The McGraw-Hill Companies

PowerPoint to accompany

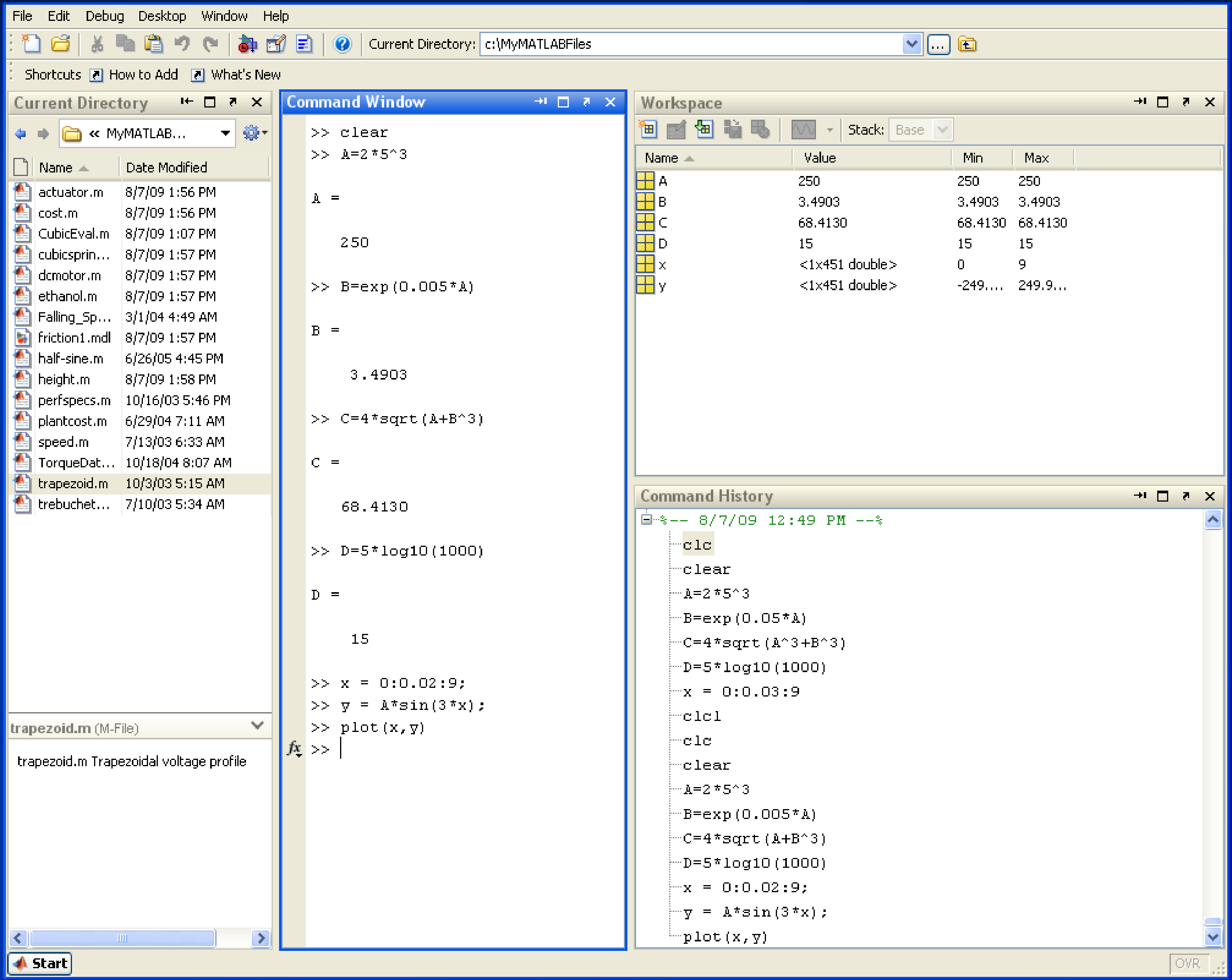
Introduction to MATLAB for Engineers, Third Edition

William J. Palm III

Chapter 1

An Overview of MATLAB®

The Default MATLAB



Name	Date Modified
actuator.m	8/7/09 1:56 PM
cost.m	8/7/09 1:56 PM
CubicEval.m	8/7/09 1:07 PM
cubicsprin...	8/7/09 1:57 PM
dcmotor.m	8/7/09 1:57 PM
ethanol.m	8/7/09 1:57 PM
Falling_Sp...	3/1/04 4:49 AM
friction1.mdl	8/7/09 1:57 PM
half-sine.m	6/26/05 4:45 PM
height.m	8/7/09 1:58 PM
perfspecs.m	10/16/03 5:46 PM
plantcost.m	6/29/04 7:11 AM
speed.m	7/13/03 6:33 AM
TorqueDat...	10/18/04 8:07 AM
trapezoid.m	10/3/03 5:15 AM
trebuchet...	7/10/03 5:34 AM

trapezoid.m (M-File)

trapezoid.m Trapezoidal voltage profile

```
>> 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)
>> |
```

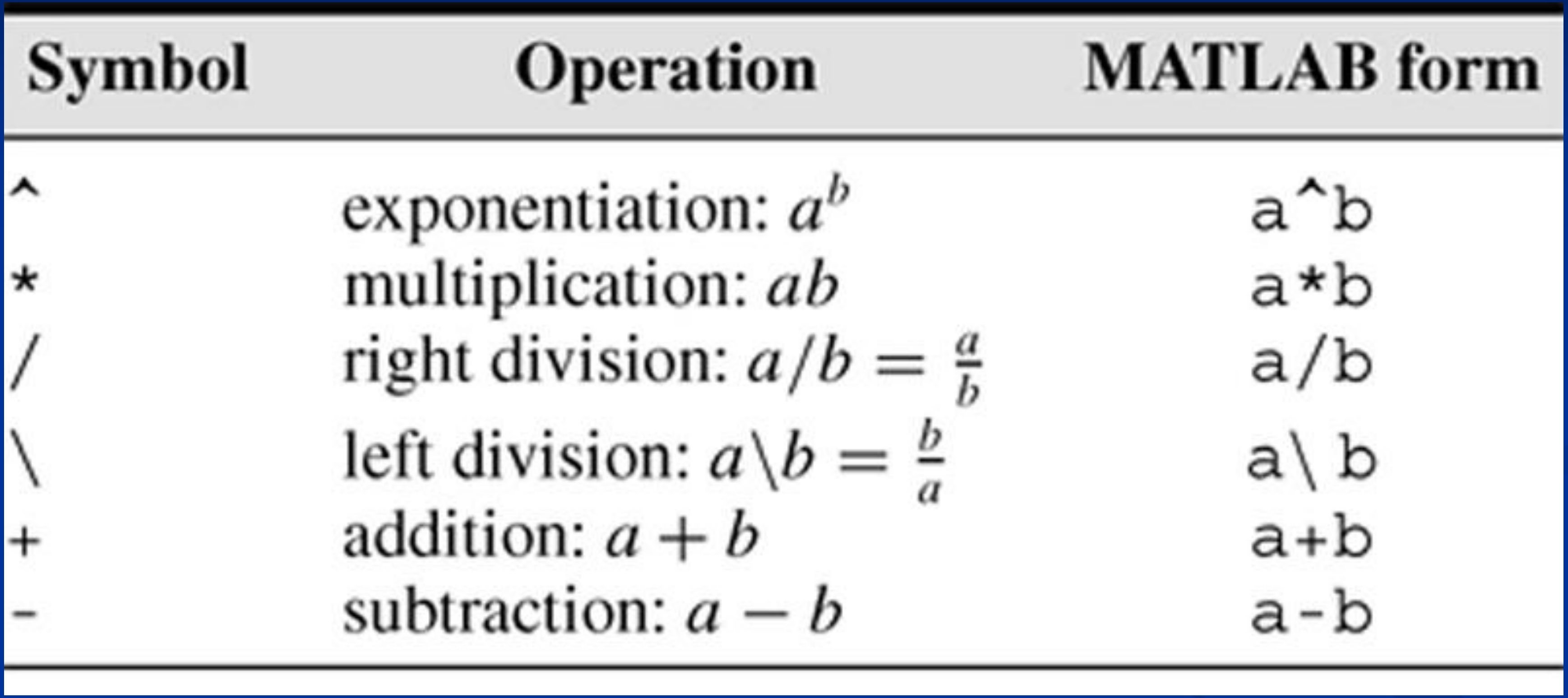
Name	Value	Min	Max
A	250	250	250
B	3.4903	3.4903	3.4903
C	68.4130	68.4130	68.4130
D	15	15	15
x	<1x451 double>	0	9
y	<1x451 double>	-249....	249.9...

```
%-- 8/7/09 12:49 PM --%
  clc
  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
  clcl
  clc
  clear
  A=2*5^3
  B=exp(0.005*A)
  C=4*sqrt(A+B^3)
  D=5*log10(1000)
  x = 0:0.02:9;
  y = A*sin(3*x);
  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

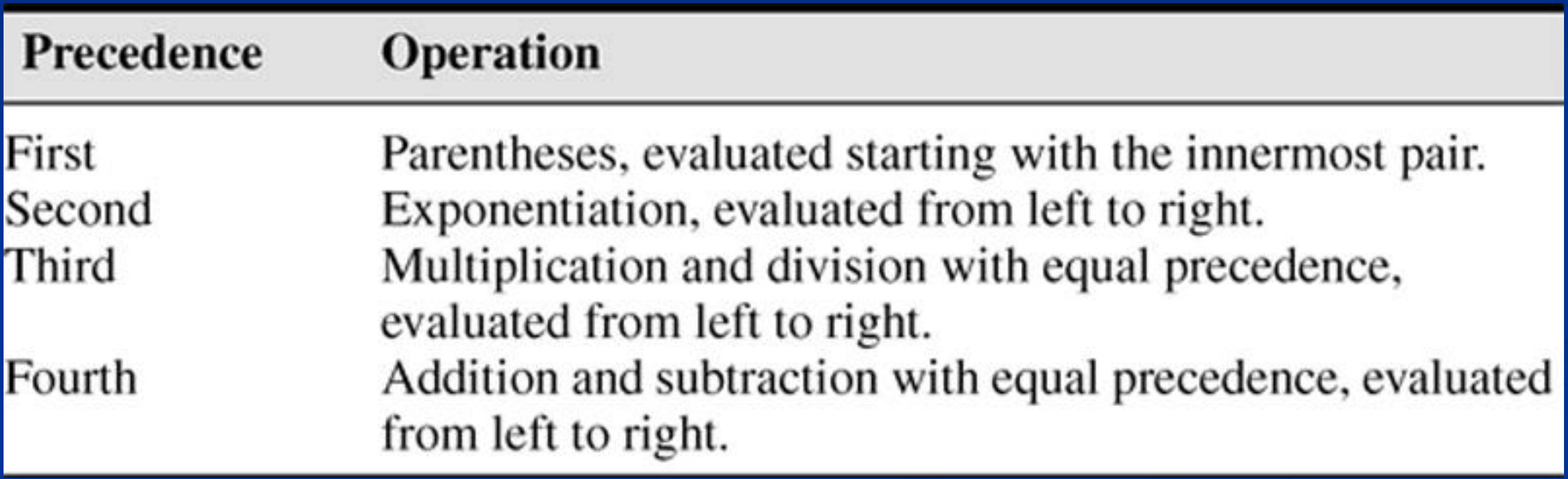


Symbol	Operation	MATLAB form
^	exponentiation: a^b	`a^b`
*	multiplication: ab	`a*b`
/	right division: $a/b = \frac{a}{b}$	`a/b`
\	left division: $a\backslash b = \frac{b}{a}$	`a\ b`
+	addition: $a + b$	`a+b`
-	subtraction: $a - b$	`a-b`

An Example Session

```
>> 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



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

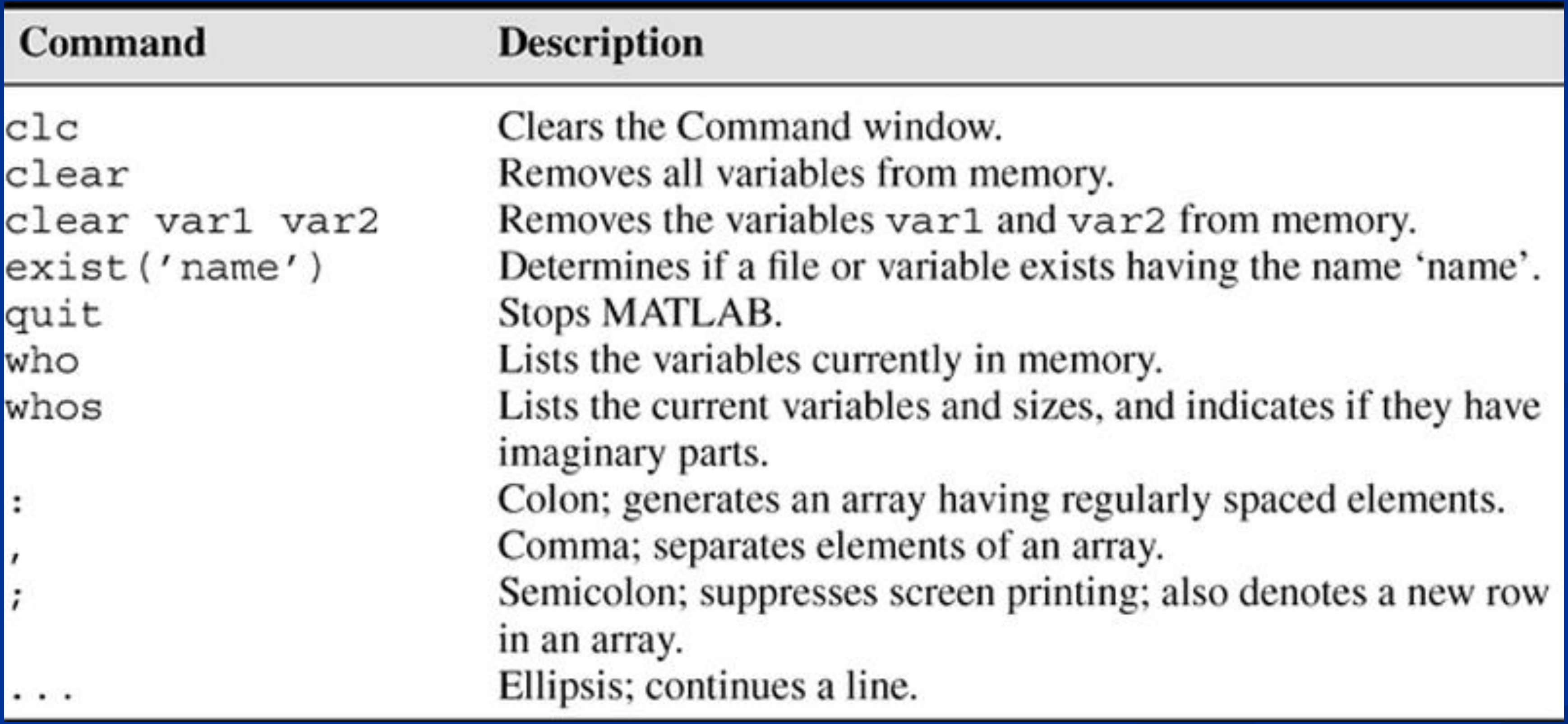
```
>> 8 + 3*5
ans =
     23
>> 8 + (3*5)
ans =
     23
>>(8 + 3)*5
ans =
     55
>>4^2128/4*2
ans =
     0
>>4^212 8/(4*2)
ans =
     3
```

Examples of Precedence,

```
>> 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

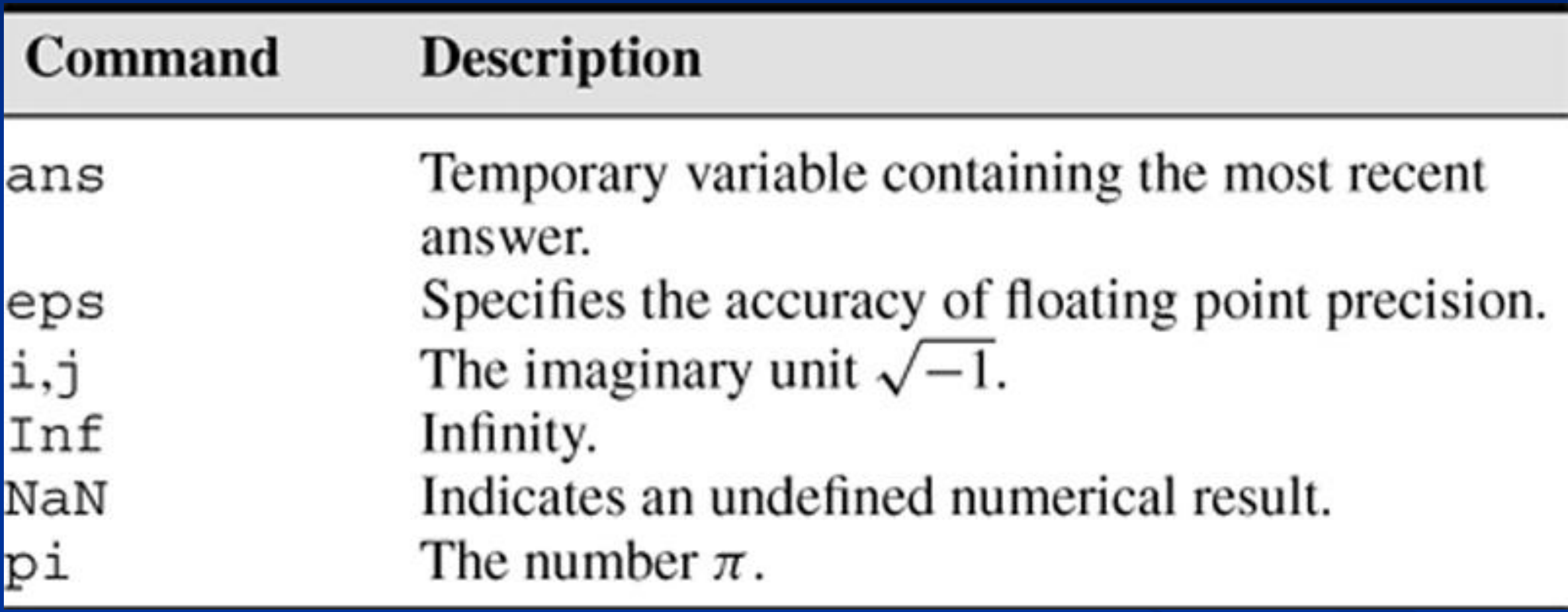
Table



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

Special variables and constants

Table



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,

- The number $c_1 = 1 - 2i$ is entered as follows:
  `c1 = 12i`.
- 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
```

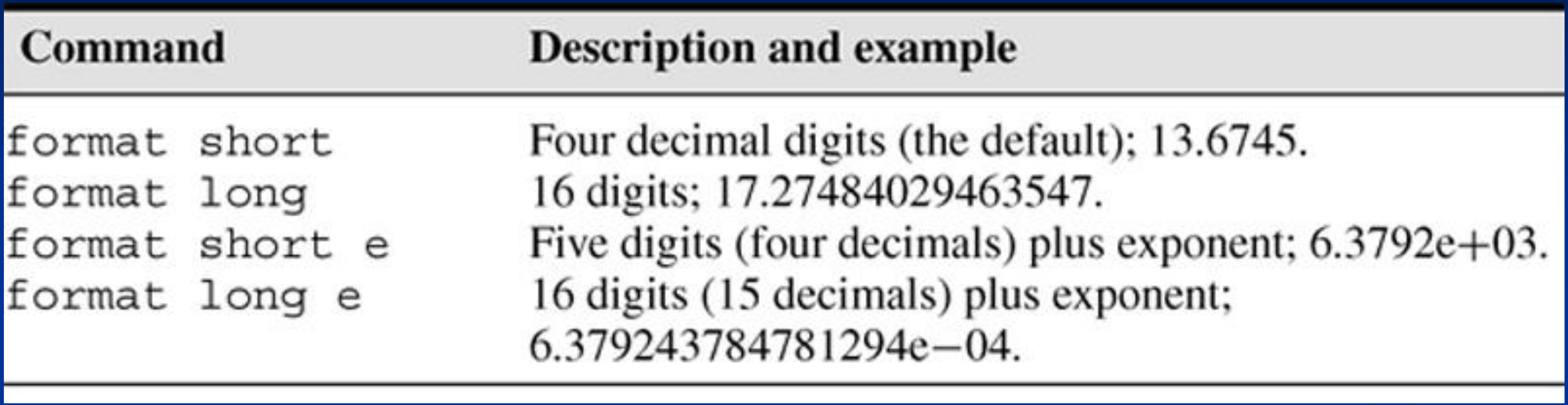
give two different results:

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

and

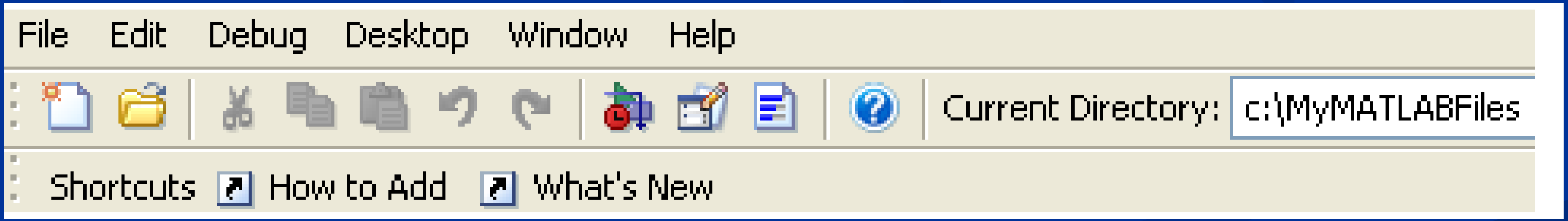
$x = 7/(2i) = -3.5i$.

Numeric display formats. Table



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

The Desktop Menus and Toolbar. Figure



Current Directory: c:\MyMATLABFiles

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 $u = 0, 0.1, 0.2, \ldots, 10$, 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

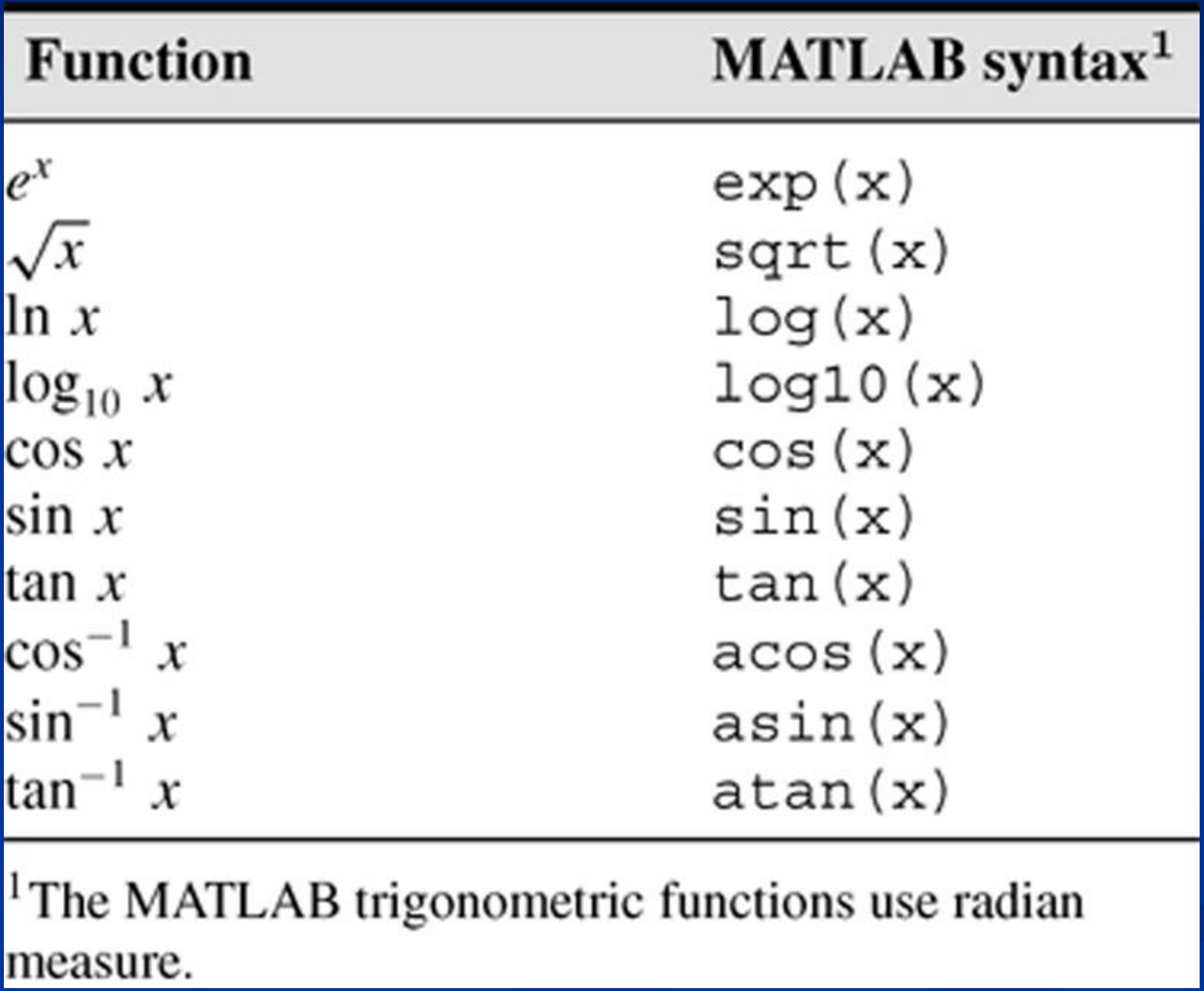
To find the roots of $x^3 - 7x^2 + 40x - 34 = 0$, the session is

```
>>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

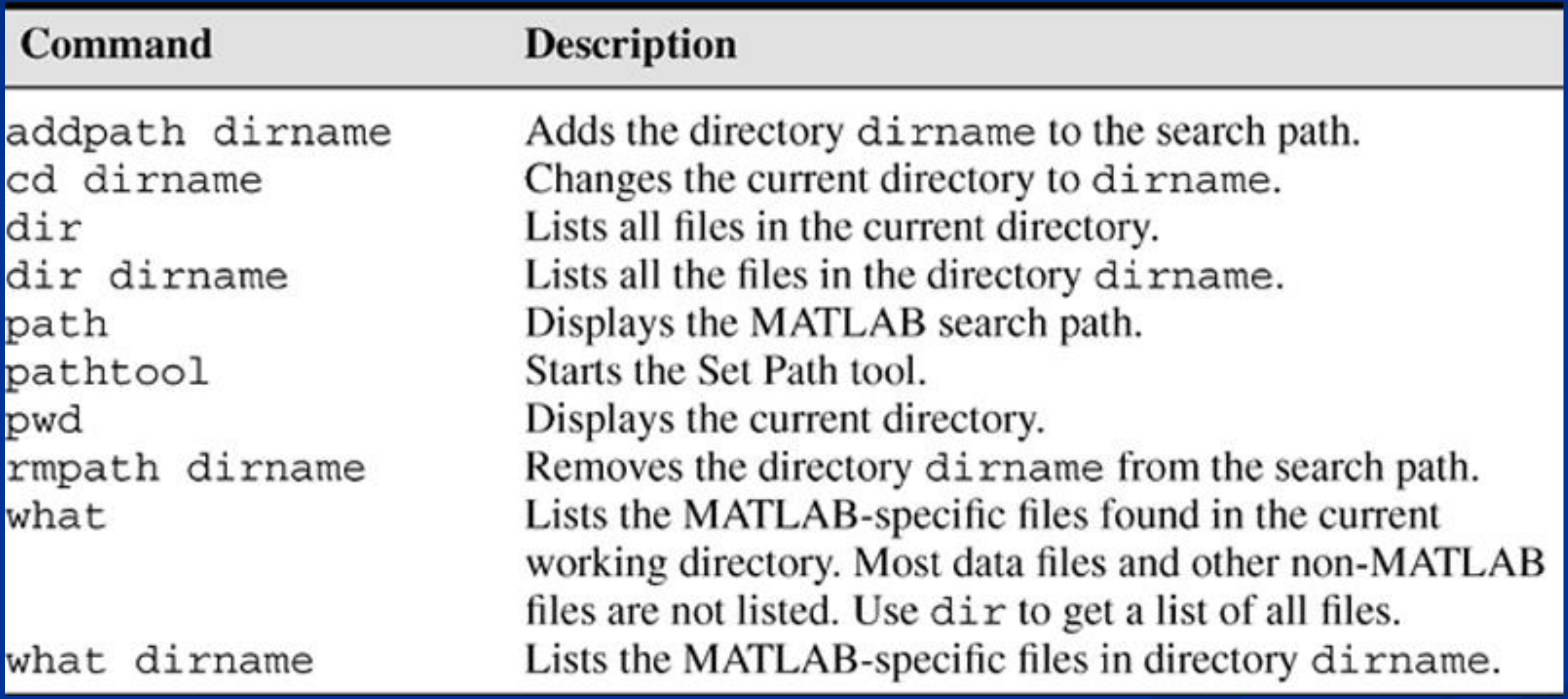


Function	MATLAB syntax[1]
e^x	`exp(x)`
$\sqrt{x}$	`sqrt(x)`
$\ln x$	`log(x)`
$\log_{10} x$	`log10(x)`
$\cos x$	`cos(x)`
$\sin x$	`sin(x)`
$\tan x$	`tan(x)`
$\cos^{-1} x$	`acos(x)`
$\sin^{-1} x$	`asin(x)`
$\tan^{-1} x$	`atan(x)`

[1]The MATLAB trigonometric functions use radian measure.

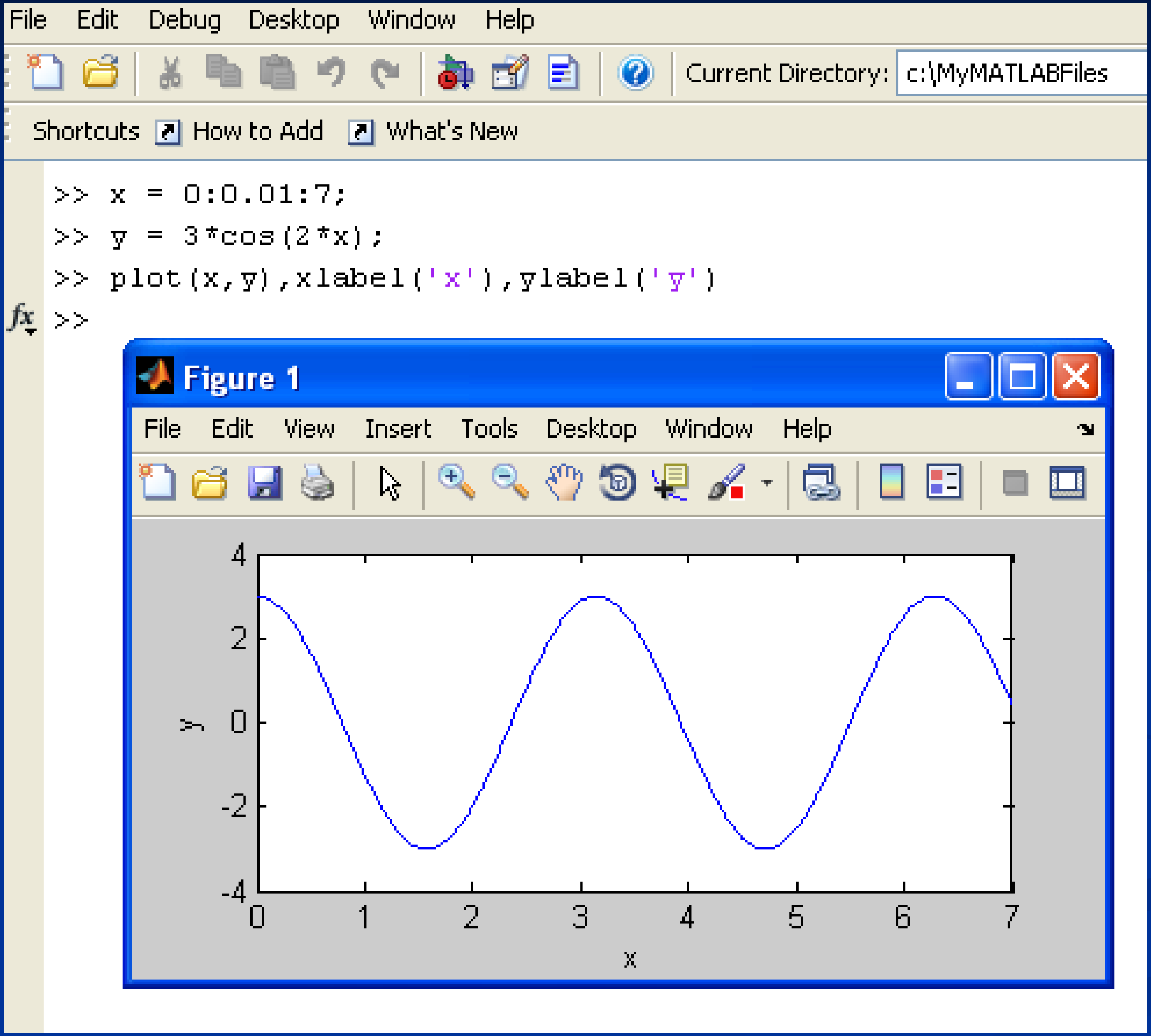
System, directory, and file commands

Table



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

A graphics window showing a plot. Figure



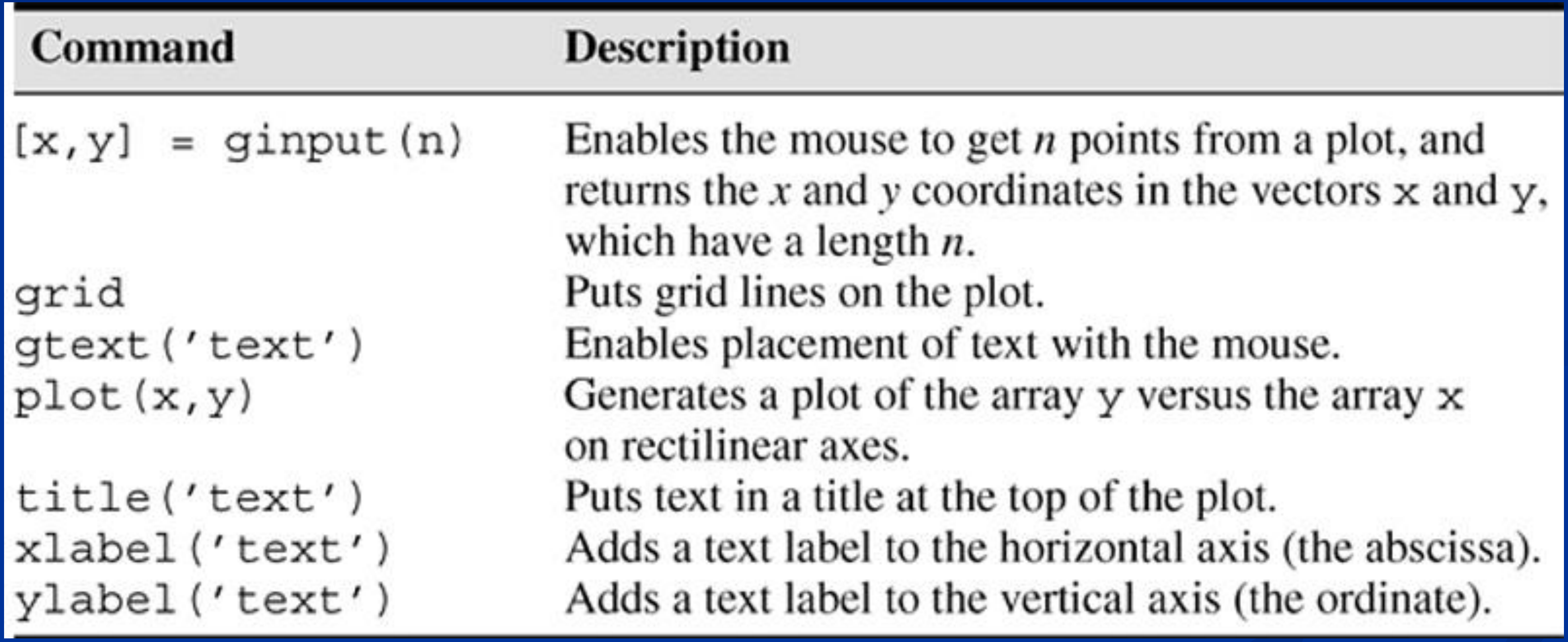
```
>> x = 0:0.01:7;
>> y = 3*cos(2*x);
>> plot(x,y),xlabel('x'),ylabel('y')
>>
```

File Edit Debug Desktop Window Help
Current Directory: c:\MyMATLABFiles
Shortcuts How to Add What's New
Figure 1
File Edit View Insert Tools Desktop Window Help
4
2
0
-2
-4
y
0 1 2 3 4 5 6 7
x

```
>> plot(x,y,'o')

>> xlabel('x ')

>> ylabel('y')

>> title('Plot of x and y')
```

Some MATLAB plotting commands

Table



Command	Description
`[x,y] = ginput(n)`	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*.
`grid`	Puts grid lines on the plot.
`gtext('text')`	Enables placement of text with the mouse.
`plot(x,y)`	Generates a plot of the array `y` versus the array `x` on rectilinear axes.
`title('text')`	Puts text in a title at the top of the plot.
`xlabel('text')`	Adds a text label to the horizontal axis (the abscissa).
`ylabel('text')`	Adds a text label to the vertical axis (the ordinate).

Linear Algebraic Equations

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

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

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

```
>>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.

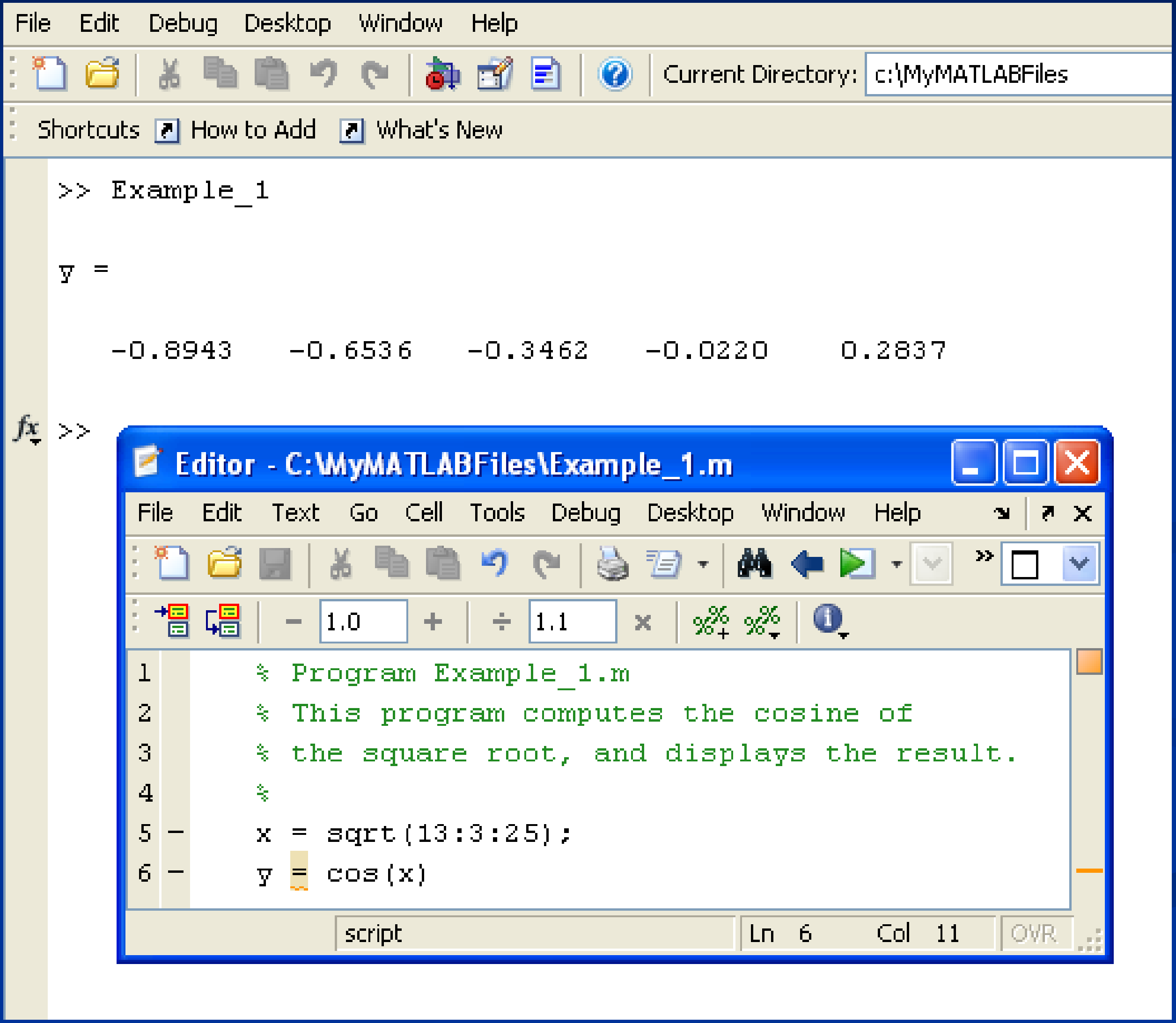
COMMENTS

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.


File Edit Debug Desktop Window Help
Current Directory: c:\MyMATLABFiles
Shortcuts How to Add What's New
Editor - C:\MyMATLABFiles\Example_1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
script
Ln 6 Col 11
OVR

```
>> Example_1

y =

   -0.8943   -0.6536   -0.3462   -0.0220    0.2837

>>
```

```
% Program Example_1.m
% This program computes the cosine of
% the square root, and displays the result.
%
x = sqrt(13:3:25);
y = cos(x)
```

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)

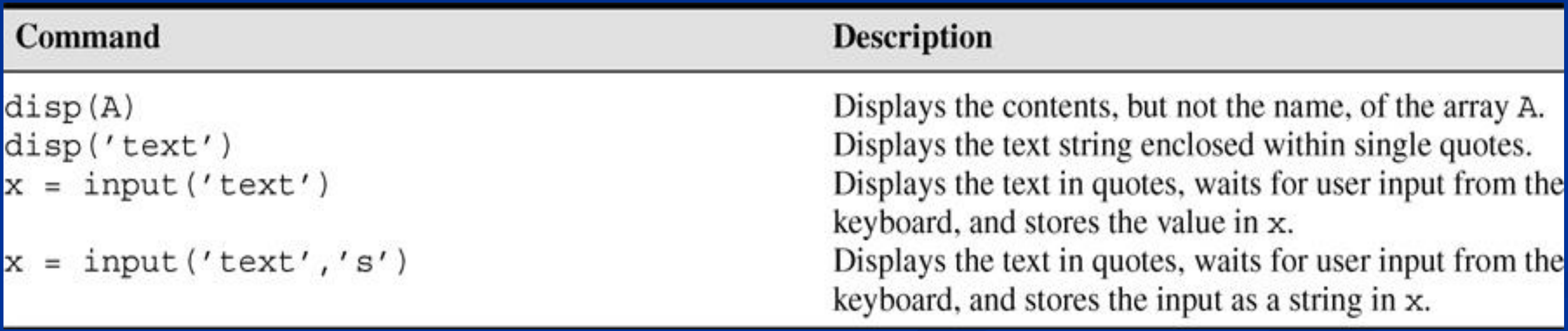
2. *Input section* Include input data and/or the input functions and comments for documentation.

3. *Calculation section*

4. *Output section* This section might contain functions for displaying the output on the screen.

Some Input/output commands

From



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

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)
%
```

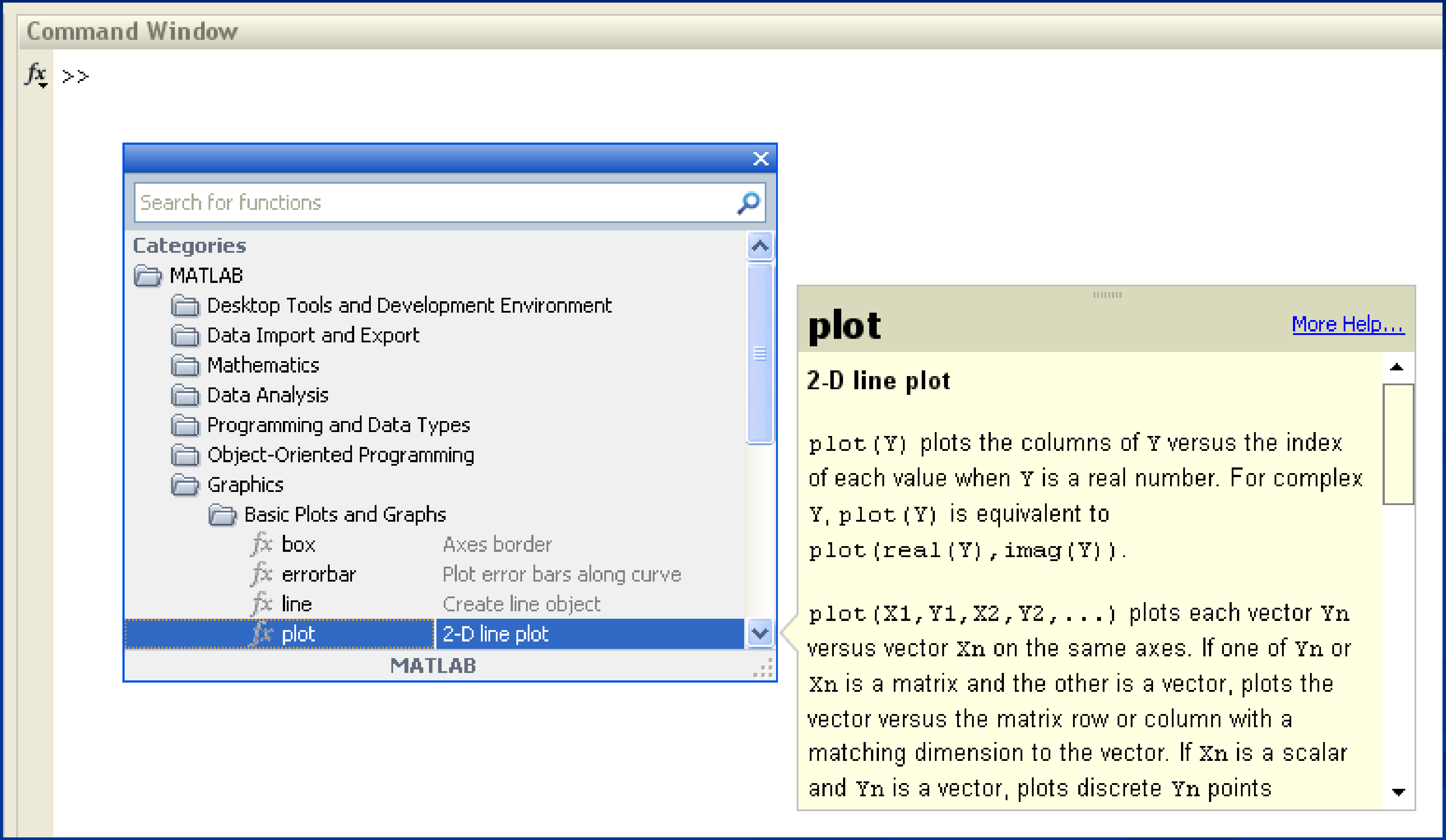
Example of a Script File (continued)

```
% 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 MATLAB: The Function Browser after `plot` has been selected



Command Window

```
>>
```

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

box	Axes border
errorbar	Plot error bars along curve
line	Create line object
plot	2-D line plot

MATLAB

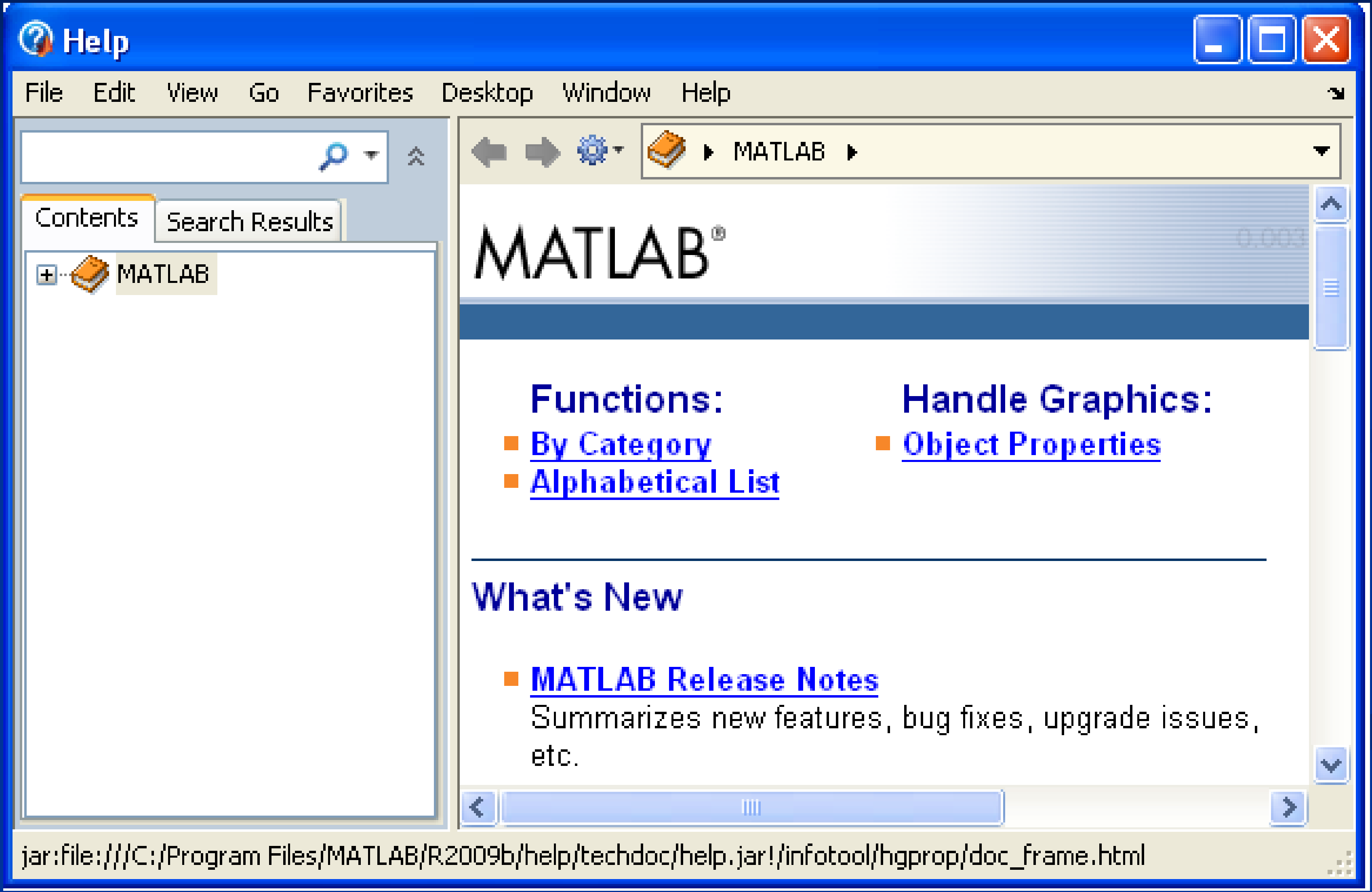
plot More Help...

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



MATLAB®

Functions:
- By Category
- Alphabetical List

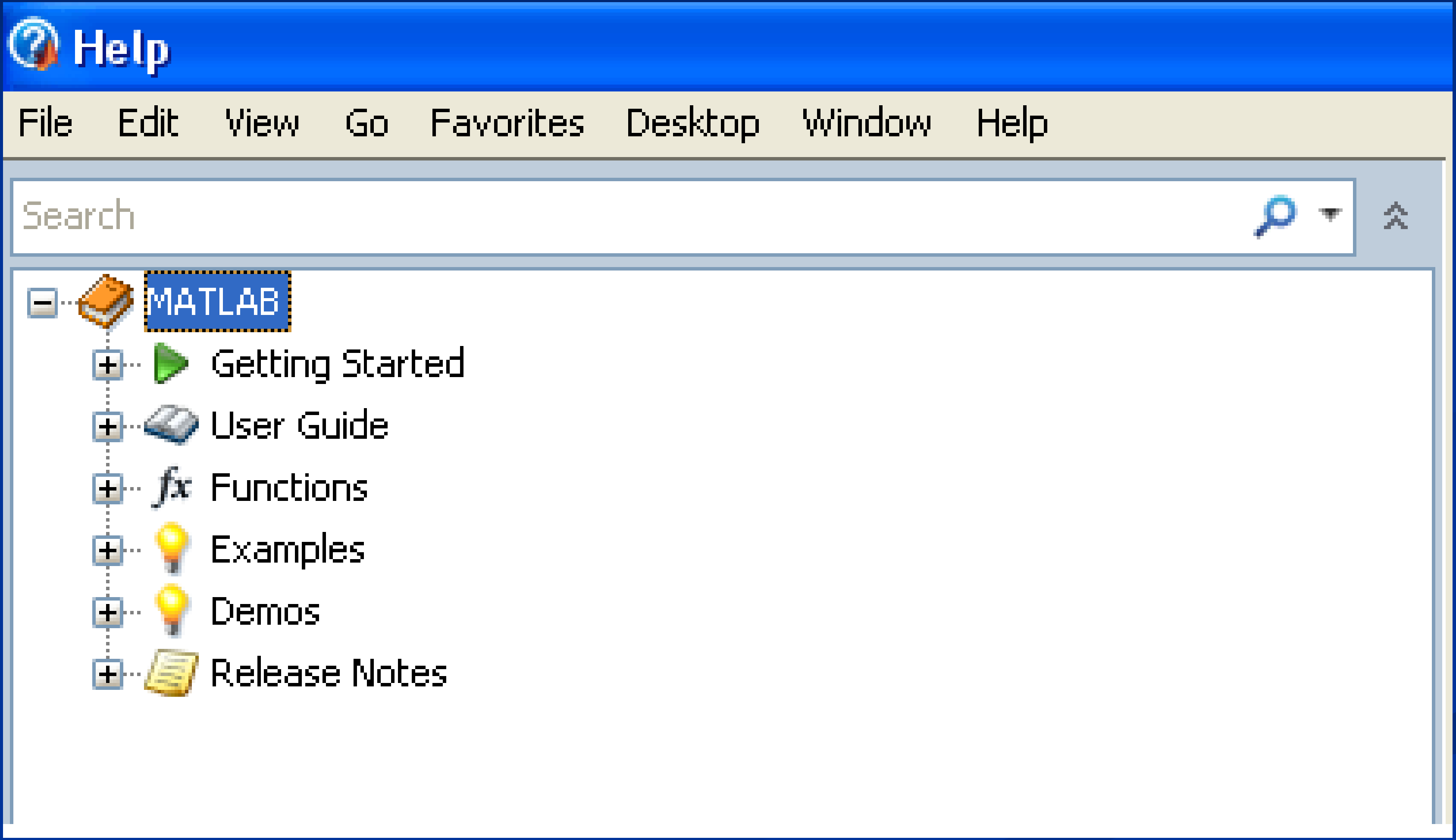
Handle Graphics:
- Object Properties

What's New

- MATLAB Release Notes
  Summarizes new features, bug fixes, upgrade issues, etc.

jar:file:///C:/Program Files/MATLAB/R2009b/help/techdoc/help.jar!/infotool/hgprop/doc_frame.html

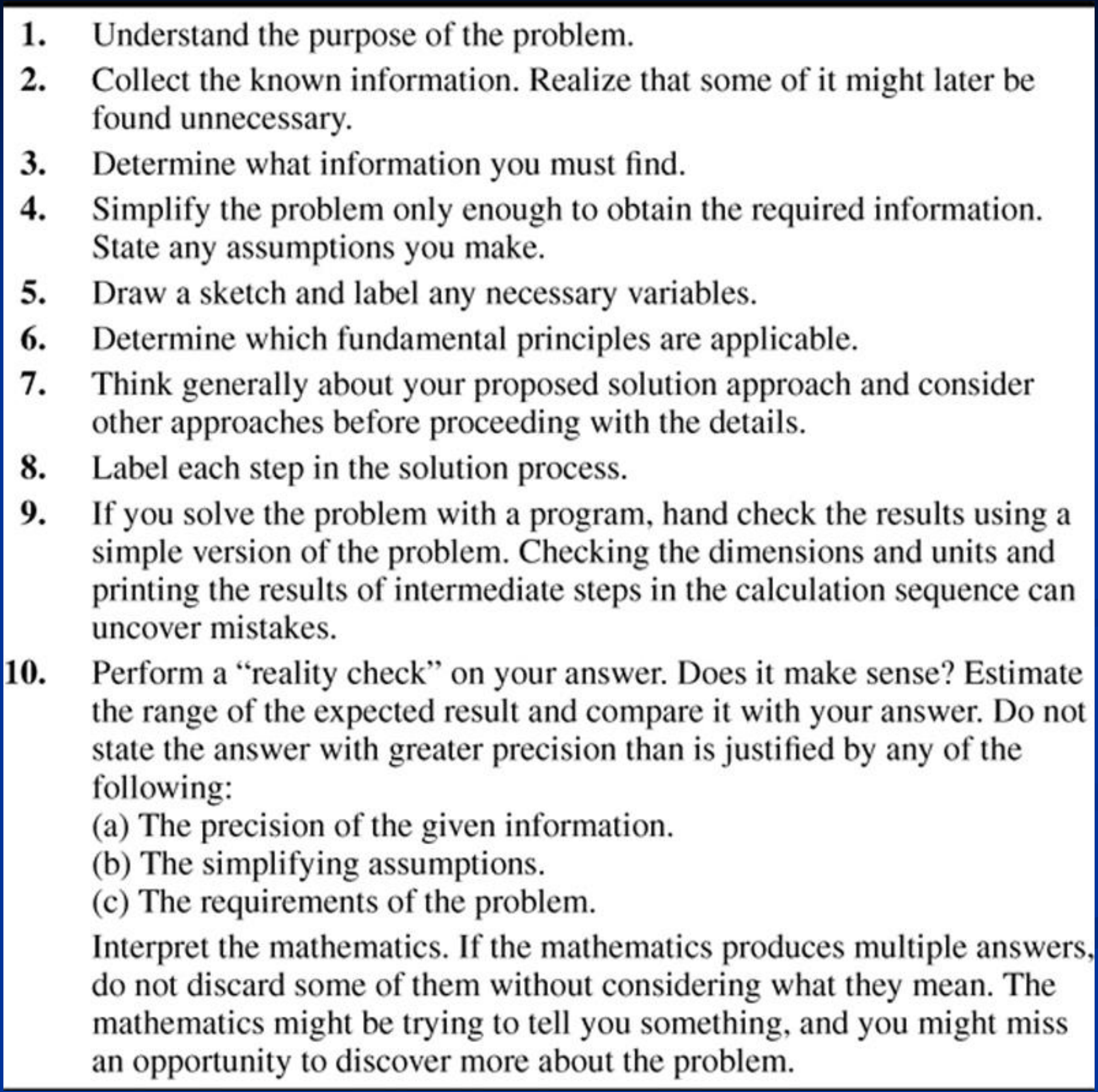
The Help Navigator



MATLAB Help Functions,

- `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.

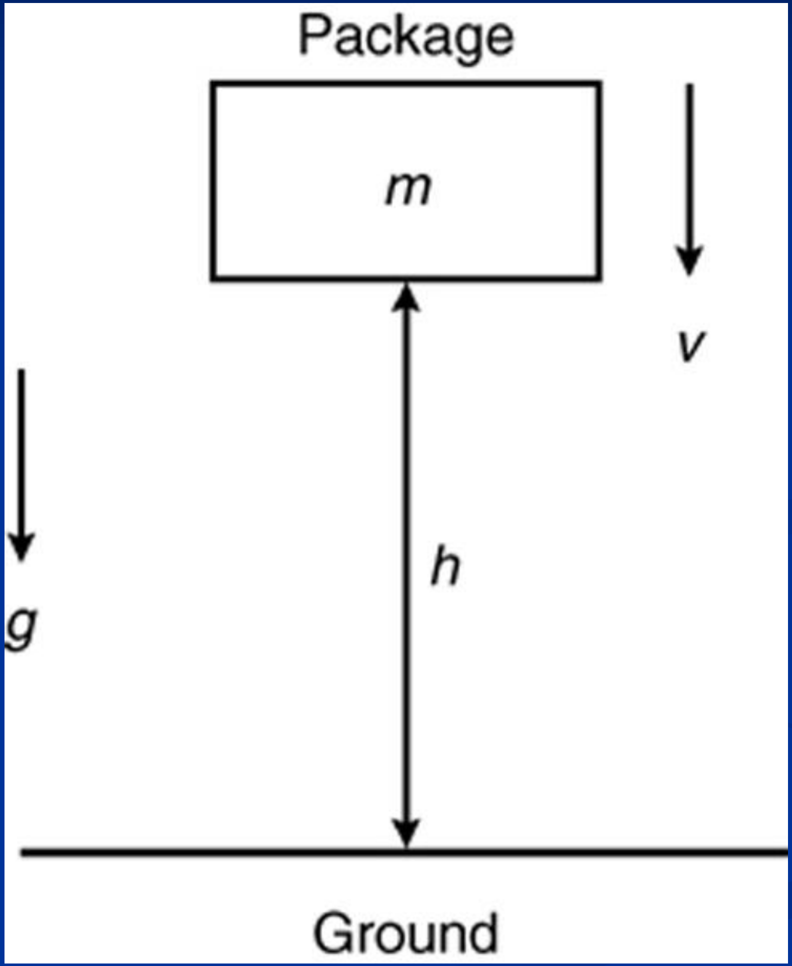
Steps in engineering problem solving



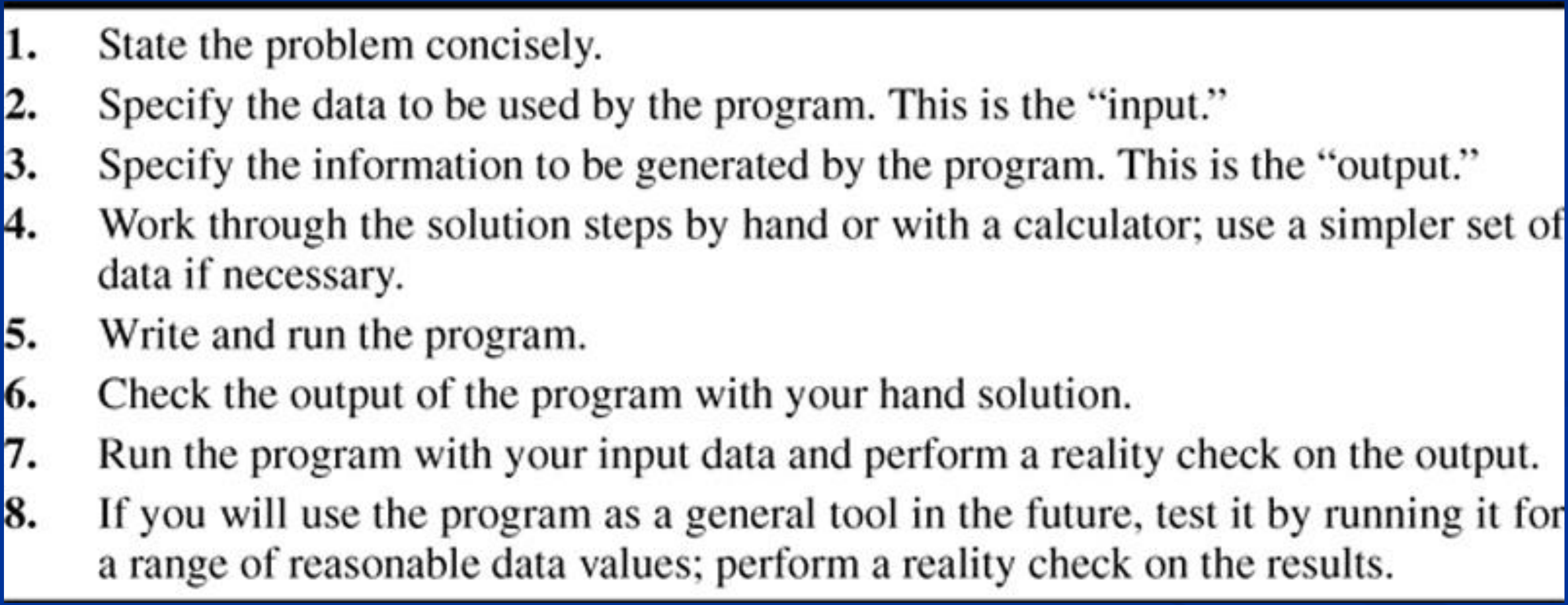
1. Understand the purpose of the problem.
2. Collect the known information. Realize that some of it might later be found unnecessary.
3. Determine what information you must find.
4. Simplify the problem only enough to obtain the required information. State any assumptions you make.
5. Draw a sketch and 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.



Steps for developing a computer solution



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.