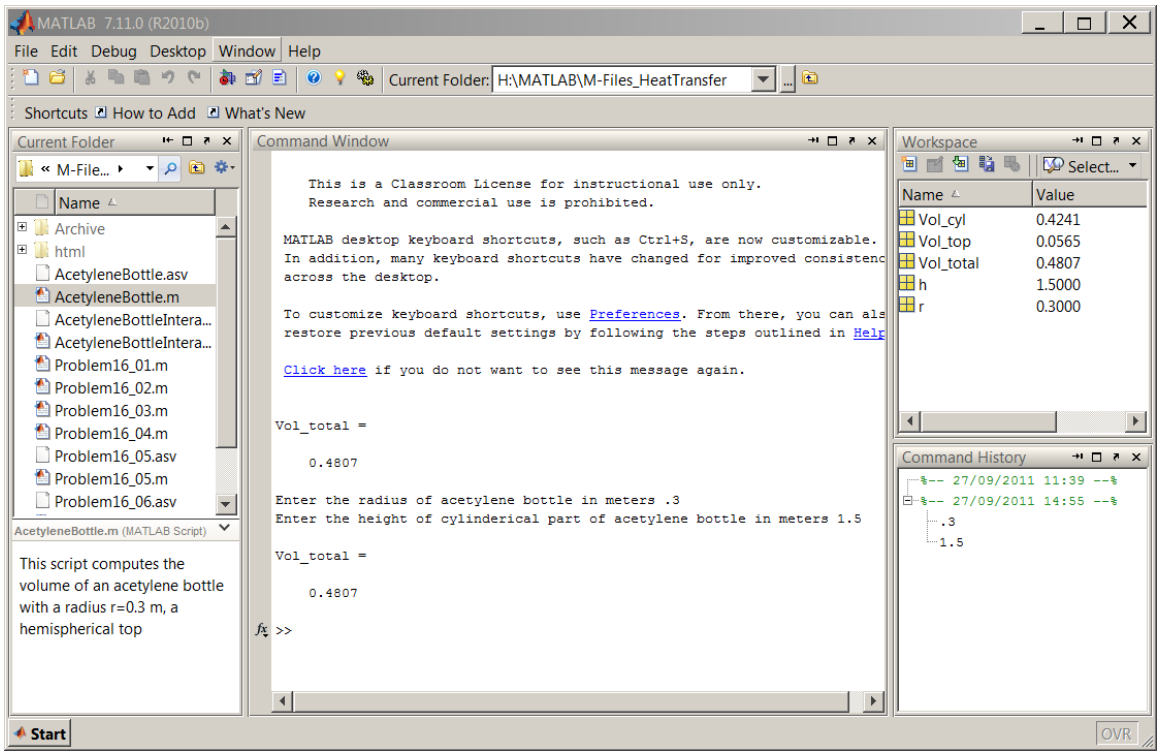


Figure 4.6: The same numerical result is obtained through interactive script.

4.1.3 The `disp` Function

As you might have noticed, the output of our script is not displayed in a well-formatted fashion. Using `disp`, we can control how text or arrays are displayed in the command window. For example, to display a text string on the screen, type in `disp('Hello world!')`. This command will return our friendly greeting as follows: `Hello world!`

`disp(variable)` can be used to display only the value of a variable. To demonstrate this, issue the following command in the command window:

```
b = [1 2 3 4 5]
```

We have created a row vector with 5 elements. The following is displayed in the command window:

```
≫ b = [1 2 3 4 5]

b =

     1     2     3     4     5
```

Now if we type in `disp(b)` and press enter, the variable name will not be displayed but its value will be printed on the screen:

```
≫ disp(b)
 1     2     3     4     5
```

The following example demonstrates the usage of `disp` function.

Example 4.3
Now, let's open `AcetyleneBottleInteractive.m` file and modify it by using the `disp` command. First save the file as `AcetyleneBottleInteractiveDisp.m`, so that we don't accidentally introduce errors to a working file and also we can easily find this particular file that utilizes the `disp` command in the future. The new file should contain the code below:

```
% This script computes the volume of an acetylene bottle
% user is prompted to enter
    % a radius r for a hemispherical top
    % a height h for a cylindrical part
clc                          % Clear screen
disp('This script computes the volume of an acetylene bottle')
r=input('Enter the radius of acetylene bottle in meters ');
h=input('Enter the height of cylindrical part of acetylene bottle in meters ');
Vol_top=(2*pi*r^3)/3;        % Calculating the volume of hemispherical top [m3]
Vol_cyl=pi*r^2*h;            % Calculating the volume of cylindrical bottom [m3]
Vol_total=Vol_top+Vol_cyl;   % Calculating the total volume of acetylene bottle [m3]
disp(' ')                    % Display blank line
disp('The volume of the acetylene bottle is') % Display text
disp(Vol_total)              % Display variable
```

Your screen output should look similar to the one below:

```
This script computes the volume of an acetylene bottle
Enter the radius of acetylene bottle in meters .3
Enter the height of cylindrical part of acetylene bottle in meters 1.5

The volume of the acetylene bottle is
    0.4807
```

4.1.4 The num2str Function

The `num2str` function allows us to convert a number to a text string. Basic syntax is `str = num2str(A)` where variable A is converted to a text and stored in `str`. Let's see how it works in `AcetyleneBottleInteractiveDisp.m`. Remember to save the file with a different name before editing it, for example, `AcetyleneBottleInteractiveDisp1.m`.

Example 4.4
Add the following line of code to your file:

```
str = ['The volume of the acetylene bottle is ', num2str(Vol_total), ' cubic meters.'];
```

Notice that the three arguments in `str` are separated with commas. The first argument is a simple text that is contained in ' '. The second argument is where the number to string conversion take place. And finally the third argument is also a simple text that completes the sentence displayed on the screen. Using semicolon at the end of the line suppresses the output. In the next line of our script, we will call `str` with `disp(str);`.

AcetyleneBottleInteractiveDisp1.m file should look like this:

```
    % This script computes the volume of an acetylene bottle
  % user is prompted to enter
        % a radius r for a hemispherical top
        % a height h for a cylindrical part
clc                          % Clear screen
disp('This script computes the volume of an acetylene bottle:')
disp(' ')                    % Display blank line
r=input('Enter the radius of acetylene bottle in meters ');
h=input('Enter the height of cylindrical part of acetylene bottle in meters ');
Vol_top=(2*pi*r^3)/3;        % Calculating the volume of hemispherical top [m3]
Vol_cyl=pi*r^2*h;            % Calculating the volume of cylindrical bottom [m3]
Vol_total=Vol_top+Vol_cyl;   % Calculating the total volume of acetylene bottle [m3]
disp(' ')                    % Display blank line
str = ['The volume of the acetylene bottle is ', num2str(Vol_total), ' cubic meters.'];
disp(str);
```

Running the script should produce the following:

```
This script computes the volume of an acetylene bottle:

Enter the radius of acetylene bottle in meters .3
Enter the height of cylindrical part of acetylene bottle in meters 1.5

The volume of the acetylene bottle is 0.48066 cubic meters.
```

4.1.5 The `fopen` and `fclose` Functions

The first command is used to open or create a file. The basic syntax for `fopen` is as follows:

```
fid = fopen(filename, permission)
```

For example, `fo = fopen('output.txt', 'w');` opens or creates a new file named `output.txt` and sets the permission for writing. If the file already exists, it discards the existing contents.

`fclose` command is used to close a file. For example, if we type in `fclose(fo);`, we close the file that was created above.

4.1.6 The `fprintf` Function

`fprintf` function writes formatted data to the computer monitor or a file. This command can be used to save the results of a calculation to a file. To do this, first we create or open an output file with `fopen`, second we issue the `fprintf` command and then we close the output file with `fclose`.

The simplified syntax for `fprintf` is as follows:

```
fprintf=(fid, format, variable1, variable 2, ...)
```

Example 4.5

Add the following lines to your .m file:

```
    fo = fopen('output.txt', 'w');
fprintf(fo,'The radius of acetylene bottle: %g meters \n', r);
fprintf(fo,'The height of cylindrical part of acetylene bottle: %g meters \n', h);
fprintf(fo,'The volume of the acetylene bottle: %g cubic meters. \n', Vol_total);
fclose(fo);
```

Here, we first create the `output.txt` file that will contain the following three variables `r, h` and `Vol_total`. In the `fo` output file, the variables are formated with `%g` which automatically uses the shortest display format. You can also use `%i` or `%d` for integers and `%e` for scientific notation. In our script above, the `\n` (newline) moves the cursor to the next line.

Naming the new .m file as `AcetyleneBottleInteractiveOutput.m`, it should look like this:

```
    % This script computes the volume of an acetylene bottle
  % user is prompted to enter
        % a radius r for a hemispherical top
        % a height h for a cylindrical part
clc                          % Clear screen
disp('This script computes the volume of an acetylene bottle:')
disp(' ')                    % Display blank line
r=input('Enter the radius of acetylene bottle in meters ');
h=input('Enter the height of cylindrical part of acetylene bottle in meters ');
Vol_top=(2*pi*r^3)/3;        % Calculating the volume of hemispherical top [m3]
Vol_cyl=pi*r^2*h;            % Calculating the volume of cylindrical bottom [m3]
Vol_total=Vol_top+Vol_cyl;   % Calculating the total volume of acetylene bottle [m3]
disp(' ')                    % Display blank line
str = ['The volume of the acetylene bottle is ', num2str(Vol_total), ' cubic meters.'];
disp(str);
fo = fopen('output.txt', 'w');
fprintf(fo,'The radius of acetylene bottle: %g meters \n', r);
fprintf(fo,'The height of cylindrical part of acetylene bottle: %g meters \n', h);
fprintf(fo,'The volume of the acetylene bottle: %g cubic meters. \n', Vol_total);
fclose(fo);
```

Upon running the file, the `output.txt` file will display the following:

```
    The radius of acetylene bottle: 0.3 meters
The height of cylindrical part of acetylene bottle: 1.5 meters
The volume of the acetylene bottle: 0.480664 cubic meters.
```

4.1.7 Loops

In programming, a loop executes a set of code a specified number of times or until a condition is met.

4.1.7.1 For Loop

This loop iterates an index variable from an initial value using a specified increment to a final value and runs a set of code. The for loop syntax is the following:

```
for loop_index=vector_statement
  code
  ...
  code
end
```

Example 4.6
Calculate $y = \cos(x)$ for $-\pi \leq x \leq \pi$ using an increment of $\frac{\pi}{4}$.

```
for x=-pi:pi/4:pi
    y=cos(x);
    fprintf('%8.3f %8.2f \n',x,y);
end
```

In the brief script above, x is the loop index that is initiated from $-\pi$ and incremented with $\frac{\pi}{4}$ to a final value of π. At the end of each increment, $y = \cos(x)$ is calculated and displayed with the fprintf command. This process continues until $x = \pi$.

From a previous exercise we know \n creates a new line when included in the fprintf command. Here, we also use %8.3f to specify eight spaces and three decimal places for the first variable x. Likewise %8.2f specifies the formatting for the second variable y but in this case, y is displayed with two decimal places. The result is the following:

```
  -3.142    -1.00
  -2.356    -0.71
  -1.571     0.00
  -0.785     0.71
   0.000     1.00
   0.785     0.71
   1.571     0.00
   2.356    -0.71
   3.142    -1.00
```

We can improve our code by adding formatting lines as follows:

```
clear; clc;
fprintf('   x         cos(x)\n') % title row
fprintf('   ----------------\n') % title row
 for x=-pi:pi/4:pi       % loop_index=inital_value:increment_value:final_value
    y=cos(x);           % code to calculate cos(x)
    fprintf('%8.3f %8.2f \n',x,y); % code to print the output to screen
 end
```

Screen output:

```
   x         cos(x)
   ----------------
  -3.142    -1.00
  -2.356    -0.71
  -1.571     0.00
  -0.785     0.71
   0.000     1.00
   0.785     0.71
   1.571     0.00
   2.356    -0.71
   3.142    -1.00
```

4.1.7.2 While Loop

Like the for loop, a while loop executes blocks of code over and over again however it runs as long as the test condition remains true. The syntax of a while loop is

```
while test_condition
 code
 ...
 code
end
```

Example 4.7
Using a `while` loop, calculate $y = \cos(x)$ for $-\pi \leq x \leq \pi$ using an increment of $\frac{\pi}{4}$.

This time we need to initialize the x value outside the loop and then state the test condition in the first line of the while loop. We also need to create an increment statement within the while loop:

```
x=-pi;
    while x<=pi
        y=cos(x);
        fprintf('%8.3f %8.2f \n',x,y);
        x = x + (pi/4);
    end
```

The result is the same as that of the previous example:

```
  -3.142    -1.00
  -2.356    -0.71
  -1.571     0.00
  -0.785     0.71
   0.000     1.00
   0.785     0.71
   1.571     0.00
   2.356    -0.71
   3.142    -1.00
```

Now we can improve the code by adding extra formatting lines and comments:

```
clear; clc;
fprintf('   x         cos(x)\n') % title row
fprintf('   ----------------\n') % title row
x=-pi;                          % initiating the x value
while x<=pi                    % stating the test condition
    y=cos(x);                   % calculating the value of y
    fprintf('%8.3f %8.2f \n',x,y); % printing a and y
    x = x + (pi/4);      % iterating to the next step
end
```

The result should look the same as before.

```
   x         cos(x)
   ----------------
  -3.142    -1.00
  -2.356    -0.71
  -1.571     0.00
  -0.785     0.71
   0.000     1.00
   0.785     0.71
   1.571     0.00
   2.356    -0.71
   3.142    -1.00
```

4.1.8 The `diary` Function

Instead of writing a script from scratch, we sometimes solve problems in the Command Window as if we are using a scientific calculator. The steps we perform in this fashion can be used to create an m-file. For example, the `diary` function allows us to record a MATLAB session in a file and retrieve it for review. Reviewing the file and by copying relevant parts of it and pasting them in to an m-file, a script can be written easily.

Typing `diary` at the MATLAB prompt toggles the diary mode on and off. As soon as the diary mode is turned on, a file called diary is created in the current directory. If you like to save that file with a specific name, say for example problem16, type
≫ `diary problem16.txt`.
A file named problem16.txt will be created. The following is the content of a diary file called problem16.txt. Notice that in that session, the user is executing the four files we created earlier. The user's keyboard input and the resulting display output is recorded in the file. The session is ended by typing `diary` which is printed in the last line. This might be useful to create a record of your work to hand in with a lab or to create the beginnings of an m-file.

```
    AcetyleneBottle

Vol_total =

    0.4807

AcetyleneBottleInteractive
Enter the radius of acetylene bottle in meters .3
Enter the height of cylinderical part of acetylene bottle in meters 1.5

Vol_total =

    0.4807

AcetyleneBottleInteractiveDisp
This script computes the volume of an acetylene bottle
Enter the radius of acetylene bottle in meters .5
Enter the height of cylinderical part of acetylene bottle in meters 1.6

The volume of the acetylene bottle is
```

```
    1.5184

AcetyleneBottleInteractiveDisp1
This script computes the volume of an acetylene bottle:

Enter the radius of acetylene bottle in meters .9
Enter the height of cylinderical part of acetylene bottle in meters 1.9

The volume of the acetylene bottle is 6.3617 cubic meters.
diary
```

4.1.9 Style Guidelines

Try to apply the following guidelines when writing your scripts:

- Share your code or programs with others, consider adopting one of Creative Commons[2] or GNU General Public License[3] schemes
- Include your name and contact info in the opening lines
- Use comments liberally
- Group your code and use proper indentation
- Use white space liberally
- Use descriptive names for your variables
- Use descriptive names for your m-files

4.1.10 Summary of Key Points

1. A script is a file containing a sequence of MATLAB statements. Script files have a filename extension of .m.
2. Functions such as `input`, `disp` and `num2str` can be used to make scripts interactive,
3. `fopen`, `fprintf` and `fclose` functions are used to create output files,
4. A `for` loop is used to repeat a specific block of code a definite number of times.
5. A `while` loop is used to repeat a specific block of code an indefinite number of times, until a condition is met.
6. The `diary` function is useful to record a MATLAB command window session from which an m-file can be easily created,
7. Various style guidelines covered here help improve our code.

4.2 Problem Set[4]

Exercise 4.2.1 *(Solution on p. 97.)*

Write a script that will ask for pressure value in psi and display the equivalent pressure in kPa with a statement, such as "The converted pressure is: ..."

Exercise 4.2.2 *(Solution on p. 97.)*

Write a script that generates a table of conversions from Fahrenheit to Celsius temperatures for a range and increment entered by the user, such as

[2] http://creativecommons.org/
[3] http://www.gnu.org/licenses/gpl-3.0.html
[4] This content is available online at <http://cnx.org/content/m41536/1.3/>.

Enter the beginning temperature in F:
Enter the ending temperature in F:
Enter the increment value:

Test your script with 20 the beginning Fahrenheit value, 200 the ending Fahrenheit value and 20 the increment.

Exercise 4.2.3 *(Solution on p. 98.)*

Pascal's Law states that pressure is transmitted undiminished in all directions throughout a fluid at rest. (See the illustration below). An initial force of 150 N is transmitted from a piston of 25 mm^2 to a piston of 100 mm^2. This force is progressively increased up to 200 N. Write a script that computes the corresponding load carried by the larger piston and tabulate your results.

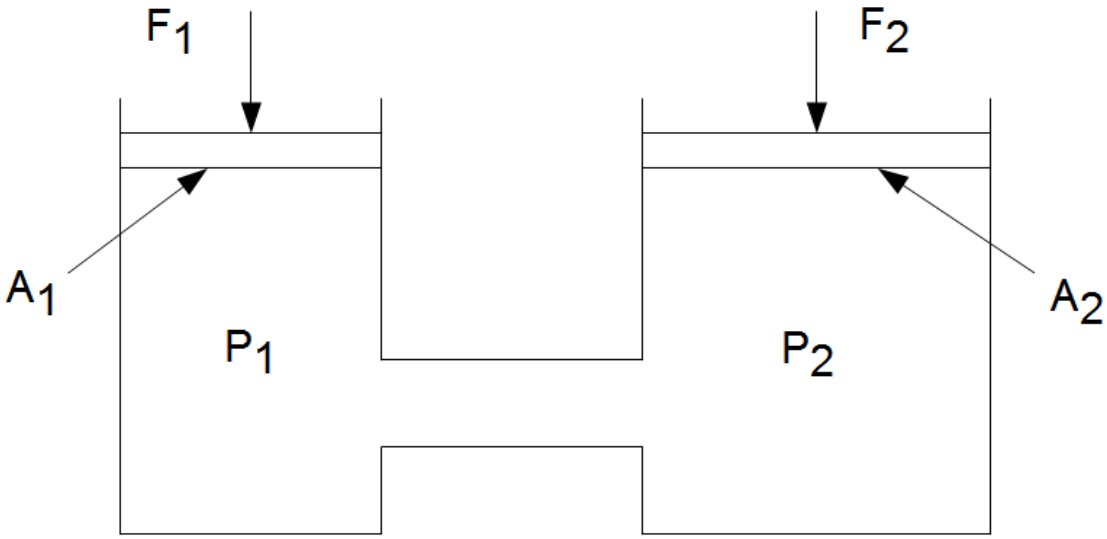


Figure 4.7: A simple hydraulic system.

Exercise 4.2.4 *(Solution on p. 98.)*

Modify your script in previous problem (Exercise 4.2.3) so that the user provides the following input:

Enter the initial force in N:
Enter the final force in N:
Enter the increment value:
Enter the area of small piston in mm^2:
Enter the area of big piston in mm^2:

Test your script with 150, 200, 10, 25 and 100 with respect to each input variable.

Exercise 4.2.5 *(Solution on p. 99.)*

Write a script to solve the Stress-Strain problem in the Problem Set (Problem 2.2.9)

Exercise 4.2.6 *(Solution on p. 100.)*
Modify the script, you wrote above (Exercise 4.2.5)and plot an annotated Stress-Strain graph.

Exercise 4.2.7 *(Solution on p. 102.)*
Repeat Problem 2 (Exercise 4.2.2), this time using a combination of `disp`, `fprintf` commands and a `for` loop.

Exercise 4.2.8 *(Solution on p. 103.)*
Repeat Problem 7 (Exercise 4.2.7), this time using a `while` loop.

Solutions to Exercises in Chapter 4

Solution to Exercise 4.2.1 (p. 94)

```
% This script converts pressures from psi to kPa
% User is prompted to enter pressure in psi
clc                          % Clear screen
disp('This script converts pressures from psi to kPa:')
disp(' ')                    % Display blank line
psi=input('What is the pressure value in psi? ');
kPa=psi*6.894757;            % Calculating pressure in kPa
disp(' ')                    % Display blank line
str = ['The converted pressure is: ', num2str(kPa), ' kPa.'];
disp(str);
```

The script output is as follows:

```
This script converts pressures from psi to kPa:

What is the pressure value in psi? 150

The converted pressure is: 1034.2135 kPa.
```

Solution to Exercise 4.2.2 (p. 94)

```
    % This script generates a table of conversions
% From Fahrenheit to Celsius temperatures
clc                          % Clear screen
disp('This script generates a table of conversions from Fahrenheit to Celsius')
disp(' ')                    % Display blank line
lowerF=input('Enter the beginning temperature in F: ');
upperF=input('Enter the ending temperature in F: ');
increment=input('Enter the increment value: ');
Fahrenheit=[lowerF:increment:upperF]; % Creating a row vector with F values
Celsius=5/9*(Fahrenheit-32);    % Converting from F to C
disp(' ')                    % Display blank line
str = ['Fahrenheit   Celsius '];% Displaying table header
disp(str);
% Tabulating results in two columns, ' is being used to transpose row to column
disp([Fahrenheit' Celsius'])
```

The script output is as follows:

```
This script generates a table of conversions from Fahrenheit to Celsius

Enter the beginning temperature in F: 20
Enter the ending temperature in F: 200
Enter the increment value: 20
```

```
Fahrenheit   Celsius
   20.0000   -6.6667
   40.0000    4.4444
   60.0000   15.5556
   80.0000   26.6667
  100.0000   37.7778
  120.0000   48.8889
  140.0000   60.0000
  160.0000   71.1111
  180.0000   82.2222
  200.0000   93.3333
```

Solution to Exercise 4.2.3 (p. 95)

```
% This script computes the load carried by the larger piston in a hydraulic system
clc                          % Clear screen
disp('This script computes the load carried by the larger piston in a hydraulic system')
disp(' ')                    % Display blank line
initialF=150;
finalF=200;
increment=10;
area1=25;
area2=100;
F1=[initialF:increment:finalF]; % Creating a row vector with F1 values
F2=F1*area2/area1; % Calculating F2 values
disp(' ')                    % Display blank line
str = ['    F1   F2 '];% Displaying table header
disp(str);
disp([F1' F2']) % Tabulating results in two columns, ' is being used to transpose row to column
```

The script output is as follows:

```
This script computes the load carried by the larger piston in a hydraulic system


    F1   F2
   150   600
   160   640
   170   680
   180   720
   190   760
   200   800
```

Solution to Exercise 4.2.4 (p. 95)

```
% This script computes the load carried by the larger piston in a hydraulic system
clc                          % Clear screen
disp('This script computes the load carried by the larger piston in a hydraulic system')
disp(' ')                    % Display blank line
```

```
initialF=input('Enter the initial force in N: ');
finalF=input('Enter the final force in N: ');
increment=input('Enter the increment value: ');
area1=input('Enter the area of small piston in mm^2: ');
area2=input('Enter the area of big piston in mm^2: ');
F1=[initialF:increment:finalF]; % Creating a row vector with F1 values
F2=F1*area2/area1; % Calculating F2 values
disp(' ')                       % Display blank line
str = ['    F1   F2 '];% Displaying table header
disp(str);
disp([F1' F2']) % Tabulating results in two columns, ' is being used to transpose row to column
```

The script output is as follows:

```
This script computes the load carried by the larger piston in a hydraulic system

Enter the initial force in N: 150
Enter the final force in N: 200
Enter the increment value: 10
Enter the area of small piston in mm^2: 25
Enter the area of big piston in mm^2: 100

    F1   F2
   150   600
   160   640
   170   680
   180   720
   190   760
   200   800
```

Solution to Exercise 4.2.5 (p. 95)

The m-file contains the following:

```
    % This script uses readings from a Tensile test and
% Computes Strain and Stress values
clc                             % Clear screen
disp('This script uses readings from a Tensile test and')
disp('Computes Strain and Stress values')
disp(' ')                       % Display a blank line
Specimen_dia=12.7; % Specimen diameter in mm
% Load in kN
Load_kN=[0;4.89;9.779;14.67;19.56;24.45;...
    27.62;29.39;32.68;33.95;34.58;35.22;...
    35.72;40.54;48.39;59.03;65.87;69.42;...
    69.67;68.15;60.81];
% Gage length in mm
Length_mm=[50.8;50.8102;50.8203;50.8305;...
    50.8406;50.8508;50.8610;50.8711;...
    50.9016;50.9270;50.9524;50.9778;...
    51.0032;51.816;53.340;55.880;58.420;...
    60.96;61.468;63.5;66.04];
```

```
% Calculate x-sectional area im m2
Cross_sectional_Area=pi/4*((Specimen_dia/1000)^2);
% Calculate change in length, initial lenght is 50.8 mm
Delta_L=Length_mm-50.8;
% Calculate Stress in MPa
Sigma=(Load_kN./Cross_sectional_Area)*10^(-3);
% Calculate Strain in mm/mm
Epsilon=Delta_L./50.8;
str = ['Specimen diameter is ', num2str(Specimen_dia), ' mm.'];
disp(str);
Results=[Load_kN Length_mm Delta_L Sigma Epsilon];
% Tabulated results
disp('      Load   Length     Delta L  Stress     Strain')
disp(Results)
```

After executed, the command window output is:

```
This script uses readings from a Tensile test and
Computes Strain and Stress values

Specimen diameter is 12.7 mm.
      Load   Length     Delta L  Stress     Strain
         0   50.8000         0         0         0
    4.8900   50.8102    0.0102   38.6022    0.0002
    9.7790   50.8203    0.0203   77.1964    0.0004
   14.6700   50.8305    0.0305  115.8065    0.0006
   19.5600   50.8406    0.0406  154.4086    0.0008
   24.4500   50.8508    0.0508  193.0108    0.0010
   27.6200   50.8610    0.0610  218.0351    0.0012
   29.3900   50.8711    0.0711  232.0076    0.0014
   32.6800   50.9016    0.1016  257.9792    0.0020
   33.9500   50.9270    0.1270  268.0047    0.0025
   34.5800   50.9524    0.1524  272.9780    0.0030
   35.2200   50.9778    0.1778  278.0302    0.0035
   35.7200   51.0032    0.2032  281.9773    0.0040
   40.5400   51.8160    1.0160  320.0269    0.0200
   48.3900   53.3400    2.5400  381.9955    0.0500
   59.0300   55.8800    5.0800  465.9888    0.1000
   65.8700   58.4200    7.6200  519.9844    0.1500
   69.4200   60.9600   10.1600  548.0085    0.2000
   69.6700   61.4680   10.6680  549.9820    0.2100
   68.1500   63.5000   12.7000  537.9830    0.2500
   60.8100   66.0400   15.2400  480.0403    0.3000
```

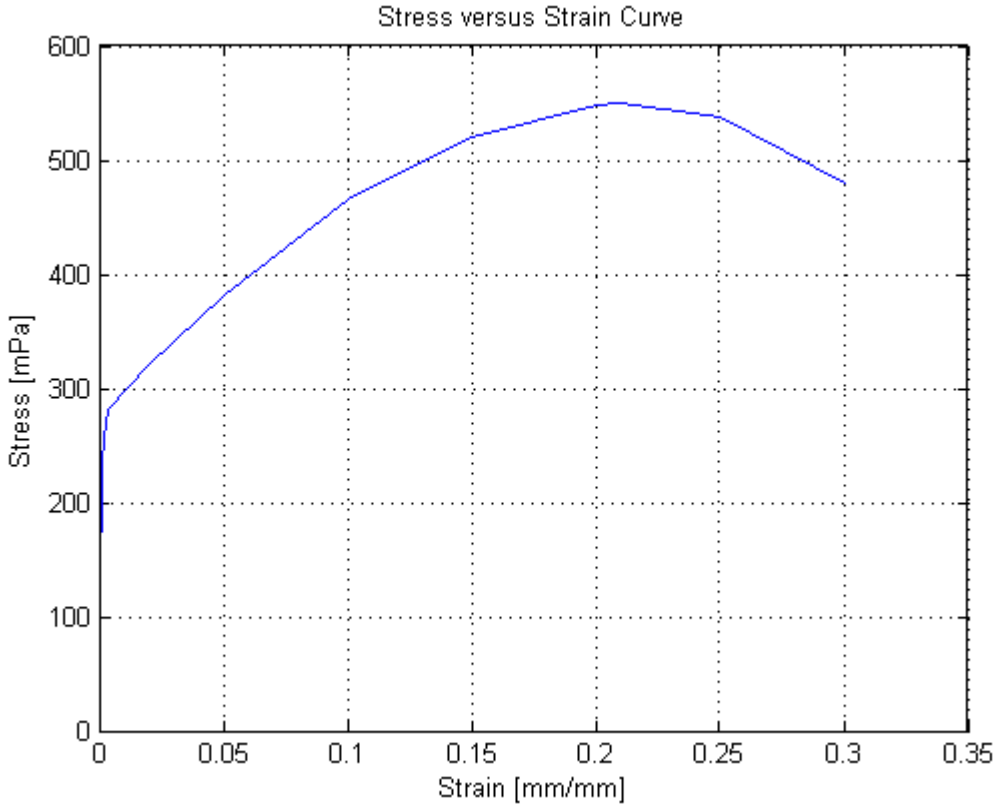
Solution to Exercise 4.2.6 (p. 96)

Edited script contains the plot commands:

```
% This script uses readings from a Tensile test and
% Computes Strain and Stress values
clc                          % Clear screen
disp('This script uses readings from a Tensile test and')
```

```
disp('Computes Strain and Stress values')
disp(' ')                      % Display a blank line
Specimen_dia=12.7; % Specimen diameter in mm
% Load in kN
Load_kN=[0;4.89;9.779;14.67;19.56;24.45;...
    27.62;29.39;32.68;33.95;34.58;35.22;...
    35.72;40.54;48.39;59.03;65.87;69.42;...
    69.67;68.15;60.81];
% Gage length in mm
Length_mm=[50.8;50.8102;50.8203;50.8305;...
    50.8406;50.8508;50.8610;50.8711;...
    50.9016;50.9270;50.9524;50.9778;...
    51.0032;51.816;53.340;55.880;58.420;...
    60.96;61.468;63.5;66.04];

% Calculate x-sectional area im m2
Cross_sectional_Area=pi/4*((Specimen_dia/1000)^2);
% Calculate change in length, initial lenght is 50.8 mm
Delta_L=Length_mm-50.8;
% Calculate Stress in MPa
Sigma=(Load_kN./Cross_sectional_Area)*10^(-3);
% Calculate Strain in mm/mm
Epsilon=Delta_L./50.8;
str = ['Specimen diameter is ', num2str(Specimen_dia), ' mm.'];
disp(str);
Results=[Load_kN Length_mm Delta_L Sigma Epsilon];
% Tabulated results
disp('      Load   Length     Delta L  Stress     Strain')
disp(Results)
% Plot Stress versus Strain
plot(Epsilon,Sigma)
title('Stress versus Strain Curve')
xlabel('Strain [mm/mm]')
ylabel('Stress [mPa]')
grid
```

In addition to Command Window output, the following plot is generated:



Solution to Exercise 4.2.7 (p. 96)
The re-worked solution:

```
% This script generates a table of conversions
% From Fahrenheit to Celsius temperatures
clear   % removes all variables from the current workspace,
clc     % clears all input and output from the Command Window display,
disp('This script generates a table of conversions from Fahrenheit to Celsius')
disp(' ')                       % Display blank line
lowerF=input('Enter the initial temperature in F: ');
upperF=input('Enter the final temperature in F: ');
increment=input('Enter the increment value: ');
disp(' ')                       % Display blank line
fprintf('Fahrenheit Celsius\n') % title row
fprintf('-----------------\n') % title row
for Fahrenheit=[lowerF:increment:upperF]; % Creating a row vector with F values
    Celsius=5/9*(Fahrenheit-32);    % Converting from F to C
    fprintf('%8.3f %8.3f \n',Fahrenheit,Celsius);  % Tabulating results in two columns
end
```

After executed, the command window output is:

```
This script generates a table of conversions from Fahrenheit to Celsius
```

```
Enter the initial temperature in F: 20
Enter the final temperature in F: 200
Enter the increment value: 20

Fahrenheit Celsius
------------------
  20.000   -6.667
  40.000    4.444
  60.000   15.556
  80.000   26.667
 100.000   37.778
 120.000   48.889
 140.000   60.000
 160.000   71.111
 180.000   82.222
 200.000   93.333
```

Solution to Exercise 4.2.8 (p. 96)
The re-worked solution:

```
% This script generates a table of conversions
% From Fahrenheit to Celsius temperatures
clear   % removes all variables from the current workspace,
clc     % clears all input and output from the Command Window display,
disp('This script generates a table of conversions from Fahrenheit to Celsius')
disp(' ')                     % Display blank line
lowerF=input('Enter the initial temperature in F: ');
upperF=input('Enter the final temperature in F: ');
increment=input('Enter the increment value: ');
disp(' ')                     % Display blank line
fprintf('Fahrenheit Celsius\n') % title row
fprintf('------------------\n') % title row
Fahrenheit=lowerF;
    while Fahrenheit<=upperF
        Celsius=5/9*(Fahrenheit-32);    % Converting from F to C
        fprintf('%8.3f %8.3f \n',Fahrenheit,Celsius);  % Tabulating results in two columns
        Fahrenheit=Fahrenheit+increment;
    end
```

After executed, the command window output is:

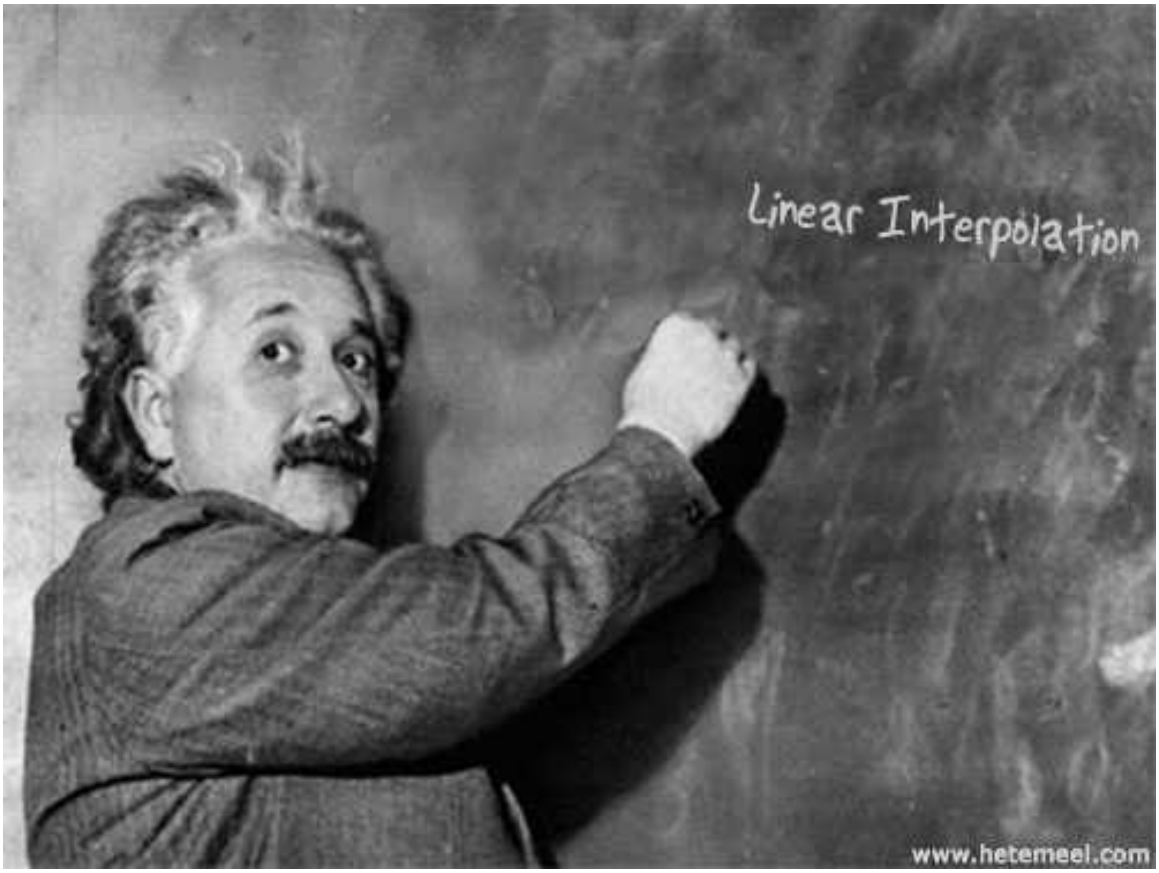
```
This script generates a table of conversions from Fahrenheit to Celsius

Enter the initial temperature in F: 20
Enter the final temperature in F: 200
Enter the increment value: 20

Fahrenheit Celsius
------------------
  20.000   -6.667
  40.000    4.444
```

```
 60.000   15.556
 80.000   26.667
100.000   37.778
120.000   48.889
140.000   60.000
160.000   71.111
180.000   82.222
200.000   93.333
```

Chapter 5

Interpolation

5.1 Interpolation[1]



Linear interpolation is one of the most common techniques for estimating values between two given data points. For example, when using steam tables, we often have to carry out interpolations. With this technique, we assume that the function between the two points is linear. MATLAB has a built-in interpolation function.

[1] This content is available online at <http://cnx.org/content/m41455/1.3/>.

5.1.1 The `interp1` Function

Give an x-y table, `y_new = interp1(x,y,x_new)` interpolates to find y_new. Consider the following examples:

Example 5.1
To demonstrate how the `interp1` function works, let us create an x-y table with the following commands;

```
x = 0:5;
y = [0,20,60,68,77,110];
```

To tabulate the output, we can use

```
[x',y']
```

The result is

```
ans =

     0     0
     1    20
     2    60
     3    68
     4    77
     5   110
```

Suppose we want to find the corresponding value for 1.5 or interpolate for 1.5. Using `y_new = interp1(x,Y,x_new)` syntax, let us type in:

```
y_new=interp1(x,y,1.5)

y_new =

    40
```

Example 5.2
The table we created above has only 6 elements in it and suppose we need a more detailed table. In order to do that, instead of a single new x value, we can define an array of new x values, the `interp1` function returns an array of new y values:

```
new_x = 0:0.2:5;
new_y = interp1(x,y,new_x);
```

Let's display this table

```
[new_x',new_y']
```

The result is

```
ans =

         0         0
    0.2000    4.0000
    0.4000    8.0000
    0.6000   12.0000
    0.8000   16.0000
    1.0000   20.0000
    1.2000   28.0000
    1.4000   36.0000
    1.6000   44.0000
    1.8000   52.0000
    2.0000   60.0000
    2.2000   61.6000
    2.4000   63.2000
    2.6000   64.8000
    2.8000   66.4000
    3.0000   68.0000
    3.2000   69.8000
    3.4000   71.6000
    3.6000   73.4000
    3.8000   75.2000
    4.0000   77.0000
    4.2000   83.6000
    4.4000   90.2000
    4.6000   96.8000
    4.8000  103.4000
    5.0000  110.0000
```

Example 5.3
Using the table below, find the internal energy of steam at 215 °C and the temperature if the internal energy is 2600 kJ/kg (use linear interpolation).

Temperature [C]	**Internal Energy [kJ/kg]**
100	2506.7
150	2582.8
200	2658.1
250	2733.7
300	2810.4
400	2967.9
500	3131.6

Table 5.1: An extract from Steam Tables

First let us enter the temperature and energy values

```
    temperature = [100, 150, 200, 250, 300, 400, 500];
energy = [2506.7, 2582.8, 2658.1, 2733.7, 2810.4, 2967.9, 3131.6];
[temperature',energy']
```

returns

```
ans =

  1.0e+003 *

    0.1000    2.5067
    0.1500    2.5828
    0.2000    2.6581
    0.2500    2.7337
    0.3000    2.8104
    0.4000    2.9679
    0.5000    3.1316
```

issue the following for the first question:

```
 new_energy = interp1(temperature,energy,215)
```

returns

```
new_energy =

  2.6808e+003
```

Now, type in the following for the second question:

```
new_temperature = interp1(energy,temperature,2600)
```

returns

```
new_temperature =

  161.4210
```

5.1.2 Summary of Key Points

1. Linear interpolation is a technique for estimating values between two given data points,
2. Problems involving steam tables often require interpolated data,
3. MATLAB has a built-in interpolation function.

5.2 Problem Set[2]

Exercise 5.2.1 *(Solution on p. 111.)*

Determine the saturation temperature, specific liquid enthalpy, specific enthalpy of evaporation and specific enthalpy of dry steam at a pressure of 2.04 MPa.

[2] This content is available online at <http://cnx.org/content/m41624/1.2/>.

Pressure [MN/m^2]	Saturation Temperature [C]	h_f [kJ/kg]	h_{fg} [kJ/kg]	h_g [kJ/kg]
2.1	214.9	920.0	1878.2	2798.2
2.0	212.4	908.6	1888.6	2797.2

Table 5.2: An extract from steam tables

Exercise 5.2.2 *(Solution on p. 111.)*

The following table gives data for the specific heat as it changes with temperature for a perfect gas. (Data available for download[3]). [4]

Temperature [F]	Specific Heat [BTU/lbmF]
25	0.118
50	0.120
75	0.123
100	0.125
125	0.128
150	0.131

Table 5.3: Change of specific heat with temperature

Using interp1 function calculate the specific heat for 30 F, 70 F and 145 F.

Exercise 5.2.3 *(Solution on p. 112.)*

For the problem above (Exercise 5.2.2), create a more detailed table in which temperature varies between 25 and 150 with 5 F increments and corresponding specific heat values.

Exercise 5.2.4 *(Solution on p. 113.)*

During a 12-hour shift a fuel tank has varying levels due to consumption and transfer pump automatically cutting in and out to maintain a safe fuel level. The following table of fuel tank level versus time (Data available for download[5]) is missing readings for 5 and 9 AM. Using linear interpolation, estimate the fuel level at those times.

[3] See the file at <http://cnx.org/content/m41624/latest/Chp5_Exercise2.zip>
[4] *Thermodynamics and Heat Power* by Kurt C. Rolle, Pearson Prentice Hall. ©2005, (p.19)
[5] See the file at <http://cnx.org/content/m41624/latest/Chp5_Exercise4.zip>

Time [hours, AM]	Tank level [m]
1:00	1.5
2:00	1.7
3:00	2.3
4:00	2.9
5:00	?
6:00	2.6
7:00	2.5
8:00	2.3
9:00	?
10:00	2.0
11:00	1.8
12:00	1.3

Table 5.4: Fuel tank level versus time

Solutions to Exercises in Chapter 5

Solution to Exercise 5.2.1 (p. 108)
MATLAB solution is as follows;

```
    ≫ pressure=[2.1 2.0];
≫ sat_temp=[214.9 212.4];
≫ h_f=[920 908.6];
≫ h_fg=[1878.2 1888.6];
≫ h_g=[2798.2 2797.2];

≫ sat_temp_new=interp1(pressure,sat_temp,2.04)

sat_temp_new =

  213.4000

≫ h_f_new=interp1(pressure,h_f,2.04)

h_f_new =

  913.1600

≫ h_fg_new=interp1(pressure,h_fg,2.04)

h_fg_new =

  1.8844e+003

≫ h_g_new=interp1(pressure,h_g,2.04)

h_g_new =

  2.7976e+003
```

Solution to Exercise 5.2.2 (p. 109)
MATLAB solution is as follows:

```
    ≫ temperature=[25;50;75;100;125;150]

temperature =

    25
    50
    75
   100
   125
   150

≫ specific_heat=[.118;.120;.123;.125;.128;.131]

specific_heat =
```

```
    0.1180
    0.1200
    0.1230
    0.1250
    0.1280
    0.1310

≫ specific_heatAt30=interp1(temperature,specific_heat,30)

specific_heatAt30 =

    0.1184

≫ specific_heatAt70=interp1(temperature,specific_heat,70)

specific_heatAt70 =

    0.1224

≫ specific_heatAt145=interp1(temperature,specific_heat,145)

specific_heatAt145 =

    0.1304
```

Solution to Exercise 5.2.3 (p. 109)
MATLAB solution is as follows:

```
    ≫ new_temperature=25:5:150;
≫ new_specific_heat=interp1(temperature,specific_heat,new_temperature);
≫ [new_temperature',new_specific_heat']
ans =

   25.0000    0.1180
   30.0000    0.1184
   35.0000    0.1188
   40.0000    0.1192
   45.0000    0.1196
   50.0000    0.1200
   55.0000    0.1206
   60.0000    0.1212
   65.0000    0.1218
   70.0000    0.1224
   75.0000    0.1230
   80.0000    0.1234
   85.0000    0.1238
   90.0000    0.1242
   95.0000    0.1246
  100.0000    0.1250
  105.0000    0.1256
  110.0000    0.1262
```

```
  115.0000    0.1268
  120.0000    0.1274
  125.0000    0.1280
  130.0000    0.1286
  135.0000    0.1292
  140.0000    0.1298
  145.0000    0.1304
  150.0000    0.1310
```

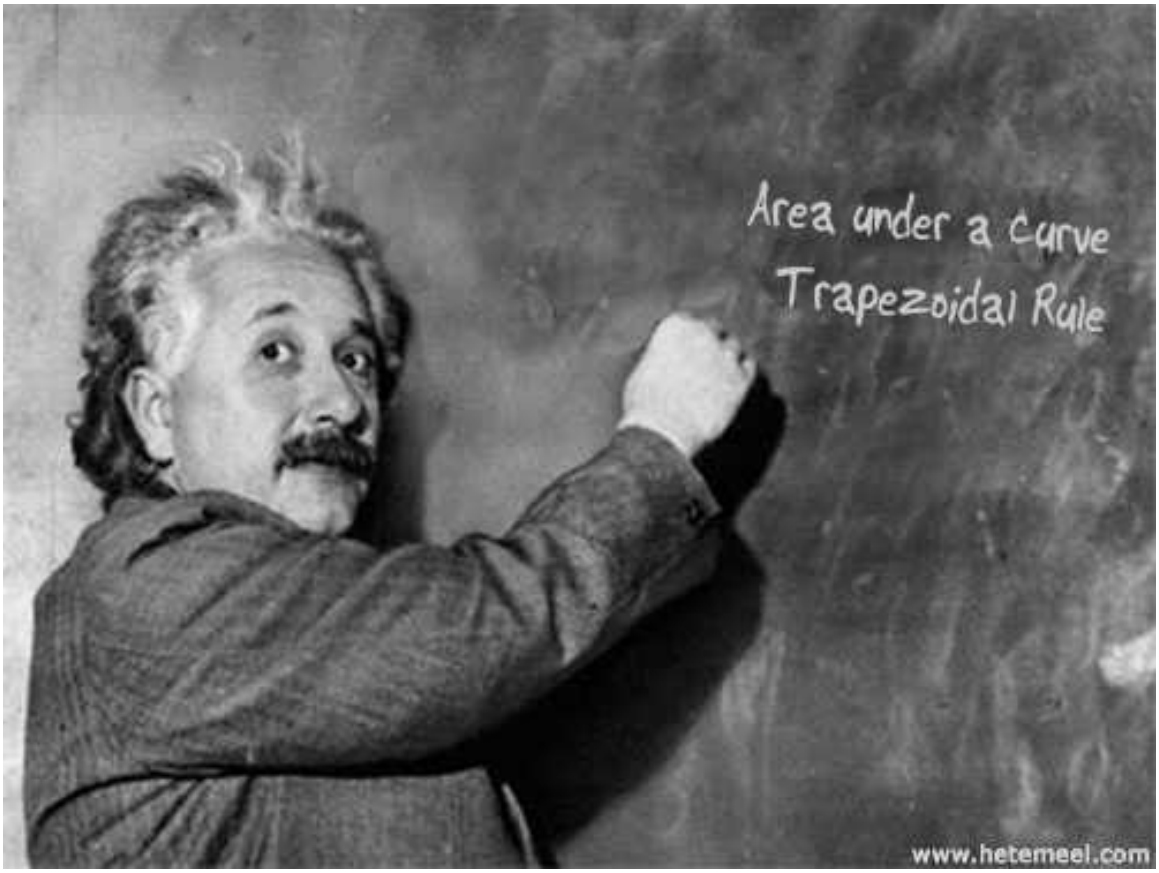
Solution to Exercise 5.2.4 (p. 109)

```
    ≫ time=[1 2 3 4 6 7 8 10 11 12];
≫ tank_level=[1.5 1.7 2.3 2.9 2.6 2.5 2.3 2.0 1.8 1.3];

≫ tank_level_at_5=interp1(time,tank_level,5)

tank_level_at_5 =

    2.7500

≫ tank_level_at_9=interp1(time,tank_level,9)

tank_level_at_9 =

    2.1500
```

Chapter 6

Numerical Integration

6.1 Computing the Area Under a Curve[1]



This chapter essentially deals with the problem of computing the area under a curve. First, we will employ a basic approach and form trapezoids under a curve. From these trapezoids, we can calculate the total area under a given curve. This method can be tedious and is prone to errors, so in the second half of the chapter, we will utilize a built-in MATLAB function to carry out numerical integration.

[1] This content is available online at <http://cnx.org/content/m41454/1.4/>.

6.1.1 A Basic Approach

There are various methods to calculating the area under a curve, for example, Rectangle Method[2] , Trapezoidal Rule[3] and Simpson's Rule[4] . The following procedure is a simplified method.

Consider the curve below:

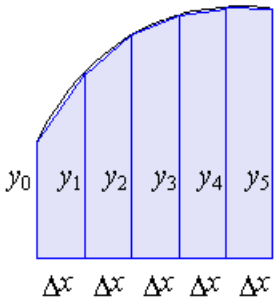


Figure 6.1: Numerical integration

Each segment under the curve can be calculated as follows:

$$\frac{1}{2}(y_0 + y_1)\,\Delta \mathrm{x} + \frac{1}{2}(y_1 + y_2)\,\Delta \mathrm{x} + \frac{1}{2}(y_2 + y_3)\,\Delta \mathrm{x} \tag{6.1}$$

Therefore, if we take the sum of the area of each trapezoid, given the limits, we calculate the total area under a curve. Consider the following example.

Example 6.1

Given the following data, plot an x-y graph and determine the area under a curve between x=3 and x=30

Index	x [m]	y [N]
1	3	27.00
2	10	14.50
3	15	9.40
4	20	6.70
5	25	5.30
6	30	4.50

Table 6.1: Data Set

First, let us enter the data set. For x, issue the following command `x=[3,10,15,20,25,30];`. And for y, `y=[27,14.5,9.4,6.7,5.3,4.5];`. If yu type in `[x',y']`, you will see the following tabulated result. Here we transpose row vectors with ' and displaying them as columns:

[2] http://en.wikipedia.org/wiki/Rectangle_method
[3] http://en.wikipedia.org/wiki/Trapezoidal_rule
[4] http://en.wikipedia.org/wiki/Simpson%27s_rule

```
ans =

 3.0000   27.0000
10.0000   14.5000
15.0000    9.4000
20.0000    6.7000
25.0000    5.3000
30.0000    4.5000
```

Compare the data set above with the given information in the question (Table 6.1).

To plot the data type the following:

```
plot(x,y),title('Distance-Force Graph'),xlabel('Distance[m]'),ylabel('Force[N]'),grid
```

The following figure is generated:

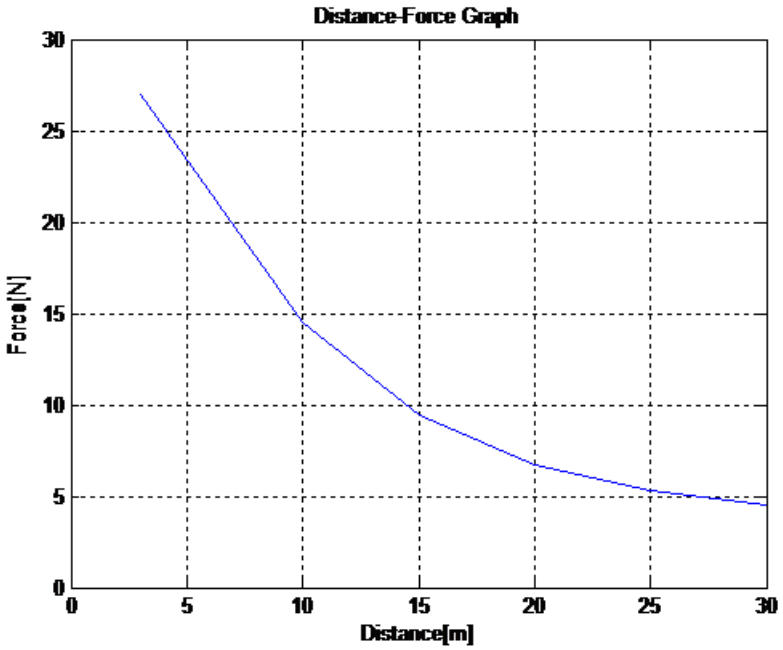


Figure 6.2: Distance-Force Graph

To compute dx for consecutive x values, we will use the index for each x value, see the given data in the question (Table 6.1).:

```
dx=[x(2)-x(1),x(3)-x(2),x(4)-x(3),x(5)-x(4),x(6)-x(5)];
```

dy is computed by the following command:

```
dy=[0.5*(y(2)+y(1)),0.5*(y(3)+y(2)),0.5*(y(4)+y(3)),0.5*(y(5)+y(4)),0.5*(y(6)+y(5))];
```

dx and dy can be displayed with the following command: `[dx',dy']`. The result will look like this:

```
    [dx',dy']

ans =

    7.0000   20.7500
    5.0000   11.9500
    5.0000    8.0500
    5.0000    6.0000
    5.0000    4.9000
```

Our results so far are shown below

x [m]	y [N]	dx [m]	dy [N]
3	27.00		
10	14.50	7.00	20.75
15	9.40	5.00	11.95
20	6.70	5.00	8.05
25	5.30	5.00	6.00
30	4.50	5.00	4.90

Table 6.2: x, y and corresponding differential elements

If we multiply dx by dy, we find da for each element under the curve. The differential area da=dx*dy, can be computed using the 'term by term multiplication' technique in MATLAB as follows:

```
    da=dx.*dy

da =

  145.2500   59.7500   40.2500   30.0000   24.5000
```

Each value above represents an element under the curve or the area of trapezoid. By taking the sum of array elements, we find the total area under the curve.

```
    sum(da)

ans =

  299.7500
```

The following (Table 6.3) illustrates all the steps and results of our MATLAB computation.

x [m]	y [N]	dx [m]	dy [N]	dA [Nm]
3	27.00			
10	14.50	7.00	20.75	145.25
15	9.40	5.00	11.95	59.75
20	6.70	5.00	8.05	40.25
25	5.30	5.00	6.00	30.00
30	4.50	5.00	4.90	24.50
				299.75

Table 6.3: Computation of the approximate area under a curve

6.1.2 The Trapezoidal Rule

Sometimes it is rather convenient to use a numerical approach to solve a definite integral. The trapezoid rule allows us to approximate a definite integral using trapezoids.

6.1.2.1 The `trapz` Command

Z = trapz(Y) computes an approximation of the integral of Y using the trapezoidal method.
Now, let us see a typical problem.

Example 6.2

Given Area = $\int_2^5 x^2 dx$, an analytical solution would produce 39. Use trapz command and solve it

1. Initialize variable x as a row vector, from 2 with increments of 0.1 to 5: `x=2:.1:5;`
2. Declare variable y as `y=x^2;`. Note the following error prompt: `??? Error using ==> mpower Inputs must be a scalar and a square matrix.` This is because x is a vector quantity and MATLAB is expecting a scalar input for y. Because of that, we need to compute y as a vector and to do that we will use the dot operator as follows: `y=x.^2;`. This tells MATLAB to create vector y by taking each x value and raising its power to 2.
3. Now we can issue the following command to calculate the first area, the output will be as follows:

```
area1=trapz(x,y)

area1 =

   39.0050
```

Notice that this numerical value is slightly off. So let us increase the number of increments and calculate the area again:

```
x=2:.01:5;
y=x.^2;
area2=trapz(x,y)

area2 =

   39.0001
```

Yet another increase in the number of increments:

```
x=2:.001:5;
y=x.^2;
area3=trapz(x,y)

area3 =

   39.0000
```

Example 6.3
Determine the value of the following integral:
$\int_0^{\pi} \sin(x)\,dx$

1. Initialize variable x as a row vector, from 0 with increments of pi/100 to pi: `x=0:pi/100:pi;`
2. Declare variable y as `y=sin(x);`
3. Issue the following command to calculate the first area, the output will be as follows:

```
area1=trapz(x,y)

area1 =

    1.9998
```

let us increase the increments as above:

```
x=0:pi/1000:pi;
y=sin(x);
area2=trapz(x,y)

area2 =

    2.0000
```

Example 6.4
A gas expands according to the law, $PV^{1.4}$=c. Initially, the pressure is 100 kPa when the volume is 1 m^3. Write a script to compute the work done by the gas in expanding to three times its original volume[5].

Recall that PV diagrams can be used to estimate the net work performed by a thermodynamic cycle, see Wikipedia[6] or we can use definite integral to compute the work done (WD) as follows:

$$\text{WD} = \int pdv \tag{6.2}$$

If we rearrange the expression pressure as a function of volume, we get:

$$P = \frac{c}{V^{1.4}} \tag{6.3}$$

By considering the initial state, we can determine the value of c:

$$\begin{aligned} c &= 100 \times 1^{1.4} \\ &= 100 \end{aligned} \tag{6.4}$$

[5] *O. N. Mathematics: 2* by J. Dobinson, Penguin Library of Technology. ©1969, (p. 184)
[6] http://en.wikipedia.org/wiki/Pressure_volume_diagram#Thermodynamics

From the equation (6.3) and the equation (6.4) above, we can write:

$$P = \frac{100}{V^{1.4}} \tag{6.5}$$

By inserting P (6.5) in WD (6.2), we get:

$$\text{WD} = \int_1^3 \frac{100}{v^{1.4}} dv \tag{6.6}$$

For MATLAB solution, we will consider P as a function of V (6.5) and WD (6.6). Now, let us apply the three-step approach we have used earlier:

1. Initialize variable volume as a row vector, from 1 with increments of 0.001 to 3: `v=1:0.001:3;`
2. Declare variable pressure as `p=100./v.^1.4;`
3. Use the `trapz` function to calculate the work done, the output will be as follows:

```
WorkDone=trapz(v,p)

WorkDone =

   88.9015
```

These steps can be combined in an m-file as follows:

```
clc
disp('A gas expands according to the law, pv^1.4=C')
disp('Initial pressure is 100 kPa when the volume is 1 m3')
disp('Compute the work done by the gas in expanding')
disp('To three times its original volume')
disp(' ')                      % Display blank line
v=1:.001:3;          % Creating a row vector for volume, v
p=100./(v.^1.4);     % Computing pressure for volume
WorkDone=trapz(v,p) % Integrating p*dv over 1 to 3 cubic meters
```

Example 6.5

A body moves from rest under the action of a direct force given by $F = \frac{15}{x+3}$ where x is the distance in meters from the starting point. Write a script to compute the total work done in moving a distance 10 m.[7]

Recall that the general definition of mechanical work is given by the following integral, see Wikipedia[8] :

$$\text{WD} = \int F dx \tag{6.7}$$

Therefore we can write:

$$\text{WD} = \int_0^{10} \frac{15}{x+3} dx \tag{6.8}$$

Applying the steps we followed in the previous examples, we write:

[7] *O. N. Mathematics: 2* by J. Dobinson, Penguin Library of Technology. ©1969, (p. 183)
[8] http://en.wikipedia.org/wiki/Work_%28physics%29#Force_and_displacement

```
    clc
disp('A body moves from rest under the action of a direct force given')
disp('by F=15/(x+3) where x is the distance in meters')
disp('From the starting point.')
disp('Compute the total work done in moving a distance 10 m.')
disp(' ')                      % Display blank line
x=0:.001:10;     % Creating a row vector for distance, x
F=15./(x+3);    % Computing Force for x
WorkDone=trapz(x,F) % Integrating F*dx over 0 to 10 meters.
```

The output of the above code is:

```
    A body moves from rest under the action of a direct force given
by F=15/(x+3) where x is the distance in meters
From the starting point.
Compute the total work done in moving a distance 10 m.


WorkDone =

   21.9951
```

6.1.3 Summary of Key Points

1. In its simplest form, numerical integration involves calculating the areas of segments that make up the area under a curve,
2. MATLAB has built-in functions to perform numerical integration,
3. `Z = trapz(Y)` computes an approximation of the integral of Y using the trapezoidal method.

6.2 Problem Set[9]

Exercise 6.2.1 ***(Solution on p. 124.)***

Let the function y defined by $y = \cos(x)$. Plot this function over the interval [-pi,pi]. Use numerical integration techniques to estimate the integral of y over [0, pi/2] using increments of pi/10 and pi/1000.

Exercise 6.2.2 ***(Solution on p. 124.)***

Let the function y defined by $y = 0.04x^2 - 2.13x + 32.58$. Plot this function over the interval [3,30]. Use numerical integration techniques to estimate the integral of y over [3,30].

Exercise 6.2.3 ***(Solution on p. 124.)***

A 2000-liter tank is full of lube oil. It is known that if lube oil is drained from the tank, the mass flow rate will decrease from the maximum when the tank level is at the highest. The following data were collected when the tank was drained.

[9] This content is available online at <http://cnx.org/content/m41541/1.9/>.

Time [min]	Mass Flow [kg/min]
0	50.00
5	48.25
10	46.00
15	42.50
20	37.50
25	30.50
30	19.00
35	9.00

Table 6.4: Data

Write a script to estimate the amount of oil drained in 35 minutes.

Exercise 6.2.4 ***(Solution on p. 125.)***

A gas is expanded in an engine cylinder, following the law $PV^{1.3}$=c. The initial pressure is 2550 kPa and the final pressure is 210 kPa. If the volume at the end of expansion is 0.75 m^3, compute the work done by the gas. [10]

Exercise 6.2.5 ***(Solution on p. 126.)***

A force F acting on a body at a distance s from a fixed point is given by $F = 3s + \frac{1}{s^2}$. Write a script to compute the work done when the body moves from the position where s=1 to that where s=10. [11]

Exercise 6.2.6 ***(Solution on p. 126.)***

The pressure p and volume v of a given mass of gas are connected by the relation $\left(p + \frac{a}{v^2}\right)(v - b) = k$ where a, b and k are constants. Express p in terms of v, and write a script to compute the work done by the gas in expanding from an initial volume to a final volume. [12]

Test your solution with the following input:
a: 0.01
b: 0.001
The initial pressure [kPa]: 100
The initial volume [m3]: 1
The final volume [m3]: 2

[10] *Applied Heat for Engineers* by W. Embleton and L Jackson, Thomas Reed Publications. ©1999, (p. 80)
[11] *O. N. Mathematics: 2* by J. Dobinson, Penguin Library of Technology. ©1969, (p. 213)
[12] *O. N. Mathematics: 2* by J. Dobinson, Penguin Library of Technology. ©1969, (p. 212)

Solutions to Exercises in Chapter 6

Solution to Exercise 6.2.1 (p. 122)

1. Plotting:

```
x=-pi:pi/100:pi;
y=cos(x);
plot(x,y),title('Graph of y=cos(x)'),xlabel('x'),ylabel('y'),grid
```

2. Area calculation 1:

```
    ≫ x=0:pi/10:pi/2;
≫ y=cos(x);
≫ area1=trapz(x,y)

area1 =

  0.9918
```

3. Area calculation 2:

```
    ≫ x=0:pi/1000:pi/2;
≫ y=cos(x);
≫ area2=trapz(x,y)

area2 =

  1.0000
```

Solution to Exercise 6.2.2 (p. 122)

1. Plotting:

```
    ≫ x=3:.1:30;
≫ y=0.04*(x.^2)-2.13.*x+32.58;
≫ plot(x,y), title('Graph of ...
y=.04*(x^2)-2.13*x+32.58'),xlabel('x'),ylabel('y'),grid
```

2. Area calculation:

```
≫ area=trapz(x,y)

area =

  290.3868
```

Solution to Exercise 6.2.3 (p. 122)

```
   clc
t=linspace(0,35,8)                  % Data entry for time [min]
m=[50 48.25 46 42.5 37.5 30.5 19 9] % Data entry for mass flow [kg/min]
% Calculate time intervals
dt=[t(2)-t(1),t(3)-t(2),t(4)-t(3),...
```

```
t(5)-t(4),t(6)-t(5),t(7)-t(6),t(8)-t(7)]
% Calculate mass out
dm=[0.5*(m(2)+m(1)),0.5*(m(3)+m(2)),0.5*(m(4)+m(3)),0.5*(m(5)+...
m(4)),0.5*(m(6)+m(5)),0.5*(m(7)+m(6)),0.5*(m(8)+m(7))]
% Calculate differential areas
da=dt.*dm;
% Tabulate time and mass flow
[t',m']
% Tabulate time intervals, mass out and differential areas
[dt',dm',da']
% Calculate the amount of oil drained [kg] in 35 minutes
Oil_Drained=sum(da)
```

The output is:

```
ans =

         0   50.0000
    5.0000   48.2500
   10.0000   46.0000
   15.0000   42.5000
   20.0000   37.5000
   25.0000   30.5000
   30.0000   19.0000
   35.0000    9.0000


ans =

    5.0000   49.1250  245.6250
    5.0000   47.1250  235.6250
    5.0000   44.2500  221.2500
    5.0000   40.0000  200.0000
    5.0000   34.0000  170.0000
    5.0000   24.7500  123.7500
    5.0000   14.0000   70.0000


Oil_Drained =

  1.2663e+003
```

Solution to Exercise 6.2.4 (p. 123)

```
clc
disp('A gas is expanded in an engine cylinder, following the law PV^1.3=c')
disp('The initial pressure is 2550 kPa and the final pressure is 210 kPa.')
disp('If the volume at the end of expansion is 0.75 m3,')
disp('Compute the work done by the gas.')
disp(' ')               % Display blank line
```

```
n=1.3;
P_i=2550;               % Initial pressure
P_f=210;                % Final pressure
V_f=.75;                % Final volume
V_i=(P_f*(V_f^n)/P_i)^(1/n); % Initial volume
c=P_f*V_f^n;
v=V_i:.001:V_f;         % Creating a row vector for volume, v
p=c./(v.^n);            % Computing pressure for volume
WorkDone=trapz(v,p)     % Integrating p*dv
```

The output is:

```
A gas is expanded in an engine cylinder, following the law PV^1.3=c
The initial pressure is 2550 kPa and the final pressure is 210 kPa.
If the volume at the end of expansion is 0.75 m3,
Compute the work done by the gas.


WorkDone =

  409.0666
```

Solution to Exercise 6.2.5 (p. 123)

```
    clc
disp('A force F acting on a body at a distance s from a fixed point is given by')
disp('F=3*s+(1/(s^2)) where s is the distance in meters')
disp('Compute the total work done in moving')
disp('From the position where s=1 to that where s=10.')
disp(' ')                     % Display blank line
s=1:.001:10;     % Creating a row vector for distance, s
F=3.*s+(1./(s.^2));    % Computing Force for s
WorkDone=trapz(s,F) % Integrating F*ds over 1 to 10 meters.
```

The output is:

```
    A force F acting on a body at a distance s from a fixed point is given by
F=3*s+(1/(s^2)) where s is the distance in meters
Compute the total work done in moving
From the position where s=1 to that where s=10.


WorkDone =

  149.4000
```

Solution to Exercise 6.2.6 (p. 123)

```
    clc                           % Clear screen
disp('This script computes the work done by')
disp('The gas in expanding from volume v1 to v2')
disp(' ')                     % Display blank line
```

```
a=input('Enter the constant a: ');
b=input('Enter the constant b: ');
p_i=input('Enter the initial pressure [kPa]: ');
v_i=input('Enter the initial volume [m3]: ');
v_f=input('Enter the final volume [m3]: ');
k=(p_i+(a/(v_i^2))*(v_i-b)); % Calculating constant k
v=v_i:.001:v_f;              % Creating a row vector for volume
p=(k./(v-b))-(a./(v.^2));    % Computing pressure for volume
WorkDone=trapz(v,p);         % Integrating p*dv
disp(' ')                    % Display blank line
str = ['The work done by the gas in expanding from ', num2str(v_i),...
 ' m3 to ' num2str(v_f), ' m3 is ', num2str(WorkDone), ' kW.'];
disp(str);
```

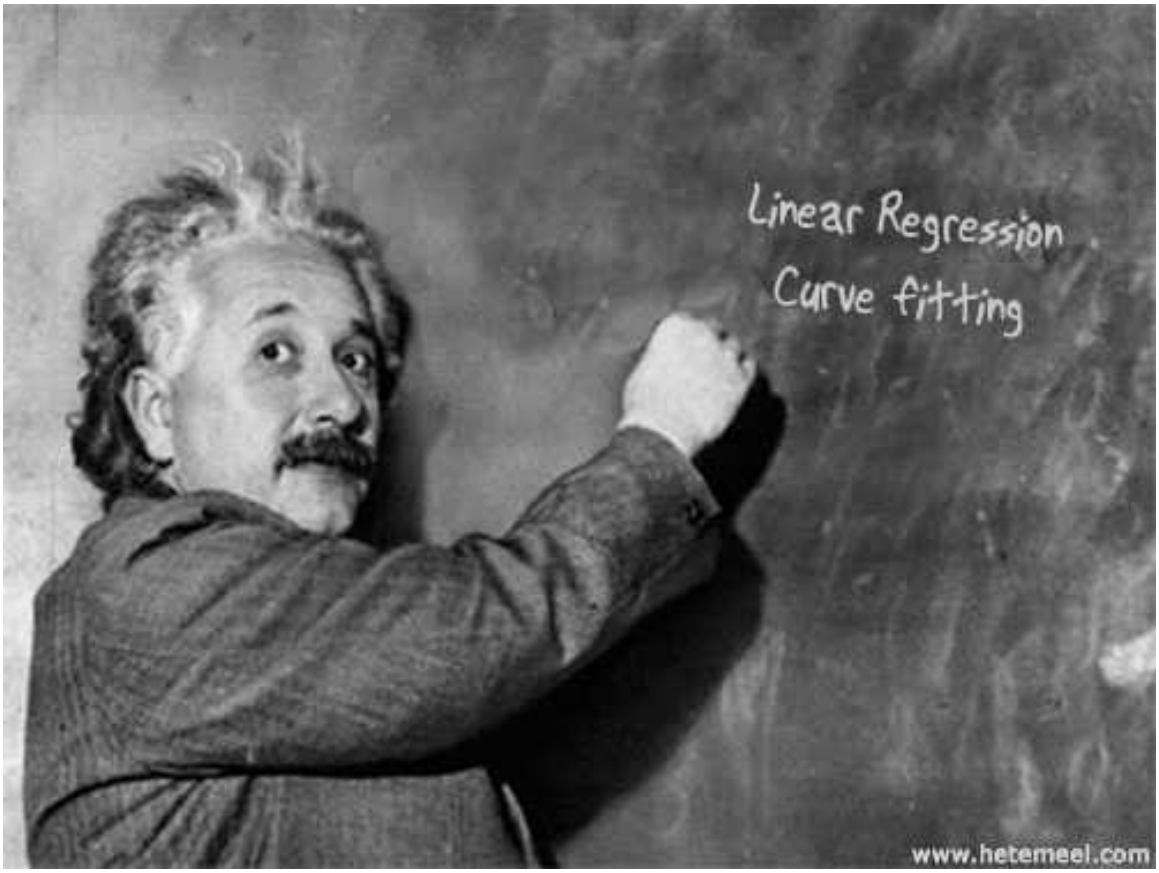
The output is:

```
    This script computes the work done by
The gas in expanding from volume v1 to v2

Enter the constant a: .01
Enter the constant b: .001
Enter the initial pressure [kPa]: 100
Enter the initial volume [m3]: 1
Enter the final volume [m3]: 2

The work done by the gas in expanding from 1 m3 to 2 m3 is 69.3667 kW.
```

Chapter 7

Regression Analysis

7.1 Regression Analysis[1]



7.1.1 What is Regression Analysis?

Suppose we calculate some variable of interest, y, as a function of some other variable x. We call y the dependent variable and x the independent variable. For example, consider the data set below, taken from a

[1] This content is available online at <http://cnx.org/content/m41448/1.4/>.

simple experiment involving a vehicle, its velocity versus time is tabulated. In this case, velocity is a function of time, thus velocity is the dependent variable and the time is the independent variable.

Time [s]	**Velocity [m/s]**
0	20
10	39
20	67
30	89
40	111
50	134
60	164
70	180
80	200

Table 7.1: Vehicle velocity versus time.

In its simplest form regression analysis involves fitting the best straight line relationship to explain how the variation in a dependent variable, y, depends on the variation in an independent variable, x. In our example above, once the relationship (in this case a linear relationship) has been estimated we can produce a linear equation in the following form:

$$y = mx + n \tag{7.1}$$

And once an analytic equation such as the one above has been determined, dependent variables at intermediate independent values can be computed.

7.1.2 Performing Linear Regression

Regression analysis with MATLAB is easy. The MATLAB Basic Fitting GUI allows us to interactively to do "curve fitting" which is a method to arrive at the best "straight line" fit for linear equations or the best curve fit for a polynomial up to the tenth degree. The procedure to perform a curve fitting with MATLAB is as follows:

1. Input the variables,
2. Plot the data,
3. Initialize the Basic Fitting GUI,
4. Select the desired regression analysis parameters.

Example 7.1

Using the data set above, determine the relationship between velocity and time.

First, let us input the variables (Workspace > New variable) as shown in the following figures.

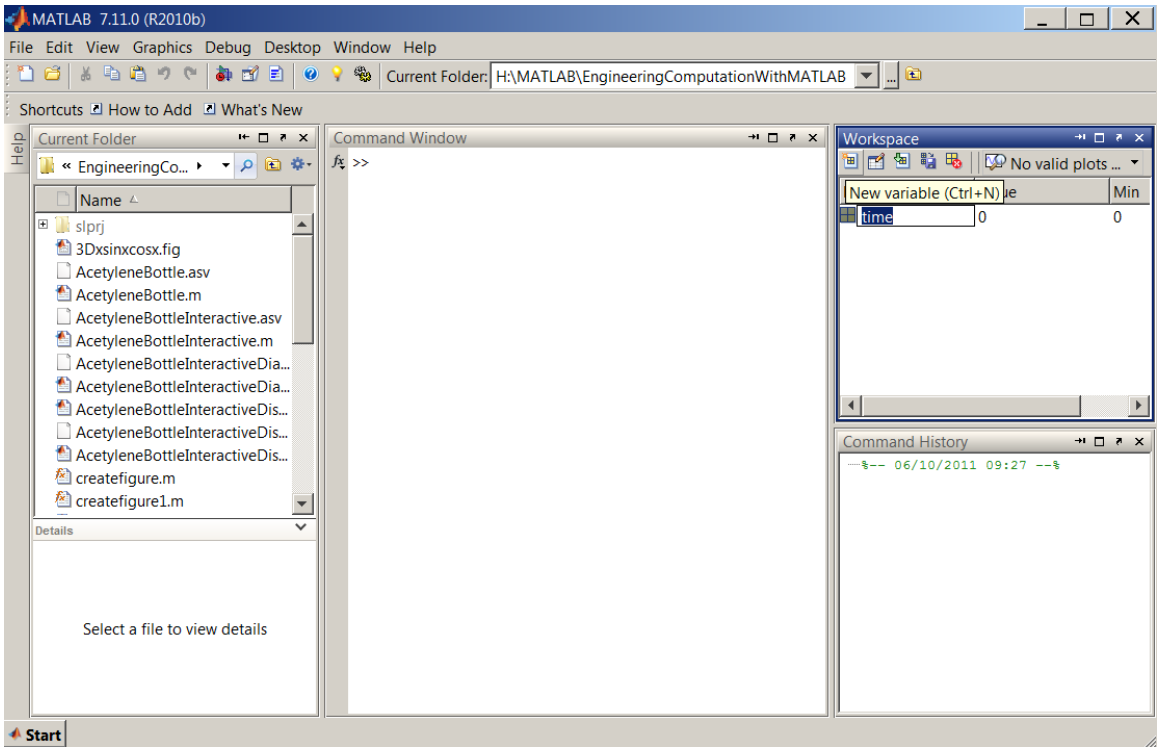


Figure 7.1: A new variable is created in the Workspace.

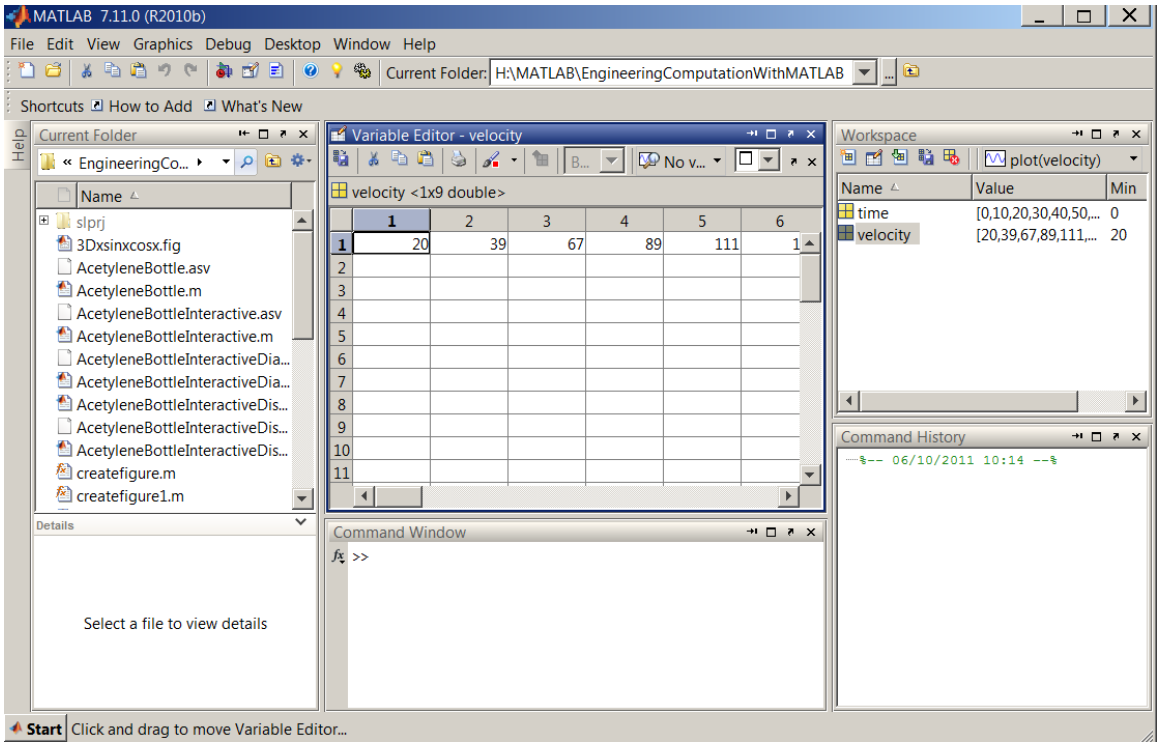


Figure 7.2: New variables are entered in the Variable Editor.

Second, we will plot the data by typing in `plot(time,velocity)` at the MATLAB prompt. The following plot is generated, select Tools > Basic Fitting:

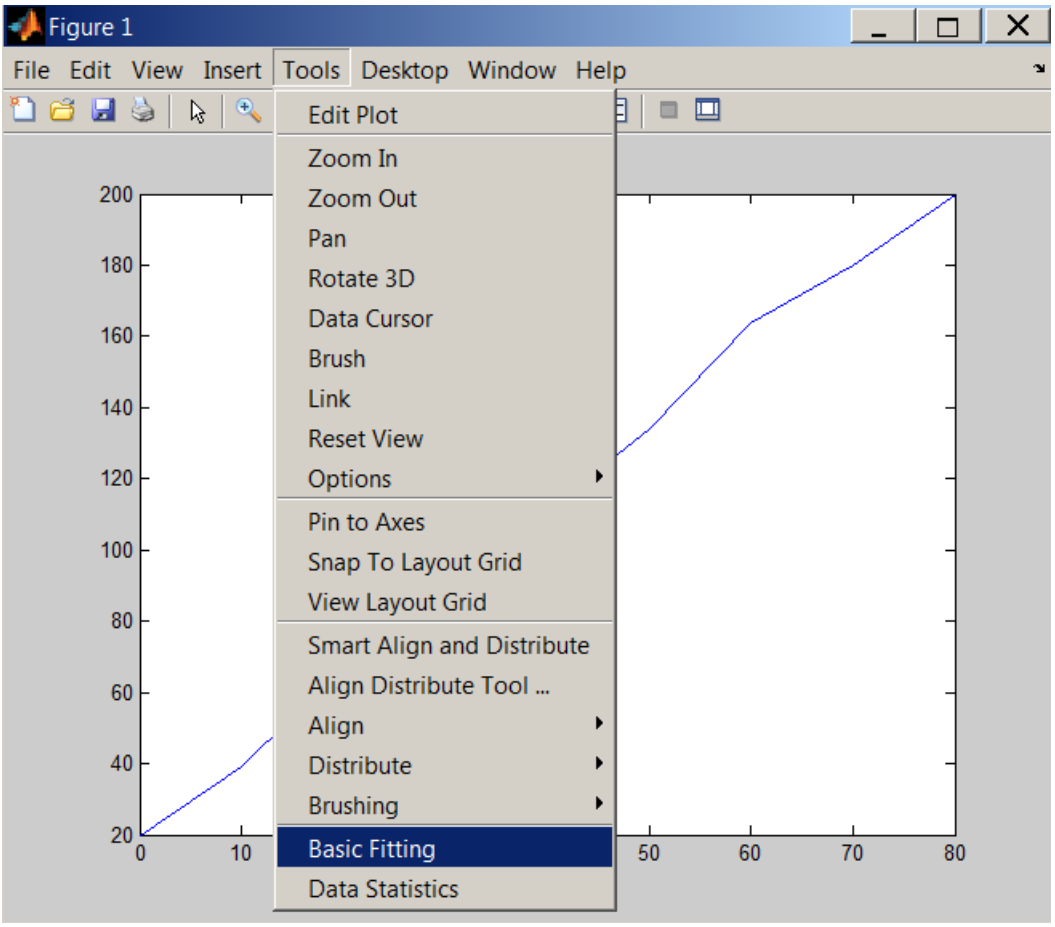


Figure 7.3: A plot is generated in Figure 1. The Basic Fitting tool can be initialized from Tools > Basic Fitting.

In the "Basic Fitting" window, select "linear" and "Show equations". The best fitting linear line along with the corresponding equation are displayed on the plot:

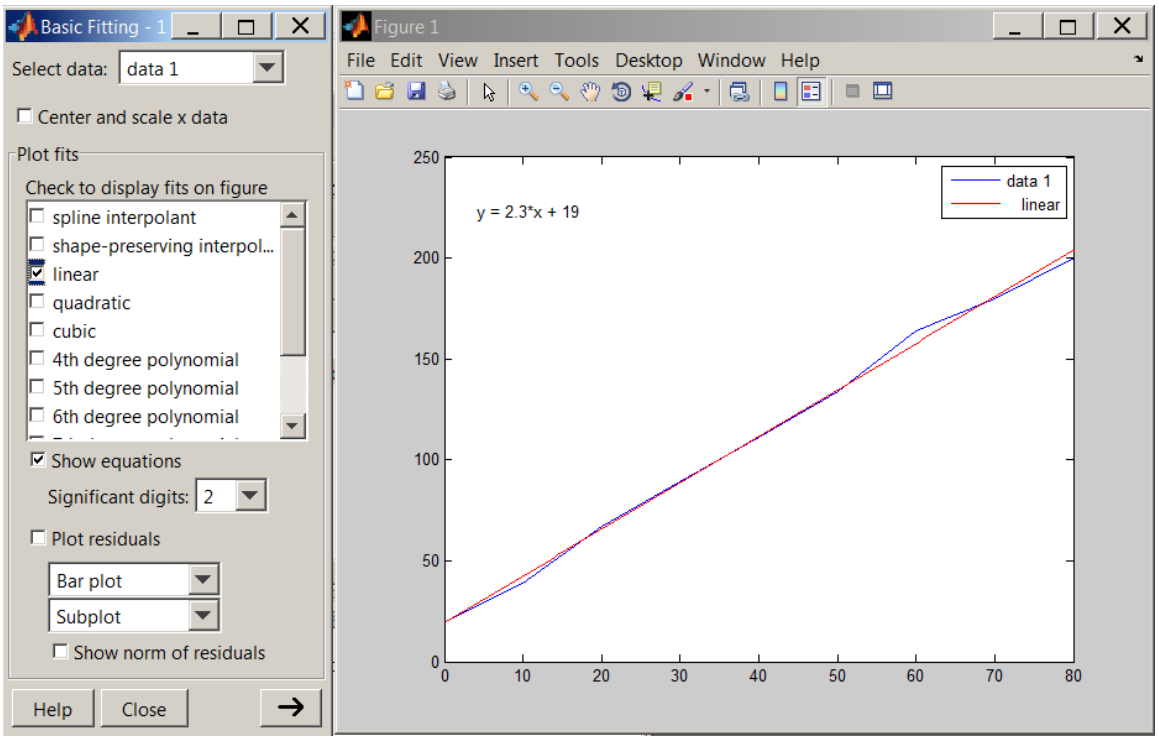


Figure 7.4: Basic Fitting window is used to select the desired regression analysis parameters.

Now let us do another curve fitting and obtain an equation for the function. Using that equation, we can evaluate the function at a desired value with `polyval`.

Example 7.2

The following is a collection of data for an iron-constantan thermocouple (data available for download[2]). [3]

[2] See the file at <http://cnx.org/content/m41448/latest/Chp7_Example2.zip>

[3] *Engineering Fundamentals and Problem Solving* by Arvid R. Eide, Roland Jenison, Larry L. Northup, Steven K. Mikelson , McGraw-Hill Higher Education. ©2007 p.114

Temperature [C]	Voltage [mV]
50	2.6
100	6.7
150	8.8
200	11.2
300	17.0
400	22.5
500	26
600	32.5
700	37.7
800	41
900	48
1000	55.2

Table 7.2: Temperature [C] vs Voltage [mV]

a. Plot a graph with Temperature as the independent variable.
b. Determine the equation of the relationship using the Basic Fitting tools.
c. Estimate the Voltage that corresponds to a Temperature of 650 C and 1150 C.

We will input the variables first:

```
Temp=[50;100;150;200;300;400;500;600;700;800;900;1000]

Voltage=[2.6;6.7;8.8;11.2;17;22.5;26;32.5;37.7;41;48;55.2]
```

To plot the graph, type in:

```
plot(Temp,Voltage)
```

We can now use the Plot Tools and Basic Fitting settings and determine the equation:

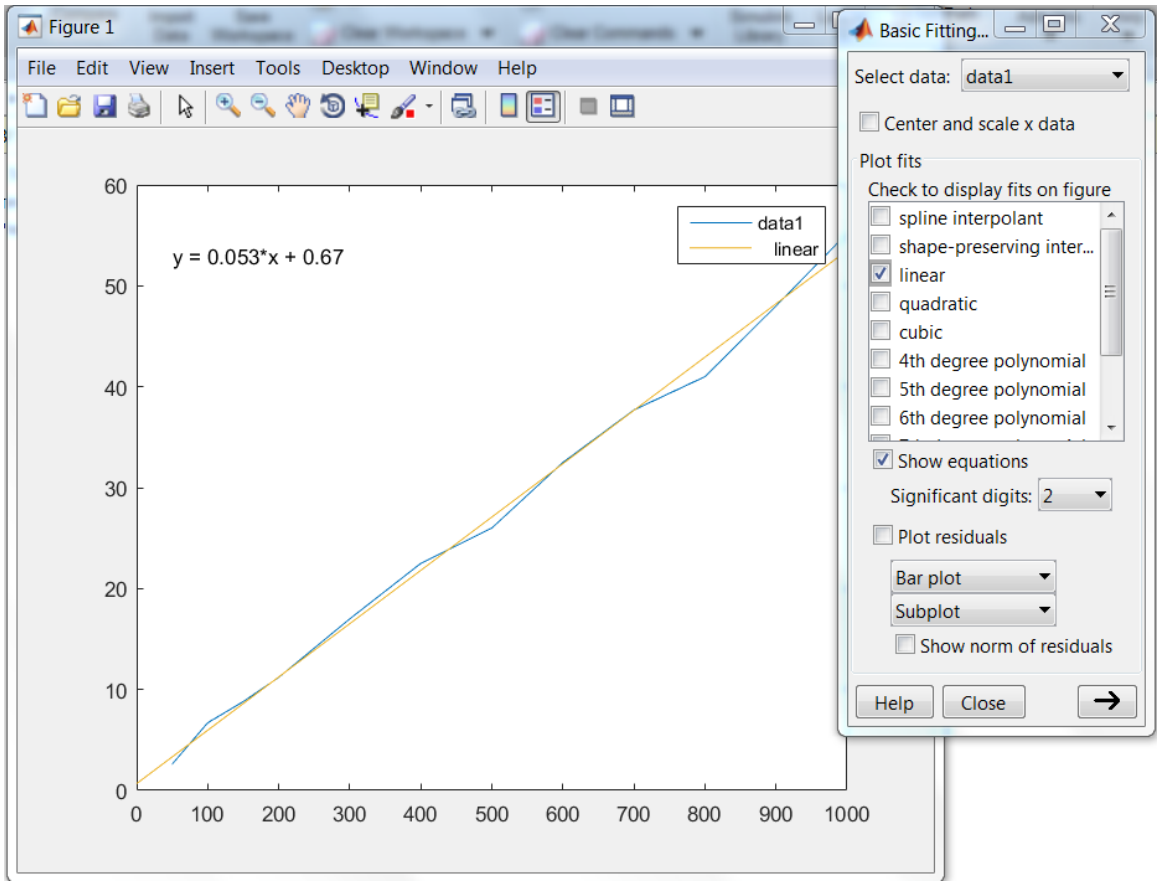


Figure 7.5: Basic Fitting window is used to select the desired regression analysis parameters.

By clicking the right arrow twice at the bottom right corner on the Basic Fitting window, we can evaluate the function at a desired value. See the figure below which illustrates this process for the temperature value 1150 C.

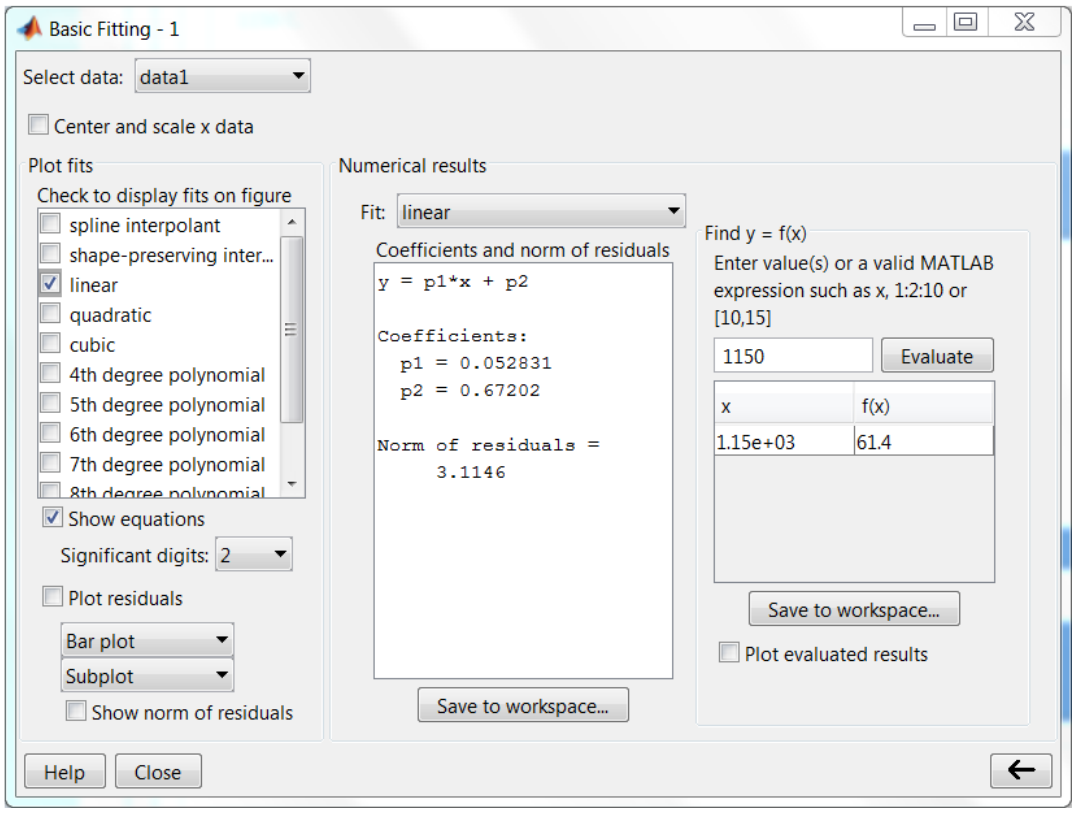


Figure 7.6: Estimating the Voltage that corresponds to a Temperature of 1150 C.

Now let us check our answer with a technique we learned earlier. As displayed on the plot, we have obtained the following equation: $y = 0.052831x + 0.67202$ This equation can be entered as polynomial and evaluated at 650 and 1150 as follows:

```
≫ p=[0.052831,0.67202]

p =

    0.0528    0.6720

≫ polyval(p,1150)

ans =

   61.4277
```

7.1.3 Summary of Key Points

1. Linear regression involves fitting the best straight line relationship to explain how the variation in a dependent variable, y, depends on the variation in an independent variable, x,
2. Basic Fitting GUI allows us to interactively perform curve fitting,
3. Some of the plot fits available are linear, quadratic and cubic functions,
4. Basic Fitting GUI can evaluate functions at given points.

7.2 Problem Set[4]

Exercise 7.2.1 *(Solution on p. 140.)*

Using the following experimental values [5], plot a distance-time graph and determine the equation, relating the distance and time for a moving object.

Distance [m]	Time [s]
0	0
24	5
48	10
72	15
96	20

Table 7.3: Experimental data.

Exercise 7.2.2 *(Solution on p. 140.)*

Using the data set below, determine the relationship between temperature and internal energy.

Temperature [C]	Internal Energy [kJ/kg]
100	2506.7
150	2582.8
200	2658.1
250	2733.7
300	2810.4
400	2967.9
500	3131.6

Table 7.4: An extract from Steam Tables

Exercise 7.2.3 *(Solution on p. 141.)*

Using the following experimental values [6], plot a velocity-time graph and determine the equation, relating the velocity and time for a moving object.

[4] This content is available online at <http://cnx.org/content/m48021/1.1/>.
[5] *Engineering Science* by E. Hughes and C. Hughes, Longman ©1994, (p. 375)
[6] *Engineering Science* by E. Hughes and C. Hughes, Longman ©1994, (p. 375)

Velocity [m/s]	**Time [s]**
12	0
142	5
512	10
1122	15
1972	20

Table 7.5: Experimental data.

Solutions to Exercises in Chapter 7

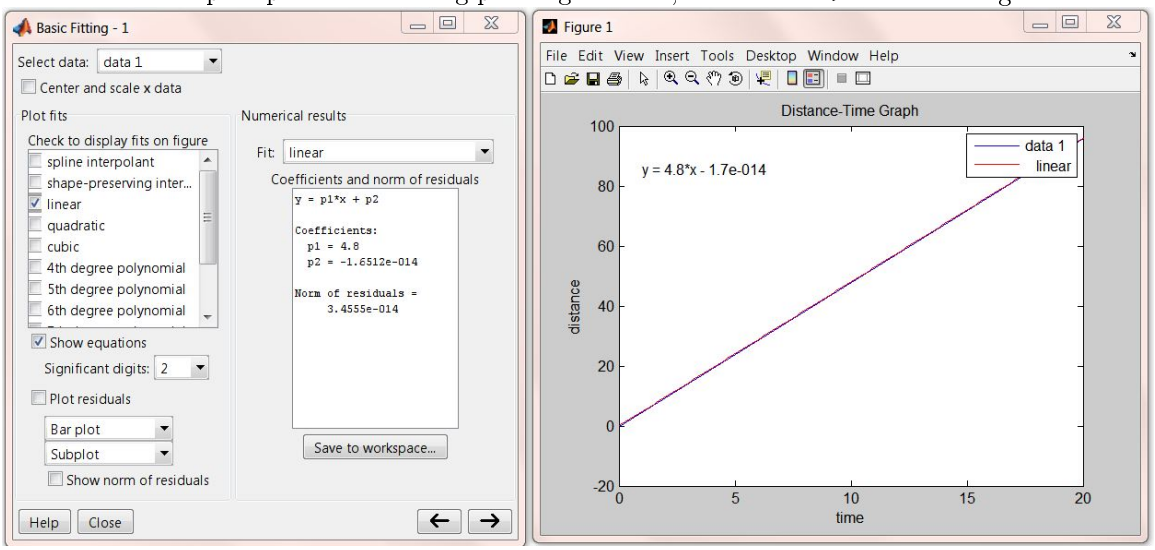
Solution to Exercise 7.2.1 (p. 138)
Data can be entered as follows:

```
    distance=[0 24 48 72 96];
time=[0 5 10 15 20];
```

we can now plot the data by typing in

```
plot(time,distance);title('Distance-Time Graph');xlabel('time');ylabel('distance');
```

at the MATLAB prompt. The following plot is generated, select Tools > Basic Fitting:



As shown above, the relationship between distance and time is:

$y = 4.8x - 1.7 \times 10^{-14}$
or
Distance $= 4.8\text{Time} - 1.7 \times 10^{-14}$

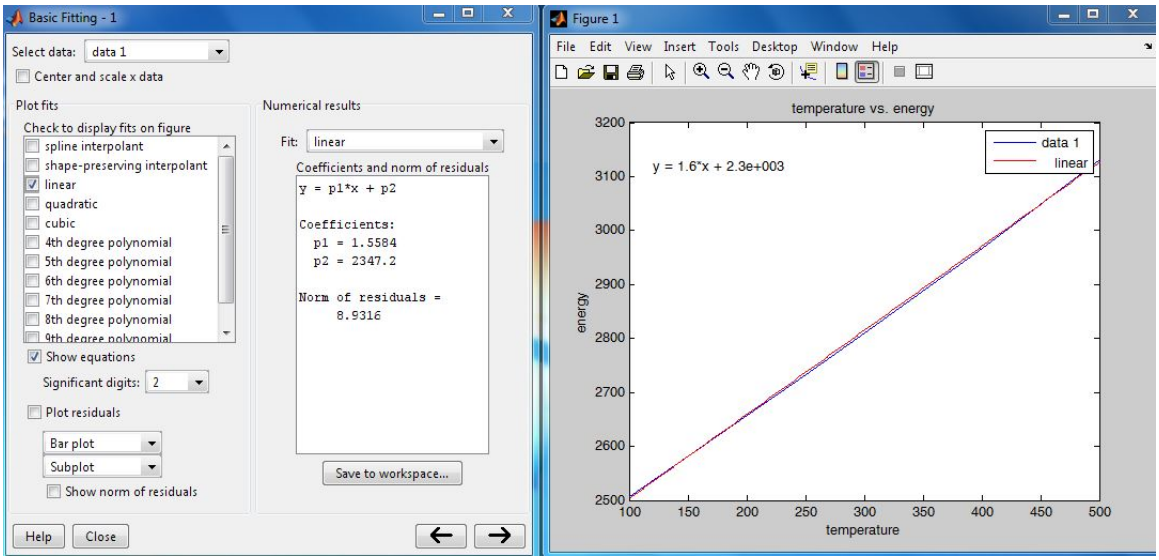
Solution to Exercise 7.2.2 (p. 138)
Data can be entered as follows:

```
    temperature = [100, 150, 200, 250, 300, 400, 500];
energy = [2506.7, 2582.8, 2658.1, 2733.7, 2810.4, 2967.9, 3131.6];
```

we can now plot the data by typing in

```
plot(temperature,energy);title('temperature vs. energy');xlabel('temperature');ylabel('energy');
```

at the MATLAB prompt. The following plot is generated, select Tools > Basic Fitting:



As shown above, the relationship between temperature and internal energy is:

$y = 1.6x + 2347.2$
or
internal energy $= 1.6$temperature $+ 2347.2$

Solution to Exercise 7.2.3 (p. 138)

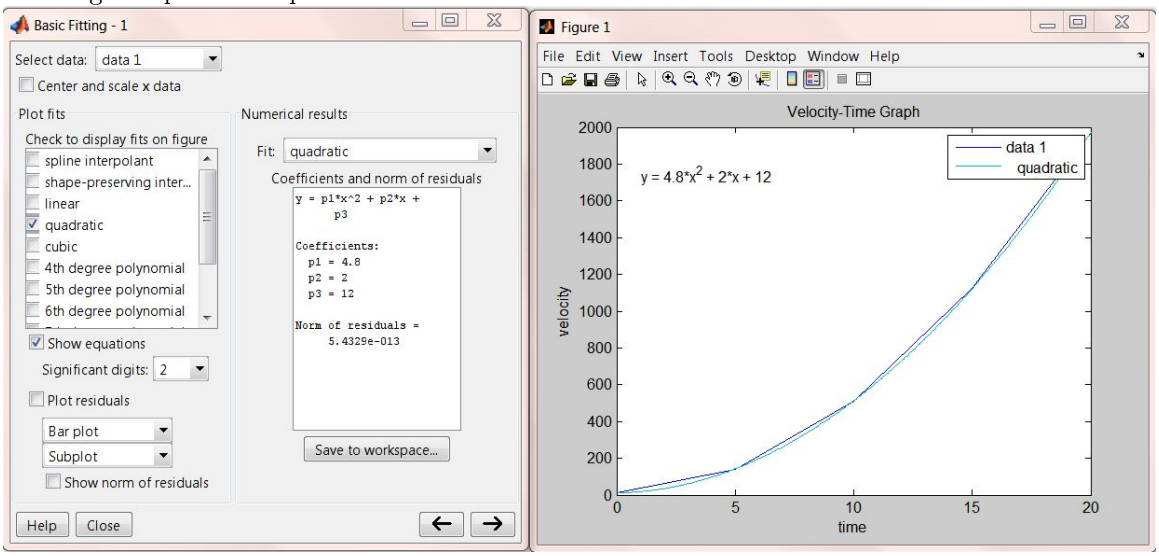
Data can be entered as follows:

```
    velocity=[12 142 512 1122 1972];
time=[0 5 10 15 20];
```

we can now plot the data by typing in

```
    plot(time,velocity);title('Velocity-Time Graph');xlabel('time');ylabel('velocity');
```

at the MATLAB prompt. The following plot is generated, select Tools > Basic Fitting, notice that we are choosing the quadratic option this time:



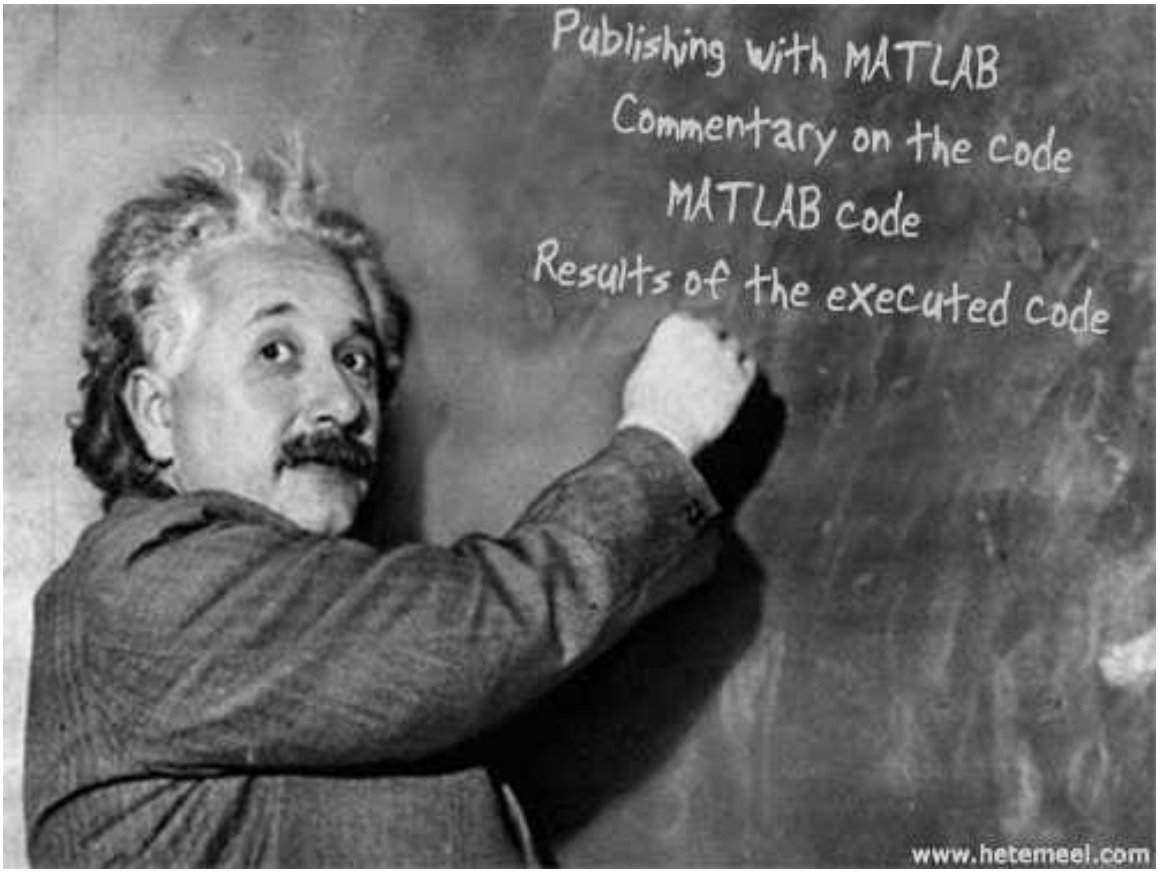
As shown above, the relationship between velocity and time is:

$y = 4.8x^2 + 2x + 12$

Chapter 8

Publishing with MATLAB

8.1 Generating Reports with MATLAB[1]



MATLAB includes an automatic report generator called publisher. The publisher publishes a script in several formats, including HTML, XML, MS Word and PowerPoint. The published file can contain the following:

- Commentary on the code,
- MATLAB code,

[1] This content is available online at <http://cnx.org/content/m41457/1.2/>.

- Results of the executed code, including the Command Window output and figures created by the code.

8.1.1 The `publish` Function

The most basic syntax is `publish('file','format')` where the m-file is called and executed line by line then saved to a file in specified format. All published files are placed in the html directory although the published output might be a doc file.

8.1.2 Publishing with Editor

The publisher is easily accessible from the Tool Strip:

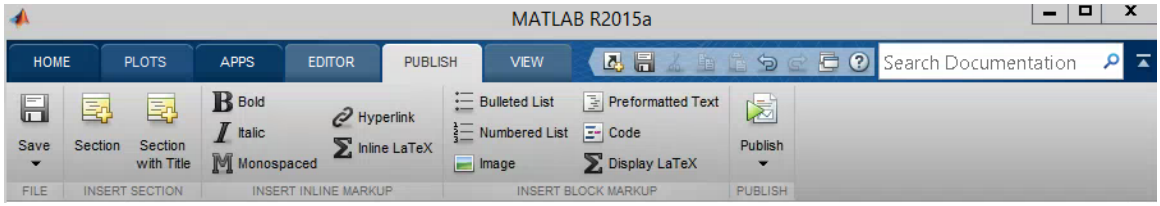


Figure 8.1: Publish tab on the Tool Strip.

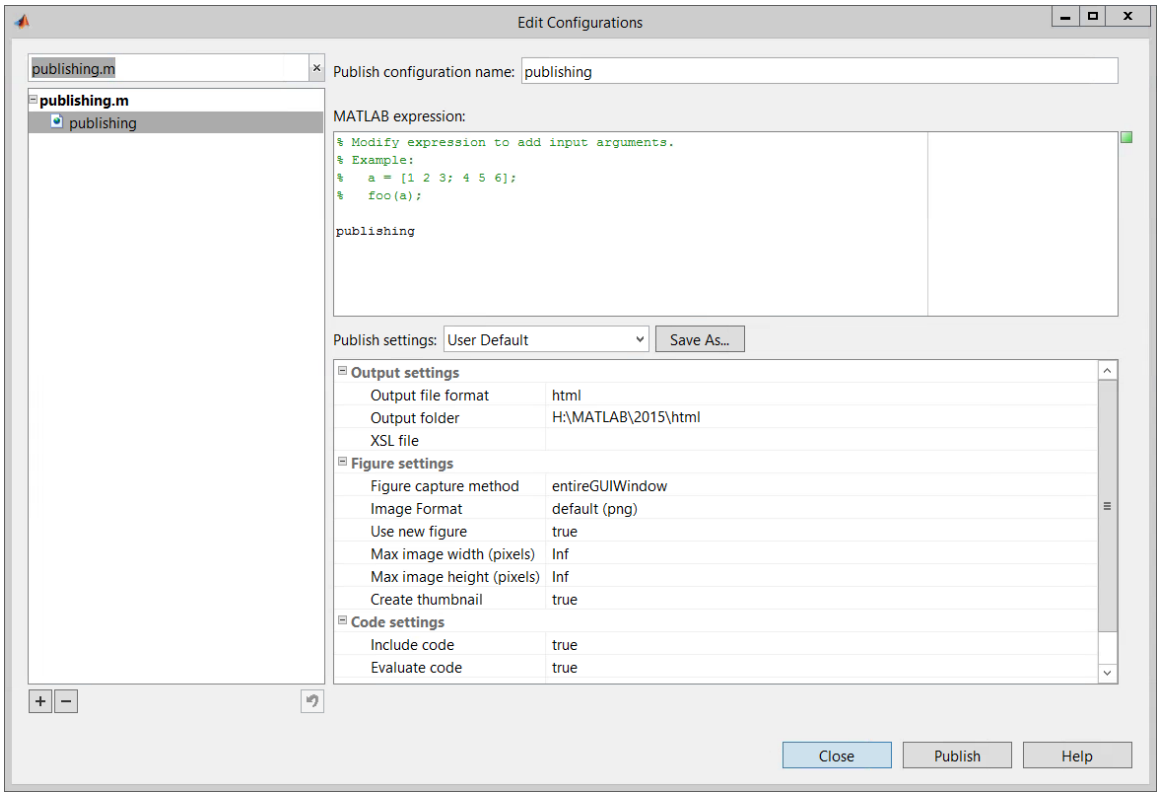


Figure 8.2: Editing publishing options, currently output format is html.

Example 8.1

Write a simple script and publish it in an html file.

Select File > New > Script to create an m-file. Once the editor is opened, type in the following code:

```
    x = [0:6];   % Create a row vector
y = 1.6*x;   % Compute y as a function of x
[x',y']      % Transpose vectors x and y
plot(x,y),title('Graph of y=f(x)'),xlabel('x'),ylabel('f(x)'),grid % Plot a graph
```

Save the script as publishing.m and select Publish > Publish.

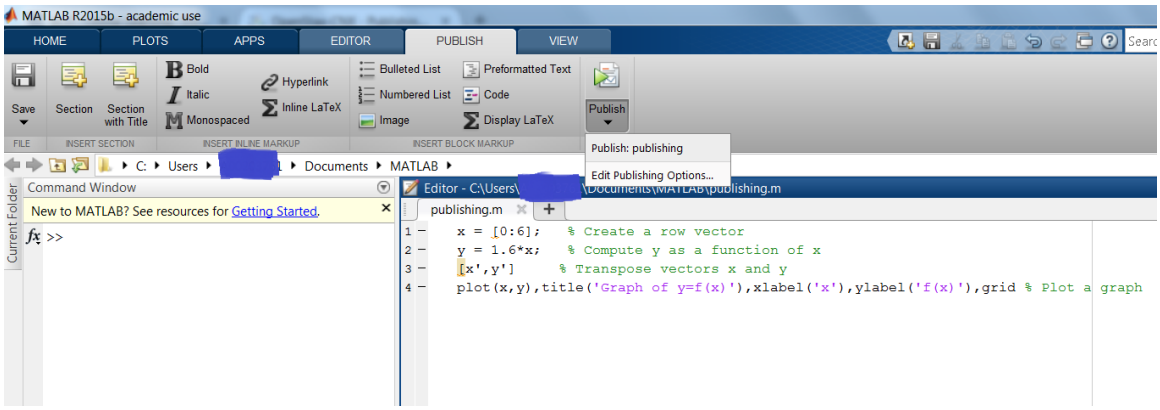


Figure 8.3: Publishing a script.

An HTML file is generated as shown in the figure below:

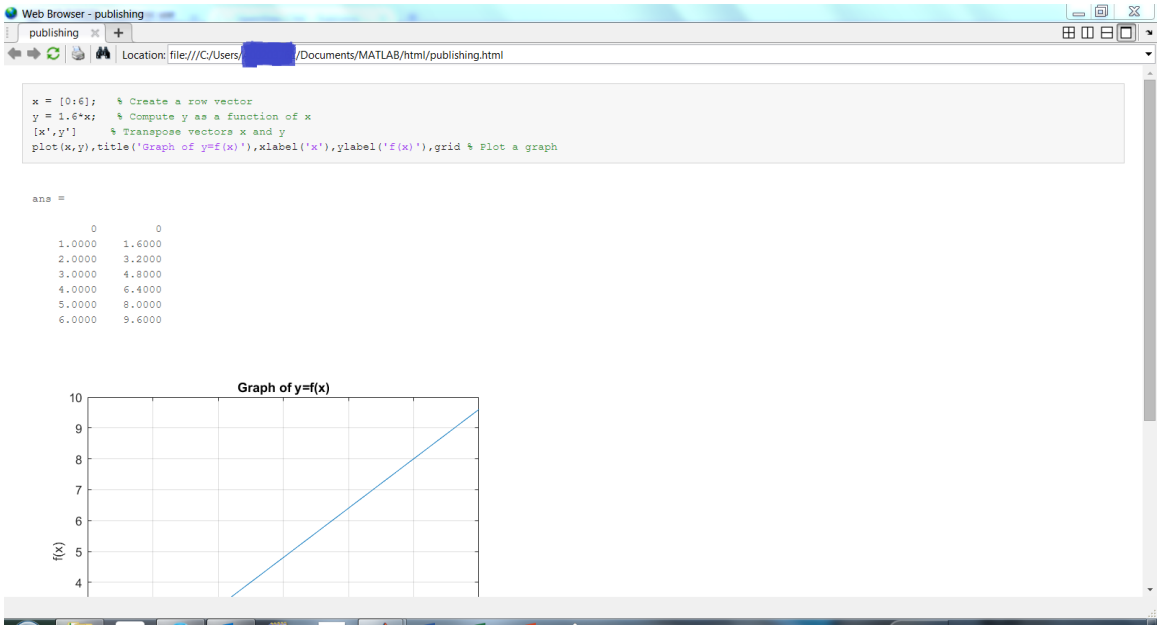


Figure 8.4: A script published in html

8.1.3 The Double Percentage %% Sign

The scripts sometimes can be very long and their readability might be reduced. To improve the publishing result, sections are introduced by adding descriptive lines to the script preceded by %%. Consider the following

example.

Example 8.2

Edit the script created in the example above to look like the code below:

```
    %% This file creates vectors, displays results and plots an x-y graph
x = [0:6];   % Create a row vector
y = 1.6*x;   % Compute y as a function of x
%% Tabulated data
[x',y']     % Transpose vectors x and y
%% Graph of y=f(x)
plot(x,y),title('Graph of y=f(x)'),xlabel('x'),ylabel('f(x)'),grid % Plot a graph
```

Save the script, a new HTML file is generated as shown in the figure below:

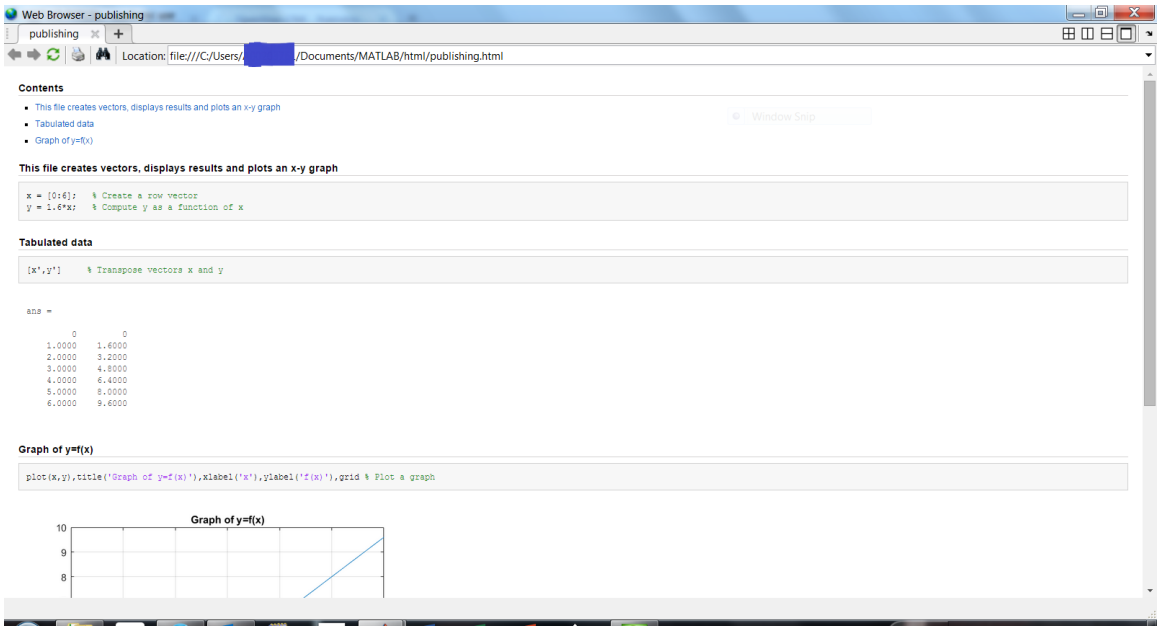


Figure 8.5: An html file with sections

8.1.4 Summary of Key Points

1. MATLAB can generate reports containing commentary on the code, MATLAB code and the results of the executed code,
2. The publisher generates a script in several formats, including HTML, XML, MS Word and PowerPoint.
3. The Double Percentage %% can be used to creates hyper-linked sections.

8.2 Problem Set[2]

Exercise 8.2.1 *(Solution on p. 149.)*

Write a script to plot function $y = \frac{\sin(x)}{x}$ for $\frac{\pi}{100} \leq x \leq 10\pi$ using increments of $\frac{\pi}{100}$. Publish your m-file to html.

Exercise 8.2.2 *(Solution on p. 150.)*

A gas is expanded in an engine cylinder, following the law PV1.3=c. The initial pressure is 2550 kPa and the final pressure is 210 kPa. If the volume at the end of expansion is 0.75 m3, write a script to compute the work done by the gas and publish your solution to an html file. This is the same problem as this Problem you have solved before. (Exercise 6.2.4)

Exercise 8.2.3 *(Solution on p. 151.)*

A force F acting on a body at a distance s from a fixed point is given by $F = 3s + \frac{1}{s^2}$. Write a script to compute the work done when the body moves from the position where s=1 to that where s=10 and and publish your solution to an html file. Include a table of contents in the output file. This is the same problem as this Problem you have solved before. (Problem 6.2.5)

[2] This content is available online at <http://cnx.org/content/m48023/1.1/>.

Solutions to Exercises in Chapter 8

Solution to Exercise 8.2.1 (p. 148)
The m-file content:

```
% This script plots a graph of Graph of y=sin(x)/x
clc                      % Clear screen
x = pi/100:pi/100:10*pi; % Create a row vector
y = sin(x)./x;           % Calculate y as function of x
plot(x,y),title('Graph of y=sin(x)/x'),xlabel('x'),ylabel('y'),grid
```

The html output:

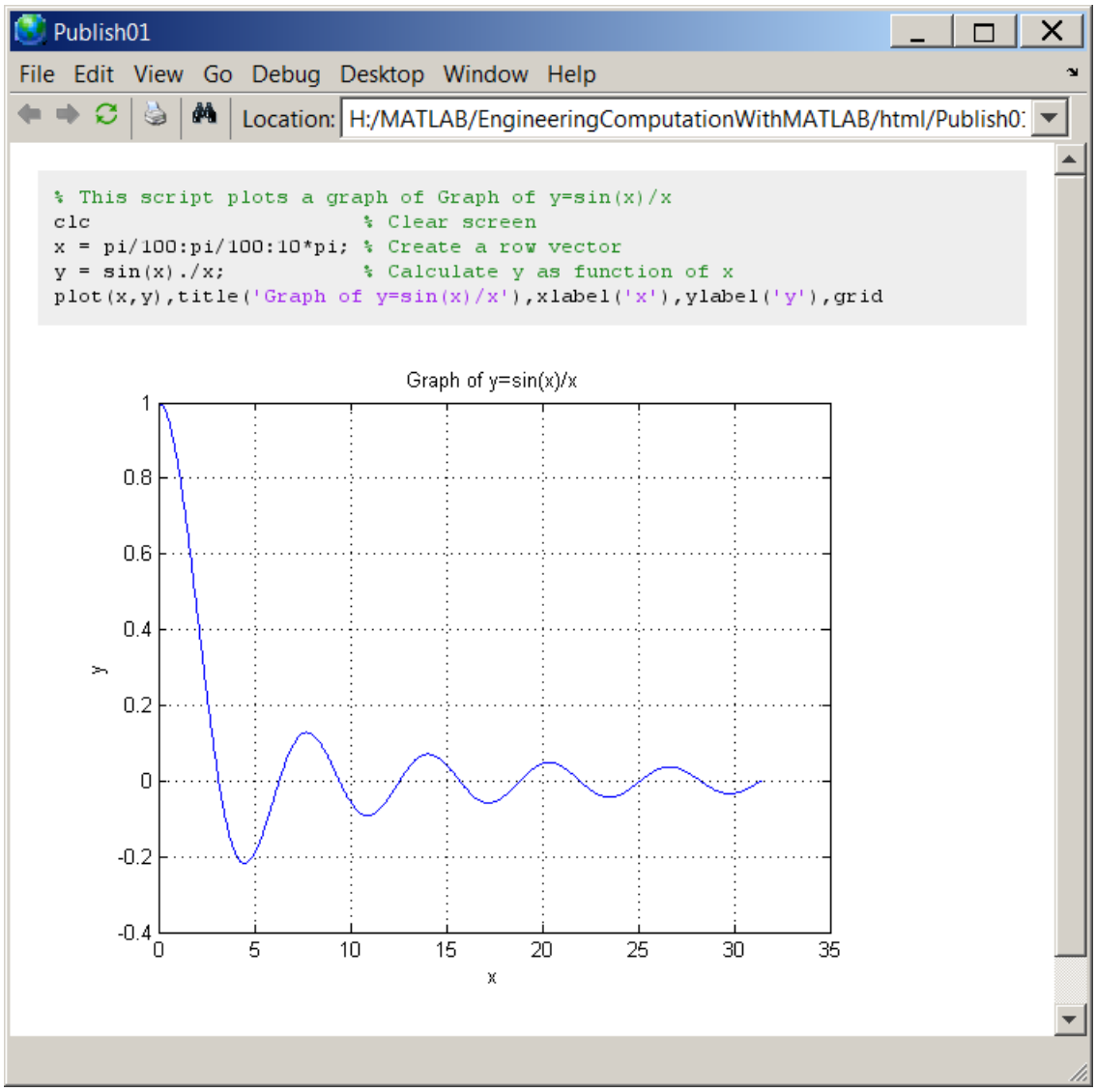


Figure 8.6: The published html file.

Solution to Exercise 8.2.2 (p. 148)

The m-file content:

```
clc
disp('A gas is expanded in an engine cylinder, following the law PV^1.3=c')
disp('The initial pressure is 2550 kPa and the final pressure is 210 kPa.')
disp('If the volume at the end of expansion is 0.75 m3,')
disp('Compute the work done by the gas.')
disp(' ')               % Display blank line
n=1.3;
```

```
P_i=2550;               % Initial pressure
P_f=210;                % Final pressure
V_f=.75;                % Final volume
V_i=(P_f*(V_f^n)/P_i)^(1/n); % Initial volume
c=P_f*V_f^n;
v=V_i:.001:V_f;         % Creating a row vector for volume, v
p=c./(v.^n);            % Computing pressure for volume
WorkDone=trapz(v,p)     % Integrating p*dv
```

The html output:

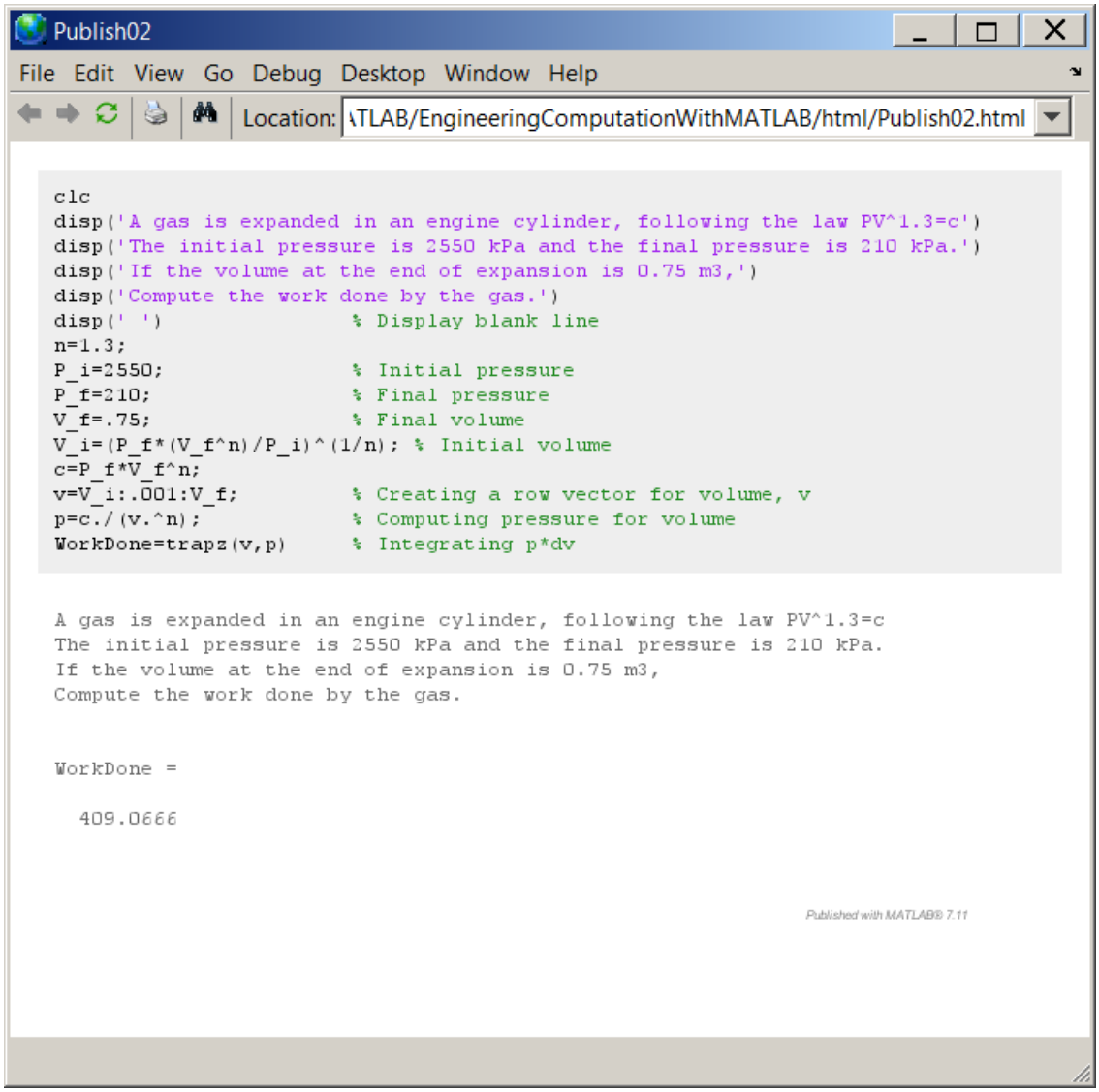


Figure 8.7: The published html file.

Solution to Exercise 8.2.3 (p. 148)
The m-file content:

```
    %% Work done
% This script computes the work done on an object
clc
disp('A force F acting on a body at a distance s from a fixed point is given by')
disp('F=3*s+(1/(s^2)) where s is the distance in meters')
disp('Compute the total work done in moving')
disp('From the position where s=1 to that where s=10.')
disp(' ')                      % Display blank line
%% Create a row vector for distance, s
s=1:.001:10;
%% Compute Force for s
F=3.*s+(1./(s.^2));    % Computing Force for s
%% Integrating F*ds over 1 to 10 meters.
WorkDone=trapz(s,F)
```

The html output:

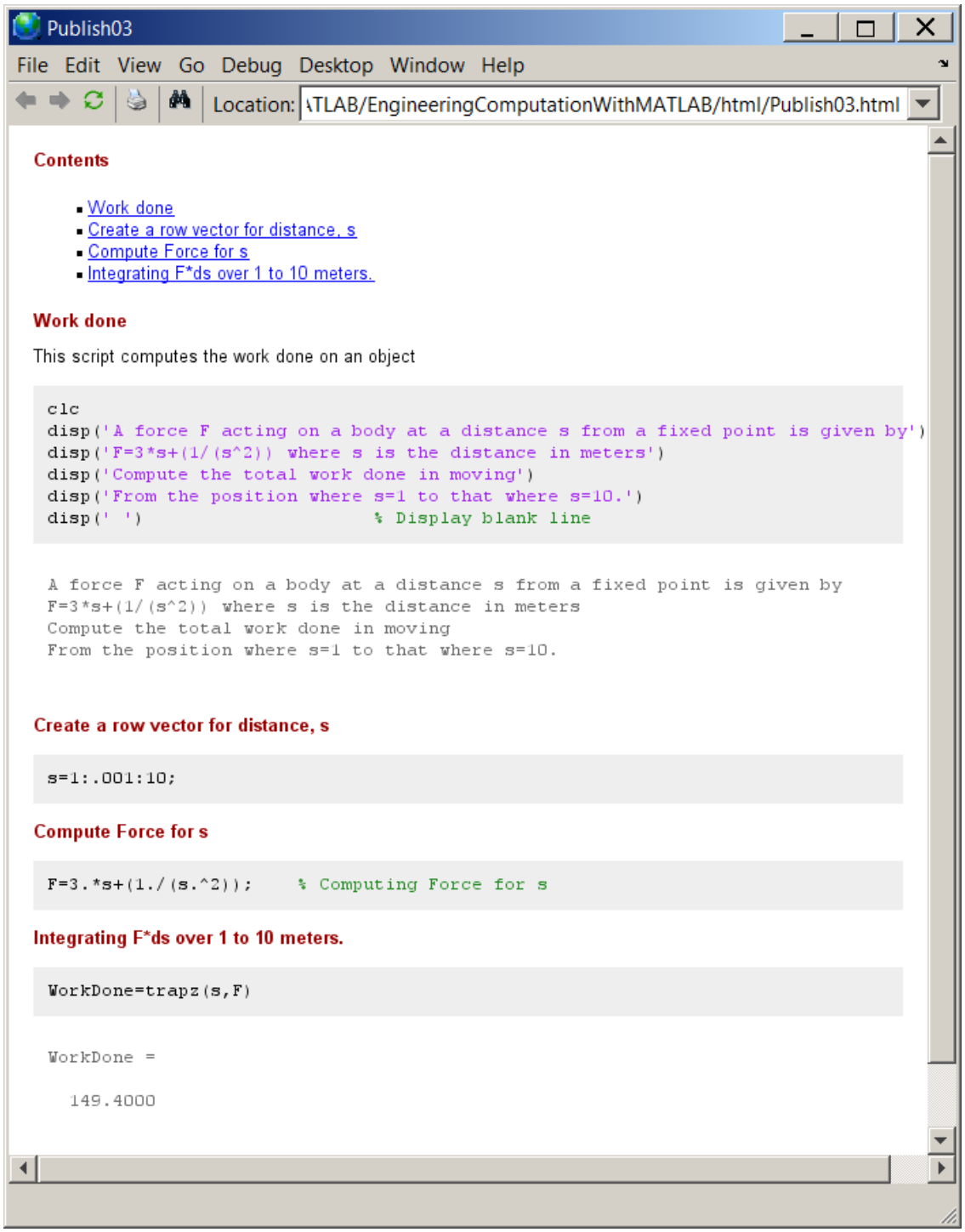


Figure 8.8: The published html file.

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Collection: *A Brief Introduction to Engineering Computation with MATLAB*
Edited by: Serhat Beyenir
URL: http://cnx.org/content/col11371/1.11/
License: http://creativecommons.org/licenses/by/4.0/

Module: "Acknowledgements"
By: Serhat Beyenir
URL: http://cnx.org/content/m50977/1.3/
Page: 1
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Preface"
By: Serhat Beyenir
URL: http://cnx.org/content/m41458/1.6/
Page: 3
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/3.0/

Module: "Study Guide"
By: Serhat Beyenir
URL: http://cnx.org/content/m41459/1.2/
Page: 5
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/3.0/

Module: "What is MATLAB?"
By: Serhat Beyenir
URL: http://cnx.org/content/m41403/1.5/
Pages: 7-22
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "What is MATLAB? | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m41463/1.3/
Page: 22
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "MATLAB Essentials"
Used here as: "Essentials"
By: Serhat Beyenir
URL: http://cnx.org/content/m41409/1.3/
Pages: 25-42
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "MATLAB Essentials | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m41464/1.7/
Pages: 42-43
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Graphing with MATLAB"
Used here as: "Plotting in MATLAB"
By: Serhat Beyenir
URL: http://cnx.org/content/m41442/1.3/
Pages: 49-67
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Graphing with MATLAB | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m41466/1.7/
Pages: 67-70
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/3.0/

Module: "Introductory Programming with MATLAB"
Used here as: "Writing Scripts to Solve Problems"
By: Serhat Beyenir
URL: http://cnx.org/content/m41440/1.6/
Pages: 81-94
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Introductory Programming with MATLAB | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m41536/1.3/
Pages: 94-96
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Interpolation with MATLAB"
Used here as: "Interpolation"
By: Serhat Beyenir
URL: http://cnx.org/content/m41455/1.3/
Pages: 105-108
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/3.0/

Module: "Interpolation with MATLAB | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m41624/1.2/
Pages: 108-110
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/3.0/

Module: "Numerical Integration with MATLAB"
Used here as: "Computing the Area Under a Curve"
By: Serhat Beyenir
URL: http://cnx.org/content/m41454/1.4/
Pages: 115-122
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/3.0/

Module: "Numerical Integration with MATLAB | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m41541/1.9/
Pages: 122-123
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Regression Analysis with MATLAB"
Used here as: "Regression Analysis"
By: Serhat Beyenir
URL: http://cnx.org/content/m41448/1.4/
Pages: 129-138
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Regression Analysis with MATLAB | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m48021/1.1/
Pages: 138-139
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/3.0/

Module: "Publishing with MATLAB"
Used here as: "Generating Reports with MATLAB"
By: Serhat Beyenir
URL: http://cnx.org/content/m41457/1.2/
Pages: 143-147
Copyright: Serhat Beyenir
License: http://creativecommons.org/licenses/by/4.0/

Module: "Publishing with MATLAB | Problem Set"
Used here as: "Problem Set"
By: Serhat Beyenir
URL: http://cnx.org/content/m48023/1.1/
Page: 148

A Brief Introduction to Engineering Computation with MATLAB
An Engineering Computation Primer.

About OpenStax-CNX
Rhaptos is a web-based collaborative publishing system for educational material.