Chapter 4
Using Script Files and Managing Data

A script file (see Section 1.8) is a list of MATLAB commands, called a program, that is saved in a file. When the script file is executed (run), MATLAB executes the commands. Section 1.8 describes how to create, save, and run a simple script file in which the commands are executed in the order in which they are listed, and in which all the variables are defined within the script file. This chapter gives more details about how to input data to a script file, how data is stored in MATLAB, various ways to display and save data that is created in script files, and how to exchange data between MATLAB and other applications. (How to write more advanced programs in which commands are not necessarily executed in a simple order is covered in Chapter 6.)

In general, variables can be defined (created) in several ways. As shown in Chapter 2, variables can be defined implicitly by assigning values to a variable name. Variables can also be assigned values by the output of a function. In addition, variables can be defined with data that is imported from files outside MATLAB. Once defined (either in the Command Window or when a script file is executed), the variables are stored in MATLAB's Workspace.

Variables that reside in the workspace can be displayed in various ways, saved, or exported to applications outside MATLAB. Similarly, data from files outside MATLAB can be imported to the workspace and then used in MATLAB.

Section 4.1 explains how MATLAB stores data in the workspace and how the user can see the data that is stored. Section 4.2 shows how variables that are used in script files can be defined in the Command Window and/or in script files. Section 4.3 shows how to output data that is generated when script files are executed. Section 4.4 explains how the variables in the workspace can be saved and then retrieved, and Section 4.5 shows how to import and export data from and to applications outside MATLAB.

4.1 THE MATLAB WORKSPACE AND THE WORKSPACE WINDOW

The MATLAB workspace consists of the set of variables (named arrays) that are defined and stored during a MATLAB session. It includes variables that have been defined in the Command Window and variables defined when script files are executed. This means that the Command Window and script files share the same memory zone within the computer. This implies that once a variable is in the workspace, it is recognized and can be used, and it can be reassigned new values, in both the Command Window and script files. As will be explained in Chapter 7 (Section 7.3), there is another type of file in MATLAB, called a function file, where variables can also be defined. These variables, however, are normally not shared with other parts of the program since they use a separate workspace.

Recall from Chapter 1 that the `who` command displays a list of the variables currently in the workspace. The `whos` command displays a list of the variables currently in the workspace and information about their size, bytes, and class. An example is shown below.

```
>> 'Variables in memory'
ans =
Variables in memory
>> a = 7;
>> E = 3;
>> d = [5,  a+E,  4,  E^2]
d =
     5    10     4     9
>> g = [a, a^2,  13;  a*E,  1,  a^E]
g =
      7    49     13
     21     1    343
>> who
Your variables are:
E    a    ans  d    g
>> whos
  Name      Size            Bytes  Class     Attributes

  E         1x1                 8  double
  a         1x1                 8  double
  ans       1x19               38  char
  d         1x4                32  double
  g         2x3                48  double

>>
```

Typing a string.

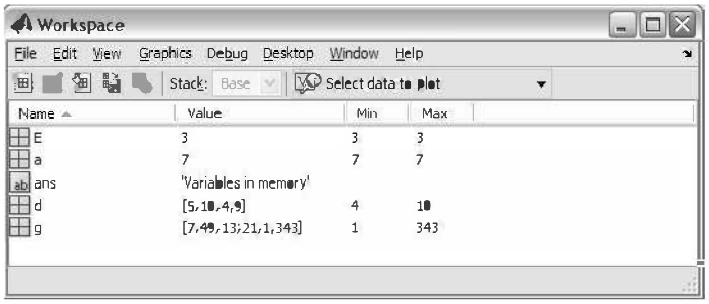
The string is assigned to `ans`.

Creating the variables `a`, `E`, `d`, and `g`.

The `who` command displays the variables currently in the workspace.

The `whos` command displays the variables currently in the workspace and information about their size and other information.

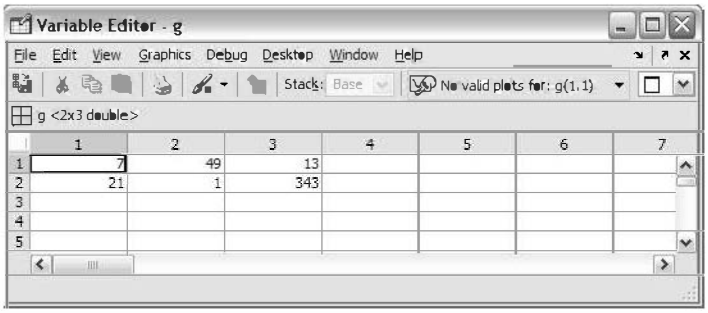
The variables currently in memory can also be viewed in the Workspace Window. This window can be opened by selecting **Workspace** in the **Desktop** menu. Figure 4-1 shows the Workspace Window that corresponds to the variables defined above. The variables that are displayed in the Workspace Window can



Name	Value	Min	Max
E	3	3	3
a	7	7	7
ans	'Variables in memory'		
d	[5,10,4,9]	4	10
g	[7,49,13;21,1,343]	1	343

Figure 4-1: The Workspace Window.

also be edited (changed). Double-clicking on a variable opens the Variable Editor Window, where the content of the variable is displayed in a table. For example, Figure 4-2 shows the Variable Editor Window that opens when the variable g in Figure 4-1 is double-clicked.


Variable Editor - g
File Edit View Graphics Debug Desktop Window Help
Stack: Base
No valid plots for: g(1,1)
g <2x3 double>

	1	2	3	4	5	6	7
1	7	49	13				
2	21	1	343				
3							
4							
5							

Figure 4-2: The Variable Editor Window.

The elements in the Variable Editor Window can be edited. The variables in the Workspace Window can be deleted by selecting them, and then either pressing the **delete** key on the keyboard or selecting **delete** from the **edit** menu. This has the same effect as entering the command `clear variable_name` in the Command Window.

4.2 INPUT TO A SCRIPT FILE

When a script file is executed, the variables that are used in the calculations within the file must have assigned values. In other words, the variables must be in the workspace. The assignment of a value to a variable can be done in three ways, depending on where and how the variable is defined.

1. The variable is defined and assigned a value in the script file.

In this case the assignment of a value to the variable is part of the script file. If the user wants to run the file with a different variable value, the file must be edited and the assignment of the variable changed. Then, after the file is saved, it can be executed again.

The following is an example of such a case. The script file (saved as Chapter4Example2) calculates the average points scored in three games.

```
% This script file calculates the average points scored in three games.
% The assignment of the values of the points is part of the script file.
game1=75;
game2=93;
game3=68;
ave_points=(game1+game2+game3)/3
```

The variables are assigned values within the script file.

The display in the Command Window when the script file is executed is:

```
>> Chapter4Example2

ave_points =
   78.6667
>>
```

The script file is executed by typing the name of the file.

The variable `ave_points` with its value is displayed in the Command Window.

2. The variable is defined and assigned a value in the Command Window.

In this case the assignment of a value to the variable is done in the Command Window. (Recall that the variable is recognized in the script file.) If the user wants to run the script file with a different value for the variable, the new value is assigned in the Command Window and the file is executed again.

For the previous example in which the script file has a program that calculates the average of points scored in three games, the script file (saved as Chapter4Example3) is:

```
% This script file calculates the average points scored in three games.
% The assignment of the values of the points to the variables
% game1, game2, and game3 is done in the Command Window.

ave_points=(game1+game2+game3)/3
```

The Command Window for running this file is:

```
>> game1 = 67;
>> game2 = 90;
>> game3 = 81;
```

The variables are assigned values in the Command Window.

```
>> Chapter4Example3
```

The script file is executed.

```
ave_points =
    79.3333
```

The output from the script file is displayed in the Command Window.

```
>> game1 = 87;
>> game2 = 70;
>> game3 = 50;
```

New values are assigned to the variables.

```
>> Chapter4Example3
```

The script file is executed again.

```
ave_points =
     69
>>
```

The output from the script file is displayed in the Command Window.

3. The variable is defined in the script file, but a specific value is entered in the Command Window when the script file is executed.

In this case the variable is defined in the script file, and when the file is executed, the user is prompted to assign a value to the variable in the Command Window. This is done by using the `input` command for creating the variable.

The form of the `input` command is:

```
variable_name = input('string with a message that
                      is displayed in the Command Window')
```

When the `input` command is executed as the script file runs, the string is displayed in the Command Window. The string is a message prompting the user to enter a value that is assigned to the variable. The user types the value and presses the **Enter** key. This assigns the value to the variable. As with any variable, the variable and its assigned value will be displayed in the Command Window unless a semicolon is typed at the very end of the `input` command. A script file that uses the `input` command to enter the points scored in each game to the program that calculates the average of the scores is shown below.

```
% This script file calculates the average of points scored in three games.
% The points from each game are assigned to the variables by
% using the input command.
game1=input('Enter the points scored in the first game ');
game2=input('Enter the points scored in the second game ');
game3=input('Enter the points scored in the third game ');
ave_points=(game1+game2+game3)/3
```

The following shows the Command Window when this script file (saved as

Chapter4Example4) is executed.

```
>> Chapter4Example4
Enter the points scored in the first game   67

Enter the points scored in the second game   91
Enter the points scored in the third game   70

ave_points =
     76
>>
```

The computer displays the message. Then the value of the score is typed by the user and the **Enter** key is pressed.

In this example scalars are assigned to the variables. In general, however, vectors and arrays can also be assigned. This is done by typing the array in the same way that it is usually assigned to a variable (left bracket, then typing row by row, and a right bracket).

The `input` command can also be used to assign a string to a variable. This can be done in one of two ways. One way is to use the command in the same form as shown above, and when the prompt message appears the string is typed between two single quotes in the same way that a string is assigned to a variable without the `input` command. The second way is to use an option in the `input` command that defines the characters that are entered as a string. The form of the command is:

```
variable_name = input('prompt message','s')
```

where the `'s'` inside the command defines the characters that will be entered as a string. In this case when the prompt message appears, the text is typed in without the single quotes, but it is assigned to the variable as a string. An example where the `input` command is used with this option is included in Sample Problem 6-4.

4.3 OUTPUT COMMANDS

As discussed before, MATLAB automatically generates a display when some commands are executed. For example, when a variable is assigned a value, or the name of a previously assigned variable is typed and the **Enter** key is pressed, MATLAB displays the variable and its value. This type of output is not displayed if a semicolon is typed at the end of the command. In addition to this automatic display, MATLAB has several commands that can be used to generate displays. The displays can be messages that provide information, numerical data, and plots. Two commands that are frequently used to generate output are `disp` and `fprintf`. The `disp` command displays the output on the screen, while the `fprintf` command can be used to display the output on the screen or to save the output to a file. The commands can be used in the Command Window, in a script file, and, as will be shown later, in a function file. When these commands are used in a script file, the display output that they generate is displayed in the Command Window.

4.3.1 *The* `disp` *Command*

The `disp` command is used to display the elements of a variable without displaying the name of the variable, and to display text. The format of the `disp` command is:

```
disp(name of a variable) or disp('text as string')
```

- Every time the `disp` command is executed, the display it generates appears in a new line. One example is:

```
>> abc = [5  9  1;  7  2  4];     A 2×3 array is assigned to variable abc.
>> disp(abc)                       The disp command is used to display the abc array.
     5     9     1
     7     2     4                 The array is displayed without its name.

>> disp('The problem has no solution.')     The disp command is used
                                             to display a message.
The problem has no solution.
>>
```

The next example shows the use of the `disp` command in the script file that calculates the average points scored in three games.

```
% This script file calculates the average points scored in three games.
% The points from each game are assigned to the variables by
% using the input command.
% The disp command is used to display the output.

game1=input('Enter the points scored in the first game   ');
game2=input('Enter the points scored in the second game  ');
game3=input('Enter the points scored in the third game   ');
ave_points=(game1+game2+game3)/3;
disp(' ')                                         Display empty line.
disp('The average of points scored in a game is:')    Display text.
disp(' ')                                         Display empty line.
disp(ave_points)     Display the value of the variable ave_points.
```

When this file (saved as Chapter4Example5) is executed, the display in the Command Window is:

```
>> Chapter4Example5
Enter the points scored in the first game    89
Enter the points scored in the second game   60
Enter the points scored in the third game    82
                                              An empty line is displayed.
The average of points scored in a game is:    The text line is displayed.
                                              An empty line is displayed.
   77                  The value of the variable ave_points is displayed.
```

- Only one variable can be displayed in a `disp` command. If elements of two variables need to be displayed together, a new variable (that contains the elements to be displayed) must first be defined and then displayed.

In many situations it is nice to display output (numbers) in a table. This can be done by first defining a variable that is an array with the numbers and then using the `disp` command to display the array. Headings to the columns can also be created with the `disp` command. Since in the `disp` command the user cannot control the format (the width of the columns and the distance between the columns) of the display of the array, the position of the headings has to be aligned with the columns by adding spaces. As an example, the script file below shows how to display the population data from Chapter 2 in a table.

```
yr=[1984 1986 1988 1990 1992 1994 1996];   The population data is
pop=[127 130 136 145 158 178 211];         entered in two row vectors.
tableYP(:,1)=yr';     yr is entered as the first column in the array tableYP.
tableYP(:,2)=pop'; pop is entered as the second column in the array tableYP.
disp('        YEAR      POPULATION')   Display heading (first line).
disp('                  (MILLIONS)')   Display heading (second line).
disp(' ')                              Display an empty line.
disp(tableYP)                          Display the array tableYP.
```

When this script file (saved as PopTable) is executed, the display in the Command Window is:

```
>> PopTable
        YEAR      POPULATION           Headings are displayed.
                  (MILLIONS)
                                       An empty line is displayed.
        1984         127
```

```
      1986        130
      1988        136
      1990        145
      1992        158
      1994        178
      1996        211
```

The `tableYP` array is displayed.

Another example of displaying a table is shown in Sample Problem 4-3. Tables can also be created and displayed with the `fprintf` command, which is explained in the next section.