

PowerPoint to accompany

**Introduction to MATLAB
for Engineers, Third Edition**

William J. Palm III

Chapter 1

An Overview of MATLAB®



Copyright © 2010. The McGraw-Hill Companies, Inc.

The Default MATLAB Desktop, Fig. 1.1-1 page 5

The screenshot displays the MATLAB Desktop environment. The top menu bar includes File, Edit, Debug, Desktop, Window, and Help. The current directory is set to c:\MyMATLABFiles. The interface is divided into several panes:

- Current Directory:** Shows a list of files and folders in the current directory, including actuator.m, cost.m, CubicEval.m, cubicsprin..., dcmotor.m, ethanol.m, Falling_Sp..., friction1.mdl, half-sine.m, height.m, perfspecs.m, plantcost.m, speed.m, TorqueDat..., trapezoid.m, and trebuchet... The selected file is trapezoid.m (M-File), which is a Trapezoidal voltage profile.
- Command Window:** Contains the following MATLAB commands and their outputs:

```
>> clear
>> A=2*5^3

A =

    250

>> B=exp(0.005*A)

B =

    3.4903

>> C=4*sqrt(A+B^3)

C =

    68.4130

>> D=5*log10(1000)

D =

    15

>> x = 0:0.02:9;
>> y = A*sin(3*x);
>> plot(x,y)
>> |
```
- Workspace:** Displays the current workspace variables:

Name	Value	Min	Max
A	250	250	250
B	3.4903	3.4903	3.4903
C	68.4130	68.4130	68.4130
D	15	15	15
x	<1x451 double>	0	9
y	<1x451 double>	-249....	249.9...
- Command History:** Shows a list of commands entered in the Command Window, including clear, A=2*5^3, B=exp(0.05*A), C=4*sqrt(A^3+B^3), D=5*log10(1000), x = 0:0.03:9, clc, and plot(x,y).

Entering Commands and Expressions

- MATLAB retains your previous keystrokes.
- Use the up-arrow key to scroll back back through the commands.
- Press the key once to see the previous entry, and so on.
- Use the down-arrow key to scroll forward. Edit a line using the left- and right-arrow keys the **Backspace** key, and the **Delete** key.
- Press the **Enter** key to execute the command.

Scalar arithmetic operations

Table 1.1–1, page 8

Symbol	Operation	MATLAB form
\wedge	exponentiation: a^b	a^b
$*$	multiplication: ab	$a*b$
$/$	right division: $a/b = \frac{a}{b}$	a/b
\backslash	left division: $a \backslash b = \frac{b}{a}$	$a \backslash b$
$+$	addition: $a + b$	$a+b$
$-$	subtraction: $a - b$	$a-b$

An Example Session, Pages 7-8

```
>> 8/10
ans =
    0.8000
>> 5*ans
ans =
     4
>> r=8/10
r =
    0.8000
>> r
r =
    0.8000
>> s=20*r
s =
    16
```

Order of precedence

Table 1.1–2, page 9

Precedence	Operation
First	Parentheses, evaluated starting with the innermost pair.
Second	Exponentiation, evaluated from left to right.
Third	Multiplication and division with equal precedence, evaluated from left to right.
Fourth	Addition and subtraction with equal precedence, evaluated from left to right.

Examples of Precedence, Page 9

```
>> 8 + 3*5
```

```
ans =
```

```
23
```

```
>> 8 + (3*5)
```

```
ans =
```

```
23
```

```
>> (8 + 3)*5
```

```
ans =
```

```
55
```

```
>> 4^2-12-8/4*2
```

```
ans =
```

```
0
```

```
>> 4^2-12- 8/(4*2)
```

```
ans =
```

```
3
```

Examples of Precedence, Page 9 Continued

```
>> 3*4^2 + 5
```

```
ans =
```

```
53
```

```
>>(3*4)^2 + 5
```

```
ans =
```

```
149
```

```
>>27^(1/3) + 32^(0.2)
```

```
ans =
```

```
5
```

```
>>27^(1/3) + 32^0.2
```

```
ans =
```

```
5
```

```
>>27^1/3 + 32^0.2
```

```
ans =
```

```
11
```


Commands for managing the work session

Table 1.1–3, Page 12

Command	Description
<code>clc</code>	Clears the Command window.
<code>clear</code>	Removes all variables from memory.
<code>clear var1 var2</code>	Removes the variables <code>var1</code> and <code>var2</code> from memory.
<code>exist('name')</code>	Determines if a file or variable exists having the name 'name'.
<code>quit</code>	Stops MATLAB.
<code>who</code>	Lists the variables currently in memory.
<code>whos</code>	Lists the current variables and sizes, and indicates if they have imaginary parts.
<code>:</code>	Colon; generates an array having regularly spaced elements.
<code>,</code>	Comma; separates elements of an array.
<code>;</code>	Semicolon; suppresses screen printing; also denotes a new row in an array.
<code>...</code>	Ellipsis; continues a line.

Special variables and constants

Table 1.1–4, Page 14

Command	Description
ans	Temporary variable containing the most recent answer.
eps	Specifies the accuracy of floating point precision.
i,j	The imaginary unit $\sqrt{-1}$.
Inf	Infinity.
NaN	Indicates an undefined numerical result.
pi	The number π .

Complex Number Operations, Pages 14-15

- The number $c_1 = 1 - 2i$ is entered as follows:
$$c1 = 1 - 2i.$$
- An asterisk is not needed between i or j and a number, although it is required with a variable, such as $c2 = 5 - i * c1$.

- Be careful. The expressions

$$y = 7/2 * i$$

and

$$x = 7/2i$$

give two different results:

$$y = (7/2)i = 3.5i$$

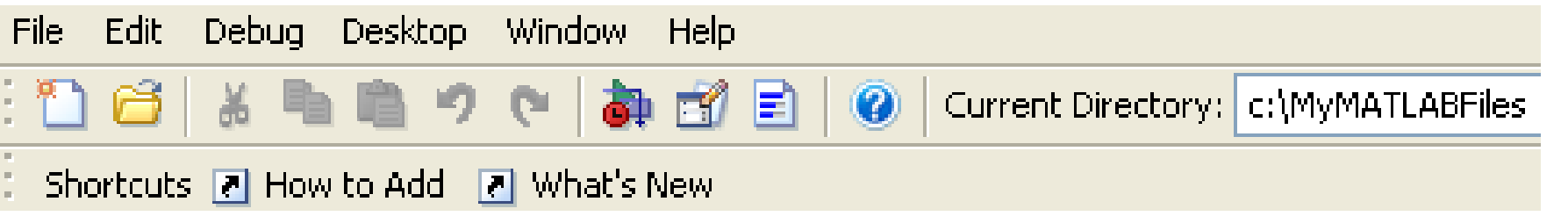
and

$$x = 7/(2i) = -3.5i.$$

Numeric display formats. Table 1.1–5, Page 15

Command	Description and example
<code>format short</code>	Four decimal digits (the default); 13.6745.
<code>format long</code>	16 digits; 17.27484029463547.
<code>format short e</code>	Five digits (four decimals) plus exponent; 6.3792e+03.
<code>format long e</code>	16 digits (15 decimals) plus exponent; 6.379243784781294e-04.

The Desktop Menus and Toolbar. Figure 1.2-1, page 16



Arrays

- The numbers 0, 0.1, 0.2, ..., 10 can be assigned to the variable u by typing `u = 0:0.1:10`.
- To compute $w = 5 \sin u$ for $u = 0, 0.1, 0.2, \dots, 10$, the session is;

```
>>u = 0:0.1:10;
```

```
>>w = 5*sin(u);
```

- The single line, `w = 5*sin(u)`, computed the formula $w = 5 \sin u$ 101 times.

Array Index

```
>>u(7)
ans =
    0.6000
>>w(7)
ans =
    2.8232
```

- Use the `length` function to determine how many values are in an array.

```
>>m = length(w)
m =
    101
```

Polynomial Roots, Page 20

To find the roots of $x^3 - 7x^2 + 40x - 34 = 0$, the session is

```
>>a = [1,-7,40,-34];  
>>roots(a)  
ans =  
    3.0000 + 5.000i  
    3.0000 - 5.000i  
    1.0000
```

The roots are $x = 1$ and $x = 3 \pm 5i$.

Some commonly used mathematical functions

Table 1.3–1, Page 21

Function	MATLAB syntax ¹
e^x	<code>exp(x)</code>
\sqrt{x}	<code>sqrt(x)</code>
$\ln x$	<code>log(x)</code>
$\log_{10} x$	<code>log10(x)</code>
$\cos x$	<code>cos(x)</code>
$\sin x$	<code>sin(x)</code>
$\tan x$	<code>tan(x)</code>
$\cos^{-1} x$	<code>acos(x)</code>
$\sin^{-1} x$	<code>asin(x)</code>
$\tan^{-1} x$	<code>atan(x)</code>

¹The MATLAB trigonometric functions use radian measure.

When you type `problem1`,

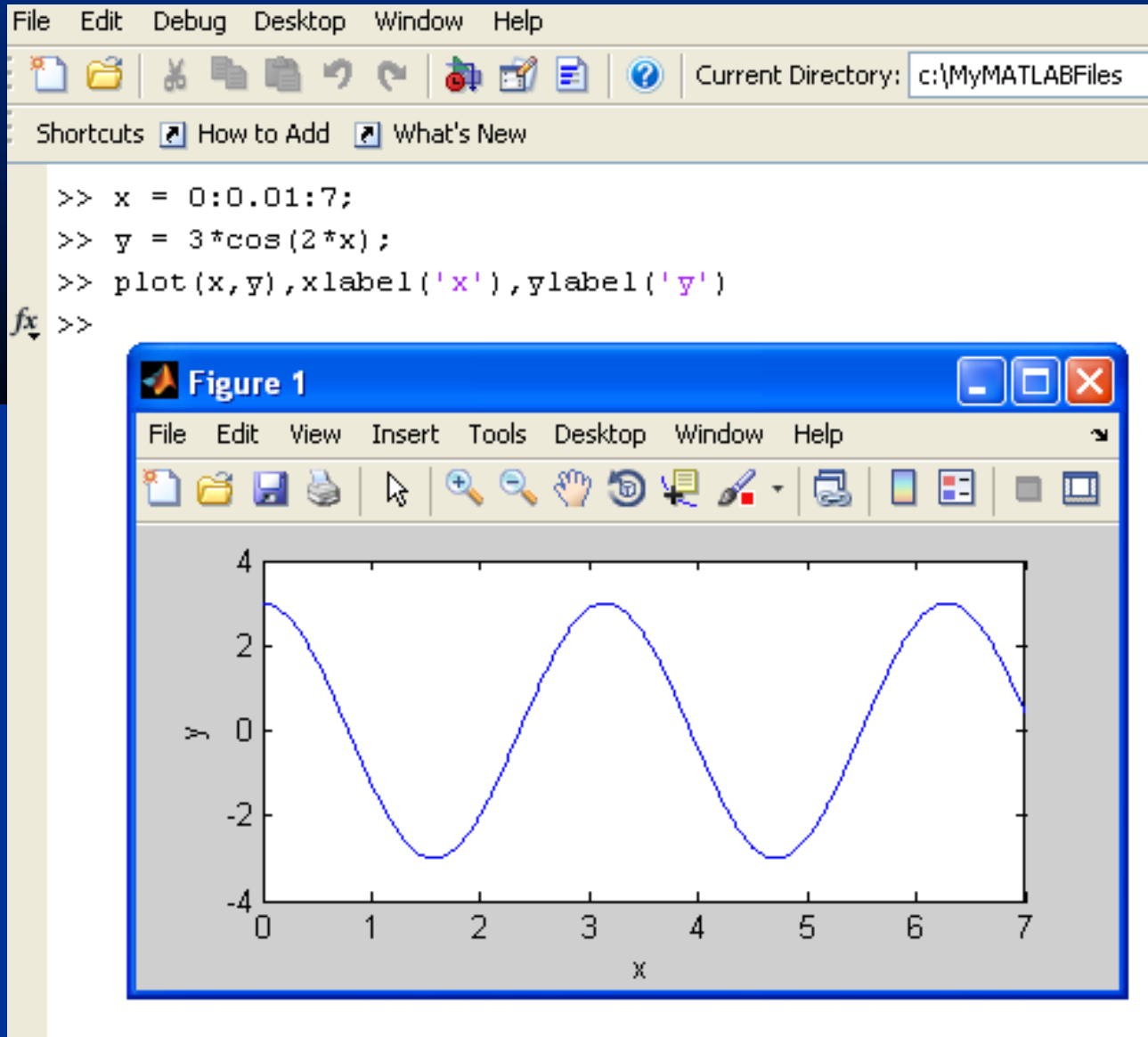
1. MATLAB first checks to see if `problem1` is a variable and if so, displays its value.
2. If not, MATLAB then checks to see if `problem1` is one of its own commands, and executes it if it is.
3. If not, MATLAB then looks in the current directory for a file named `problem1.m` and executes `problem1` if it finds it.
4. If not, MATLAB then searches the directories in its search path, in order, for `problem1.m` and then executes it if found.

System, directory, and file commands

Table 1.3–2, Page 23

Command	Description
<code>addpath dirname</code>	Adds the directory <code>dirname</code> to the search path.
<code>cd dirname</code>	Changes the current directory to <code>dirname</code> .
<code>dir</code>	Lists all files in the current directory.
<code>dir dirname</code>	Lists all the files in the directory <code>dirname</code> .
<code>path</code>	Displays the MATLAB search path.
<code>pathtool</code>	Starts the Set Path tool.
<code>pwd</code>	Displays the current directory.
<code>rmpath dirname</code>	Removes the directory <code>dirname</code> from the search path.
<code>what</code>	Lists the MATLAB-specific files found in the current working directory. Most data files and other non-MATLAB files are not listed. Use <code>dir</code> to get a list of all files.
<code>what dirname</code>	Lists the MATLAB-specific files in directory <code>dirname</code> .

A graphics window showing a plot. Figure 1.3-1, page 24.



Some MATLAB plotting commands

Table 1.3–3, Page 25

Command	Description
<code>[x,y] = ginput(n)</code>	Enables the mouse to get n points from a plot, and returns the x and y coordinates in the vectors x and y , which have a length n .
<code>grid</code>	Puts grid lines on the plot.
<code>gtext('text')</code>	Enables placement of text with the mouse.
<code>plot(x,y)</code>	Generates a plot of the array y versus the array x on rectilinear axes.
<code>title('text')</code>	Puts text in a title at the top of the plot.
<code>xlabel('text')</code>	Adds a text label to the horizontal axis (the abscissa).
<code>ylabel('text')</code>	Adds a text label to the vertical axis (the ordinate).

Linear Algebraic Equations, Page 26

$$6x + 12y + 4z = 70$$

$$7x - 2y + 3z = 5$$

$$2x + 8y - 9z = 64$$

```
>>A = [6,12,4;7,-2,3;2,8,-9];
```

```
>>B = [70;5;64];
```

```
>>Solution = A\B
```

```
Solution =
```

```
    3
```

```
    5
```

```
   -2
```

The solution is $x = 3$, $y = 5$, and $z = -2$.

You can perform operations in MATLAB in two ways:

1. In the interactive mode, in which all commands are entered directly in the Command window, or
2. By running a MATLAB program stored in *script* file. This type of file contains MATLAB commands, so running it is equivalent to typing all the commands—one at a time—at the Command window prompt. You can run the file by typing its name at the Command window prompt.

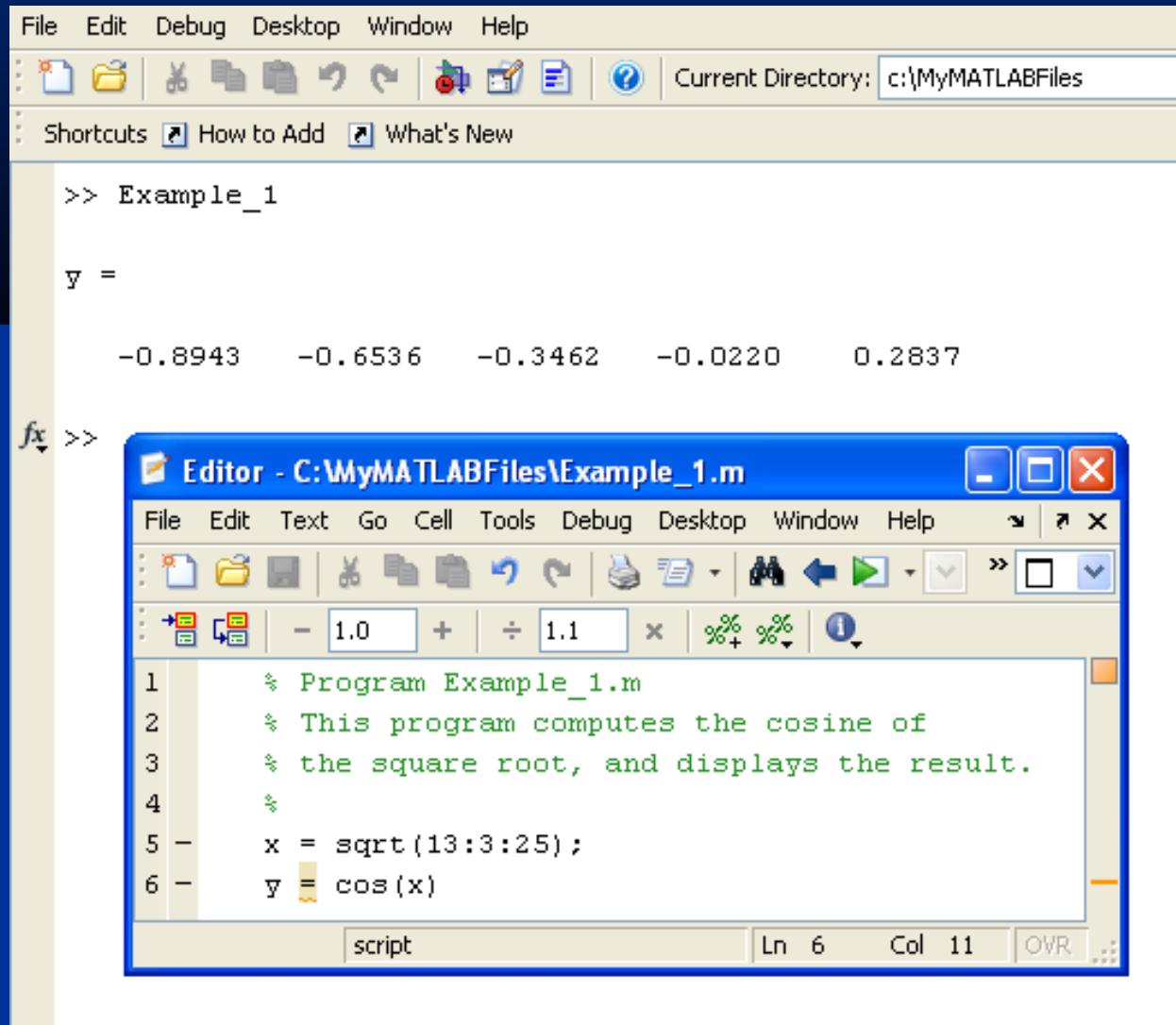
COMMENTS

The comment symbol may be put anywhere in the line. MATLAB ignores everything to the right of the % symbol. For example,

```
>>% This is a comment.  
>>x = 2+3 % So is this.  
x =  
    5
```

Note that the portion of the line before the % sign is executed to compute x.

The MATLAB Command window with the Editor/Debugger open. Figure 1.4–1, Page 28



Keep in mind when using script files:

1. The name of a script file must begin with a letter, and may include digits and the underscore character, up to 63 characters.
2. Do not give a script file the same name as a variable.
3. Do not give a script file the same name as a MATLAB command or function. You can check to see if a command, function or file name already exists by using the `exist` command.

Debugging Script Files

Program errors usually fall into one of the following categories.

1. Syntax errors such as omitting a parenthesis or comma, or spelling a command name incorrectly. MATLAB usually detects the more obvious errors and displays a message describing the error and its location.
2. Errors due to an incorrect mathematical procedure, called *runtime errors*. Their occurrence often depends on the particular input data. A common example is division by zero.

To locate program errors, try the following:

1. Test your program with a simple version of the problem which can be checked by hand.
2. Display any intermediate calculations by removing semicolons at the end of statements.
3. Use the debugging features of the Editor/Debugger.

Programming Style

1. *Comments section*

a. The name of the program and any key words in the first line.

b. The date created, and the creators' names in the second line.

c. The definitions of the variable names for every input and output variable. Include definitions of variables used in the calculations and *units of measurement for all input and all output variables!*

d. The name of every user-defined function called by the program.

Programming Style (continued)

2. *Input section* Include input data and/or the input functions and comments for documentation.
3. *Calculation section*
4. *Output section* This section might contain functions for displaying the output on the screen.

Some Input/output commands

From Table 1.4–1, Page 31

Command	Description
<code>disp(A)</code>	Displays the contents, but not the name, of the array A.
<code>disp('text')</code>	Displays the text string enclosed within single quotes.
<code>x = input('text')</code>	Displays the text in quotes, waits for user input from the keyboard, and stores the value in x.
<code>x = input('text','s')</code>	Displays the text in quotes, waits for user input from the keyboard, and stores the input as a string in x.

Example of a Script File, Page 32

Problem:

The speed v of a falling object dropped with no initial velocity is given as a function of time t by $v = gt$.

Plot v as a function of t for $0 < t < t_{final}$, where t_{final} is the final time entered by the user.

Example of a Script File (continued)

```
% Program falling_speed.m:  
% Plots speed of a falling object.  
% Created on March 1, 2009 by W. Palm  
%  
% Input Variable:  
% tfinal = final time (in seconds)  
%  
% Output Variables:  
% t = array of times at which speed is  
% computed (in seconds)  
% v = array of speeds (meters/second)  
%
```


Example of a Script File (continued)

```
% Parameter Value:  
g = 9.81; % Acceleration in SI units  
%  
% Input section:  
tfinal = input('Enter final time in  
seconds: ');  
%
```

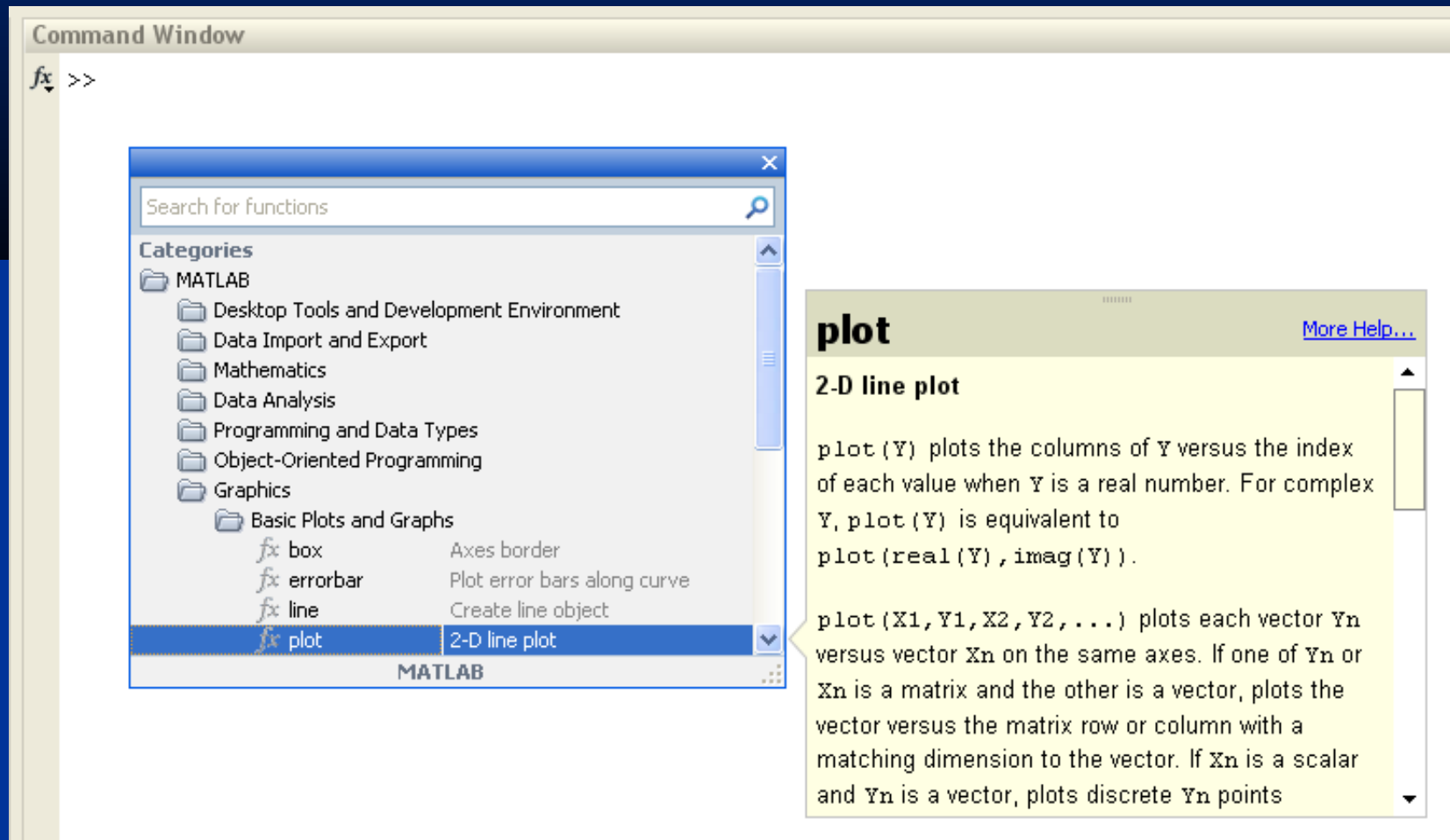
Example of a Script File (continued)

```
% Calculation section:
dt = tfinal/500;
% Create an array of 501 time values.
t = 0:dt:tfinal;
% Compute speed values.
v = g*t;
%
% Output section:
Plot(t,v),xlabel('t (s)'),ylabel('v m/s')
```

Getting Help From the Textbook

- Throughout each chapter margin notes identify where key terms are introduced.
- Each chapter contains tables summarizing the MATLAB commands introduced in that chapter.
- At the end of each chapter is a summary guide to the commands covered in that chapter.
- Appendix A contains tables of MATLAB commands, grouped by category, with the appropriate page references.
- There are four indexes. The first lists MATLAB commands and symbols, the second lists Simulink blocks, the third lists MuPAD commands, and the fourth lists topics.

Getting Help From MATLAB: The Function Browser after `plot` has been selected (Figure 1.5-1, page 33)



The screenshot shows the MATLAB Command Window with the Function Browser open. The Command Window contains the prompt `fx >>`. The Function Browser window has a search bar and a tree view of categories. The 'plot' function is selected, and its help text is displayed in a separate window.

Command Window

```
fx >>
```

Function Browser

Search for functions

Categories

- MATLAB
 - Desktop Tools and Development Environment
 - Data Import and Export
 - Mathematics
 - Data Analysis
 - Programming and Data Types
 - Object-Oriented Programming
 - Graphics
 - Basic Plots and Graphs
 - fx box Axes border
 - fx errorbar Plot error bars along curve
 - fx line Create line object
 - fx plot 2-D line plot**

plot [More Help...](#)

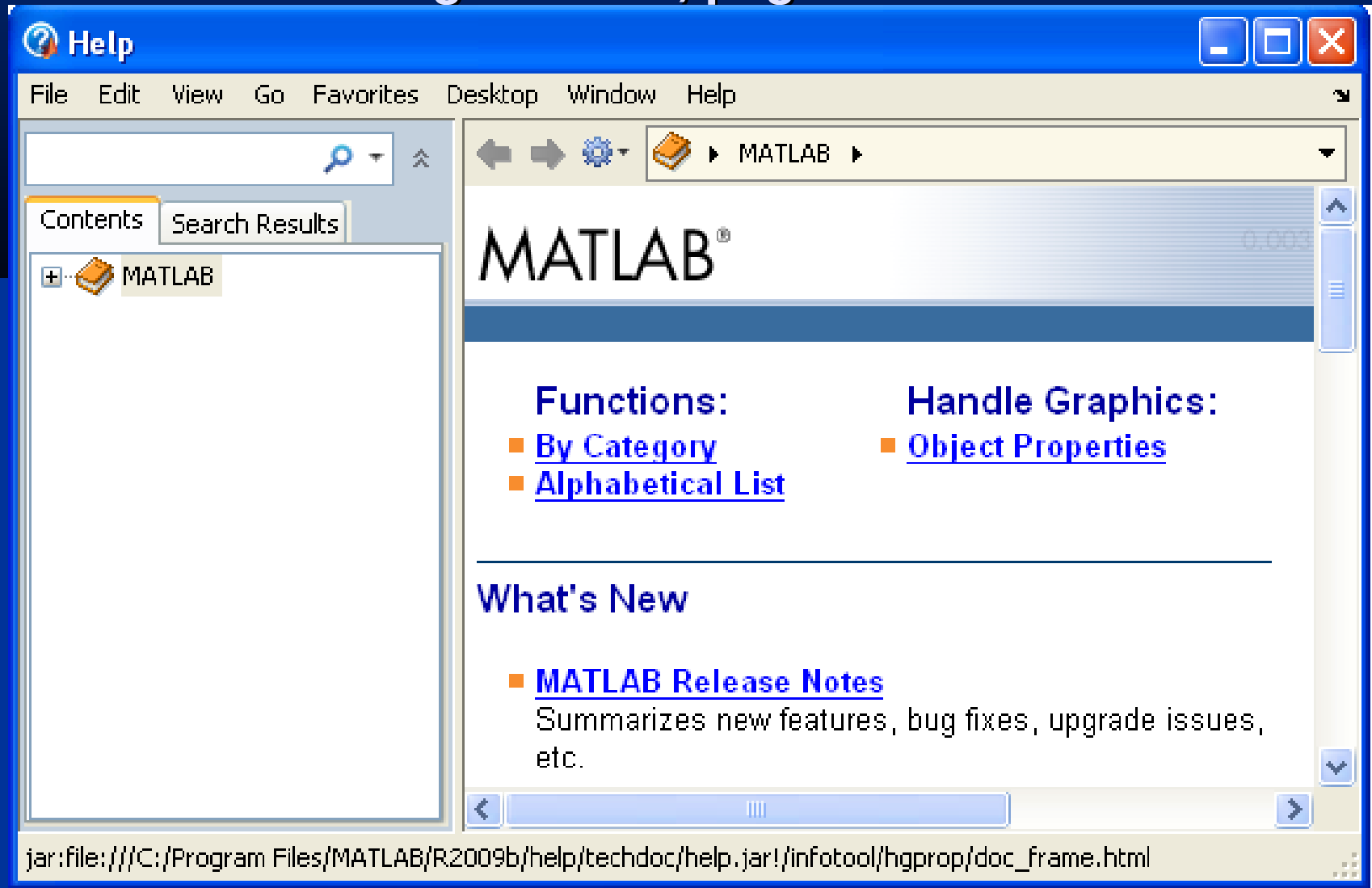
2-D line plot

`plot(Y)` plots the columns of `Y` versus the index of each value when `Y` is a real number. For complex `Y`, `plot(Y)` is equivalent to `plot(real(Y), imag(Y))`.

`plot(X1, Y1, X2, Y2, ...)` plots each vector `Yn` versus vector `Xn` on the same axes. If one of `Yn` or `Xn` is a matrix and the other is a vector, plots the vector versus the matrix row or column with a matching dimension to the vector. If `Xn` is a scalar and `Yn` is a vector, plots discrete `Yn` points

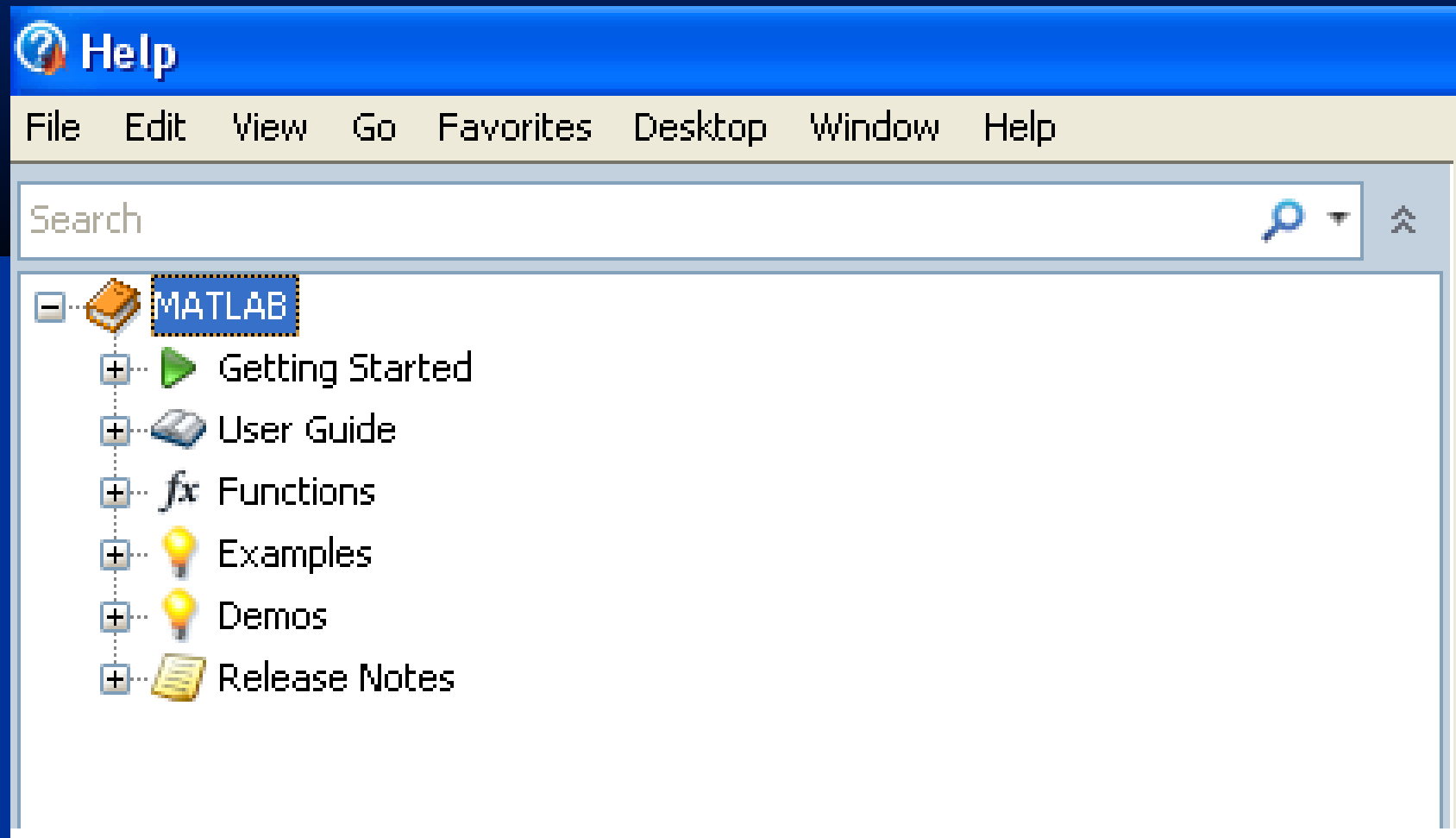
The MATLAB Help Browser

Figure 1.5-2, page 34



The Help Navigator

Fig. 1.5-3, page 35



MATLAB Help Functions, From Table 1.5-1, page 38

- `help funcname`: Displays in the Command window a description of the specified function *funcname*.
- `lookfor topic`: Looks for the string *topic* in the first comment line (the H1 line) of the HELP text of all M-files found on MATLABPATH (including private directories), and displays the H1 line for all files in which a match occurs.
- `doc funcname`: Opens the Help Browser to the reference page for the specified function *funcname*, providing a description, additional remarks, and examples.

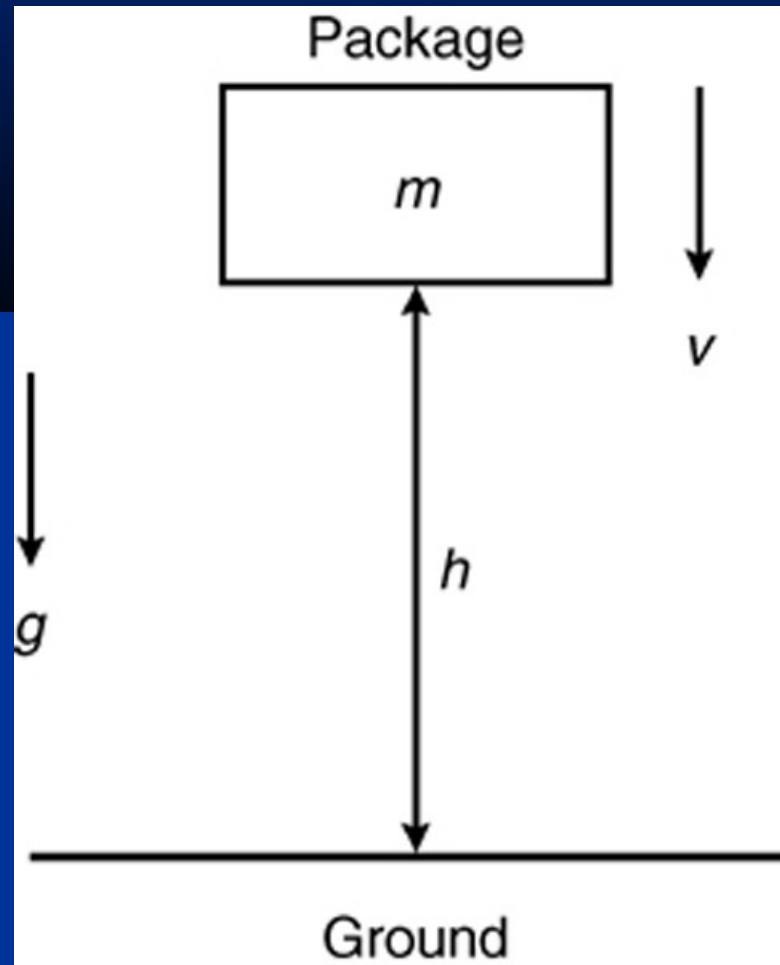
Steps in engineering problem solving

Table 1.6–1,
Page 39

1. Understand the purpose of the problem.
2. Collect the known information. Realize that some of it might later be found unnecessary.
3. Determine what information you must find.
4. Simplify the problem only enough to obtain the required information. State any assumptions you make.
5. Draw a sketch and label any necessary variables.
6. Determine which fundamental principles are applicable.
7. Think generally about your proposed solution approach and consider other approaches before proceeding with the details.
8. Label each step in the solution process.
9. If you solve the problem with a program, hand check the results using a simple version of the problem. Checking the dimensions and units and printing the results of intermediate steps in the calculation sequence can uncover mistakes.
10. Perform a “reality check” on your answer. Does it make sense? Estimate the range of the expected result and compare it with your answer. Do not state the answer with greater precision than is justified by any of the following:
 - (a) The precision of the given information.
 - (b) The simplifying assumptions.
 - (c) The requirements of the problem.Interpret the mathematics. If the mathematics produces multiple answers, do not discard some of them without considering what they mean. The mathematics might be trying to tell you something, and you might miss an opportunity to discover more about the problem.

Sketch of the dropped-package problem.

Figure 1.6–1, page 41

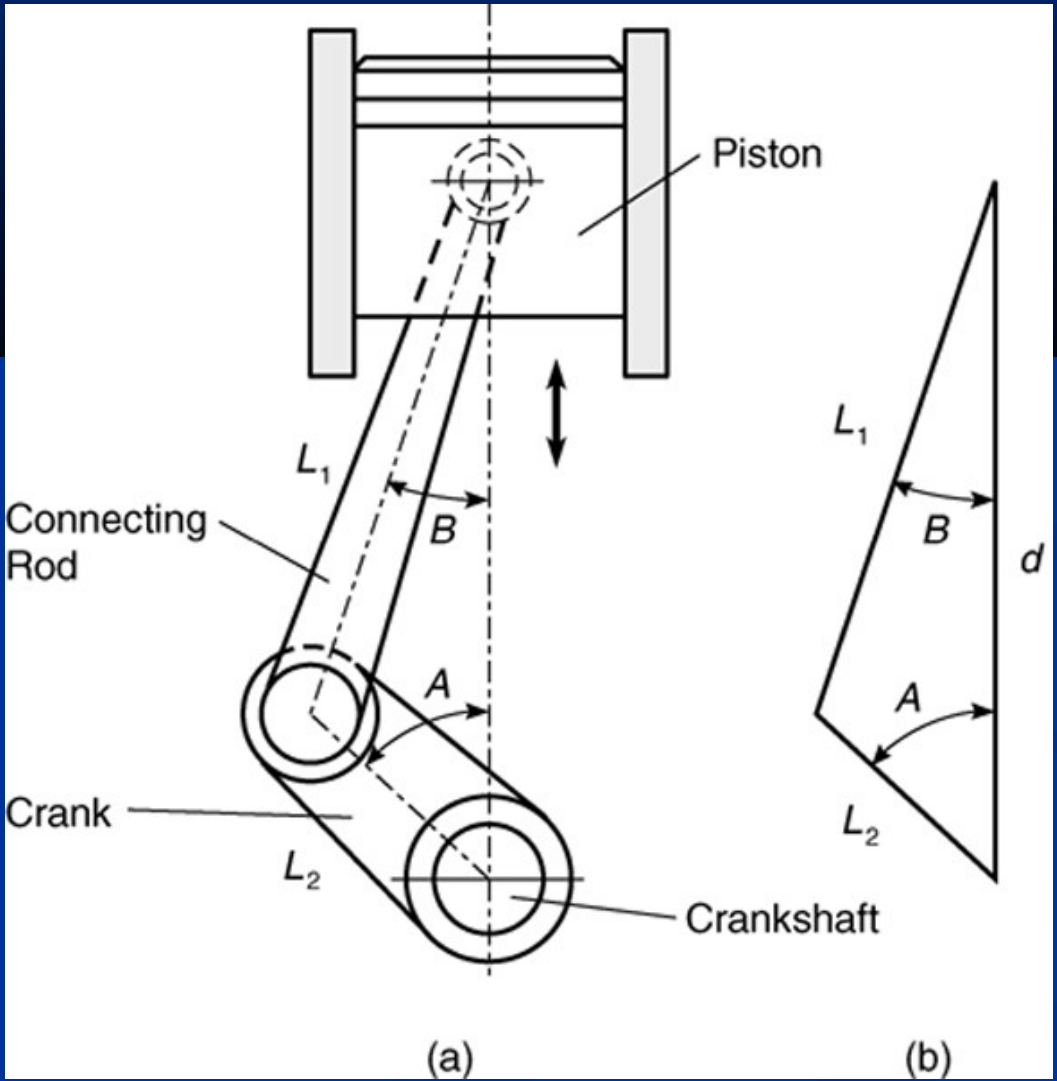


Steps for developing a computer solution

Table 1.6–2, page 42

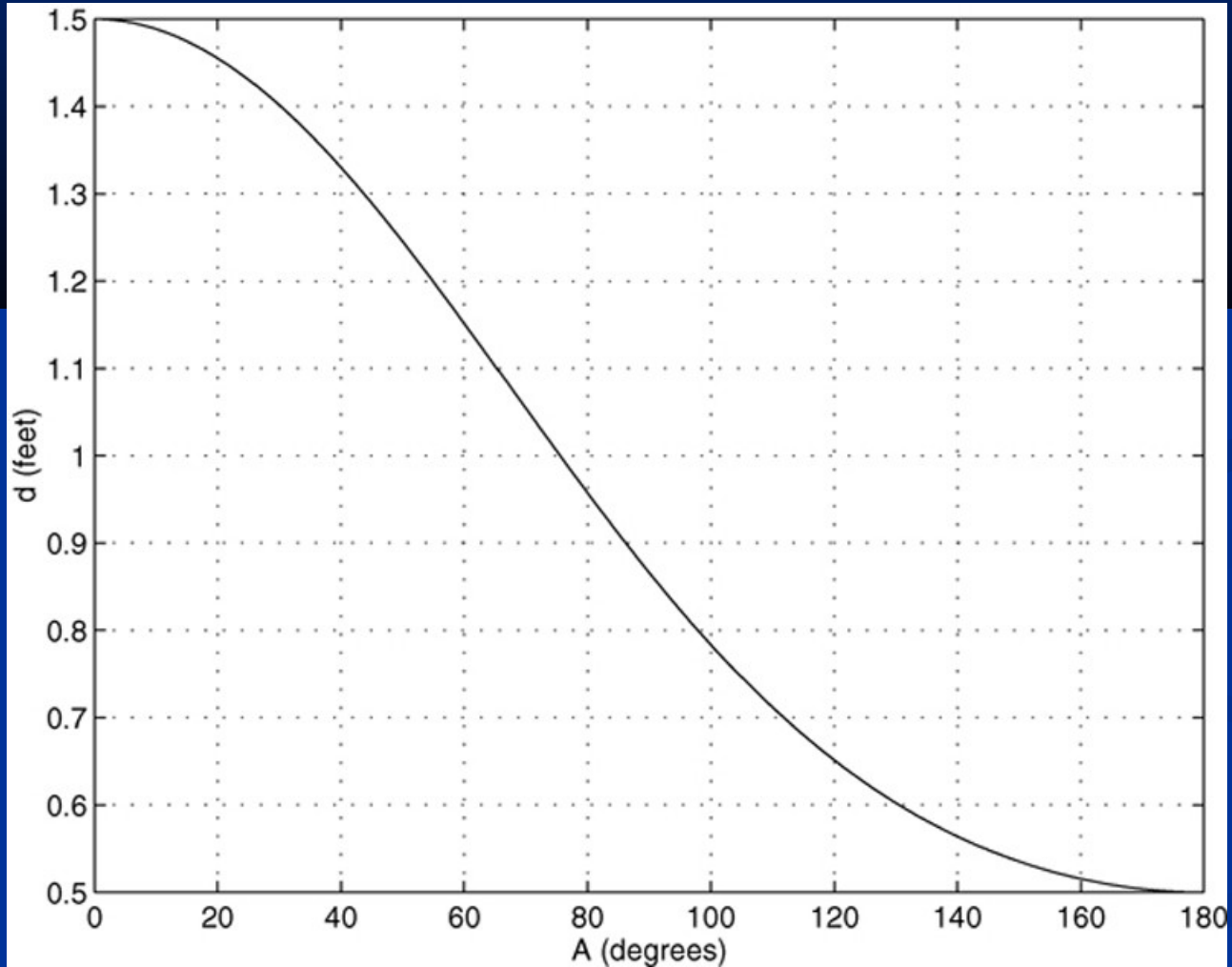
1. State the problem concisely.
2. Specify the data to be used by the program. This is the “input.”
3. Specify the information to be generated by the program. This is the “output.”
4. Work through the solution steps by hand or with a calculator; use a simpler set of data if necessary.
5. Write and run the program.
6. Check the output of the program with your hand solution.
7. Run the program with your input data and perform a reality check on the output.
8. If you will use the program as a general tool in the future, test it by running it for a range of reasonable data values; perform a reality check on the results.

A piston, connecting rod, and crank for an internal combustion engine. Figure 1.6–2, 43



Plot of the piston motion versus crank angle.

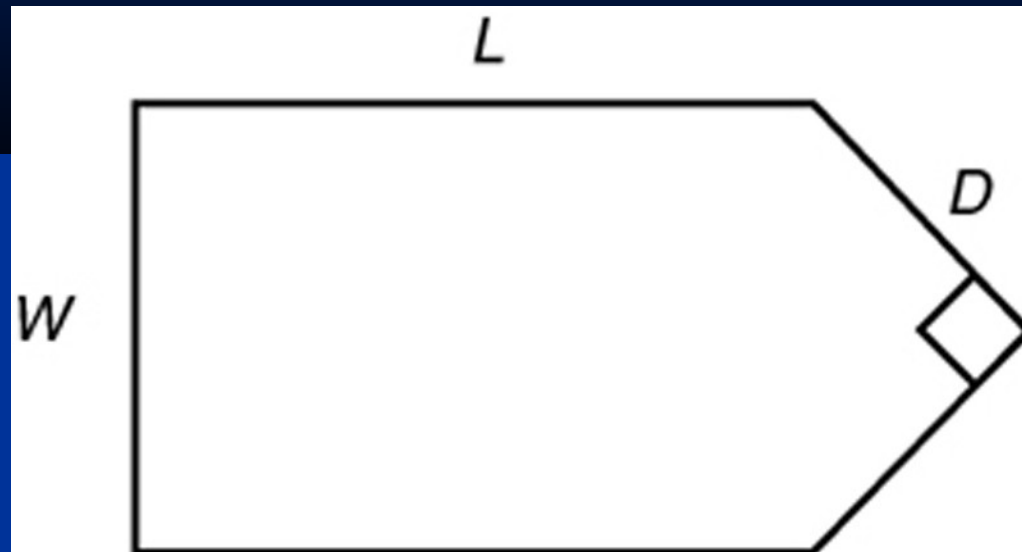
Figure 1.6–3, page 45



Key Terms with Page References

Argument, 8
Array, 19
Array index, 19
ASCII files, 21
Assignment operator, 10
Command window, 6
Comment, 27
Current directory, 16
Data file, 21
Data marker, 25
Debugging, 29
Desktop, 5
Graphics window, 23
MAT-files, 20
Model, 39
Overlay plot, 24
Path, 22
Precedence, 9
Scalar, 8
Script file, 27
Search path, 22
Session, 7
String variable, 31
Variable, 7
Workspace, 11

Homework Problem 25, Figure P25, page 50



Homework Problem 29, Figure P29, page 51

