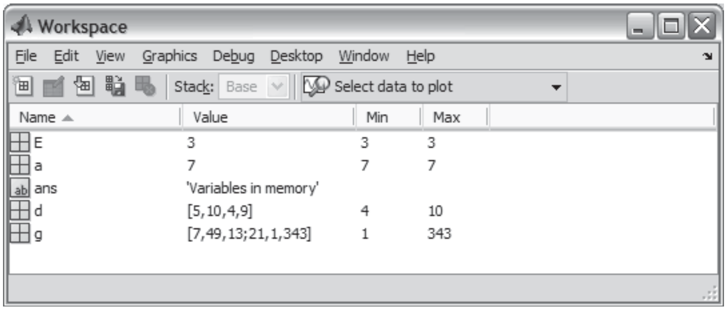
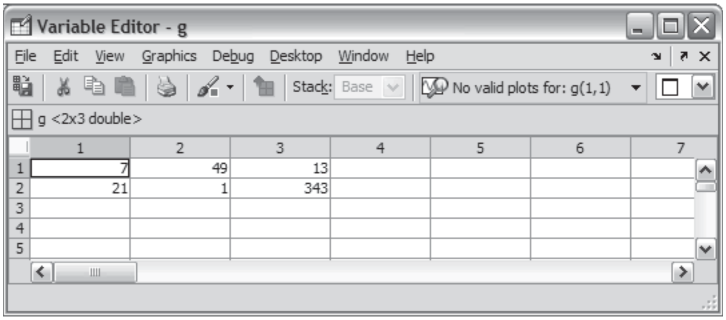
The variables currently in memory can also be viewed in the Workspace Window. This window can be opened by selecting **Workspace** in the **Desktop** menu. Figure 4-1 shows the Workspace Window that corresponds to the variables defined above. The variables that are displayed in the Workspace Window can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



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.


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.

```
% The 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:

```
% The 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.

```
>> pChater4Example3

ave_points =
   79.3333

>> game1 = 87;
>> game2 = 70;
>> game3 = 50;
>> Chapter4Example3

ave_points =
    69
>>
```

The script file is executed.

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

New values are assigned to the variables.

The script file is executed again.

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 Win-