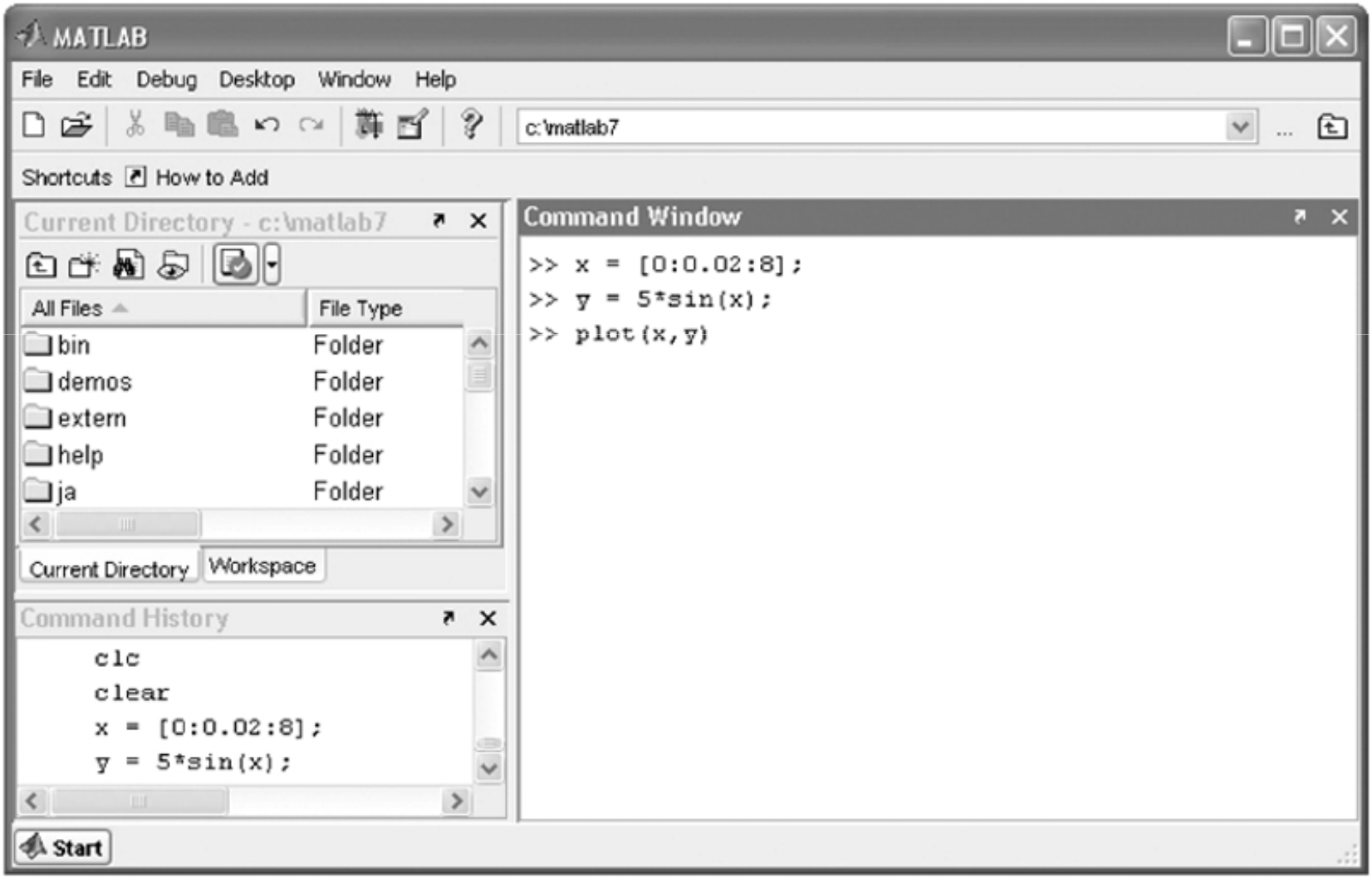
CSCI1050 – Hands-On Introduction to MATLAB

MATLAB Basic

Section 1.1 MATLAB Interactive Sessions

The default MATLAB Desktop. Figure 1.1–1



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.

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
```

Scalar arithmetic operations

Table 1.1–1

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

Order of precedence

Table 1.1–2

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.

```
>> 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
```

```
>> 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

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 1.1–4

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.

Numeric display formats

Table 1.1–5

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.

Section 1.3 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, …, 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 1.3–1

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)`

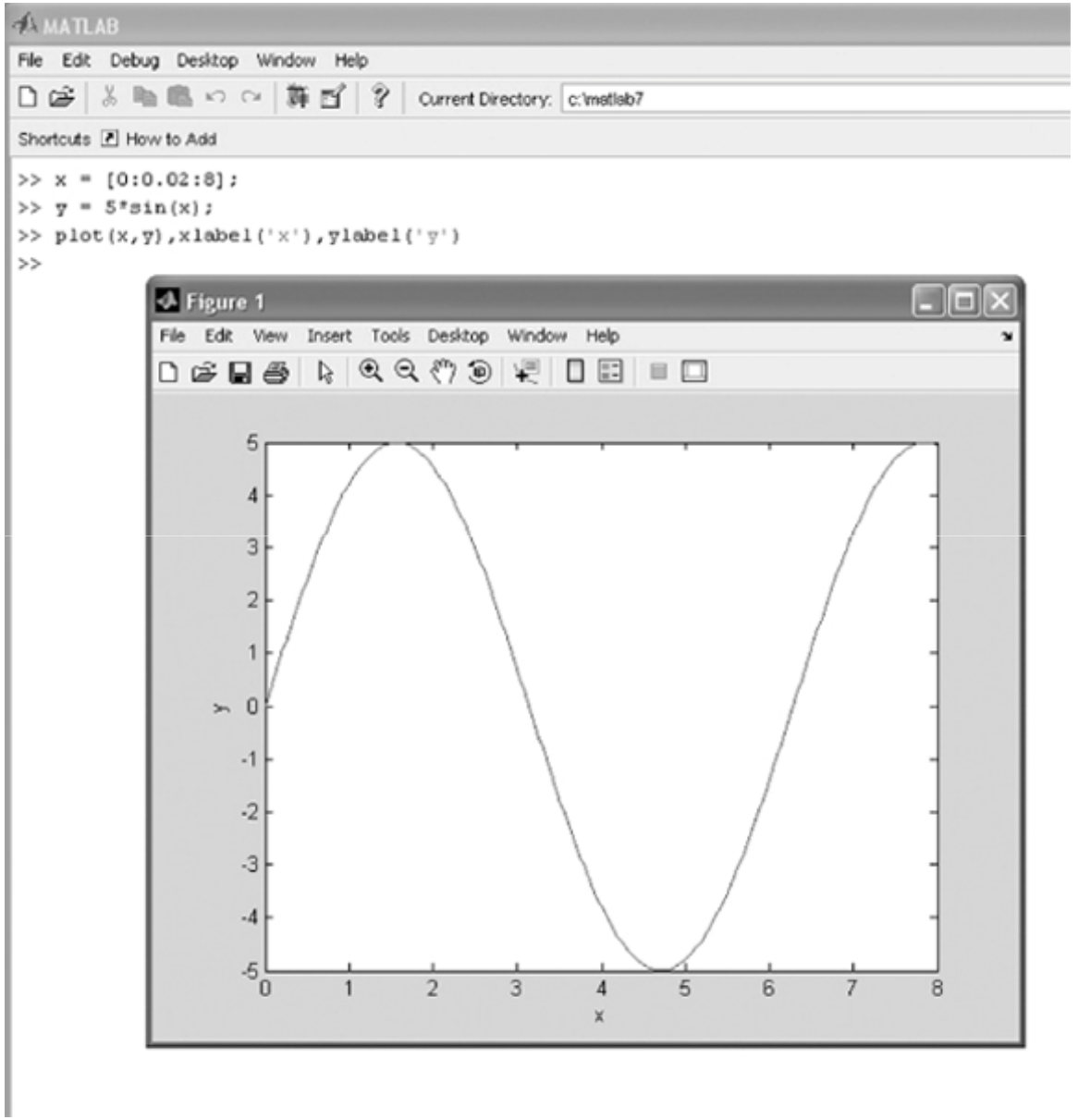
[1]The MATLAB trigonometric functions use radian measure.

When you type `problem1`,

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

A graphics window showing a plot.

Figure 1.3–1



Some MATLAB plotting commands

Table 1.3–3

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).

Section 1.4 Script Files

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.

Inserting Comments

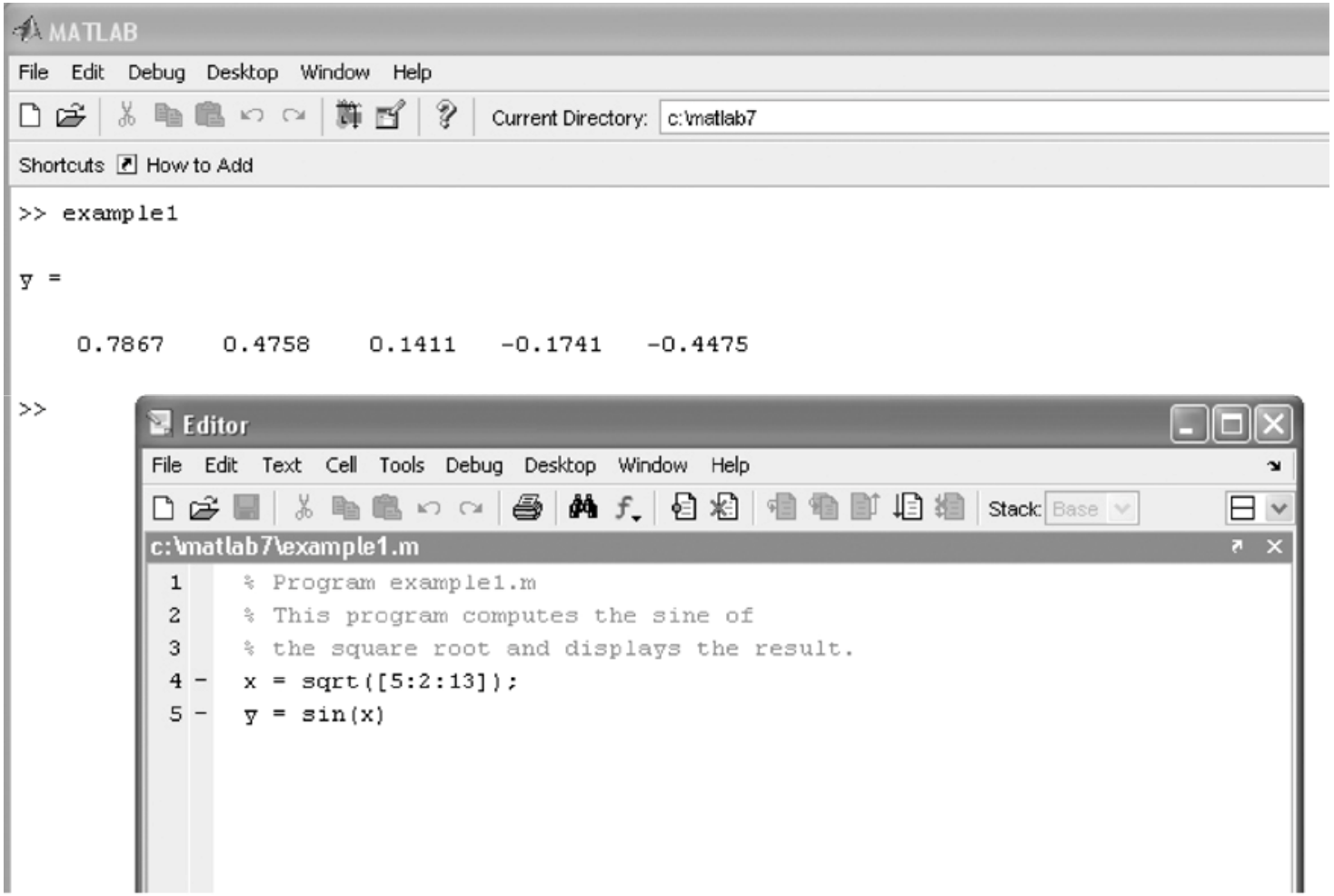
The comment symbol may be put anywhere in the line. MATLAB ignores everything to the right of the % symbol. For example,

```
>>% 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



Keep in mind when using script files:

1. The name of a script file must follow the MATLAB convention for naming variables.
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.

Input/output commands

Table 1.4–1

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 $0 < t < t_f$, where t_f 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 October 1, 2007 by W. Palm
%
% Input Variable:
% tf = 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:
tf = input('Enter final time in seconds:');
%
```

Example of a Script File (continued)

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