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

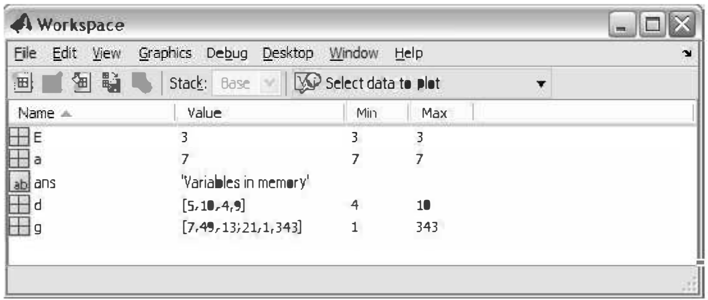


Figure 4-1: The Workspace Window.

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.

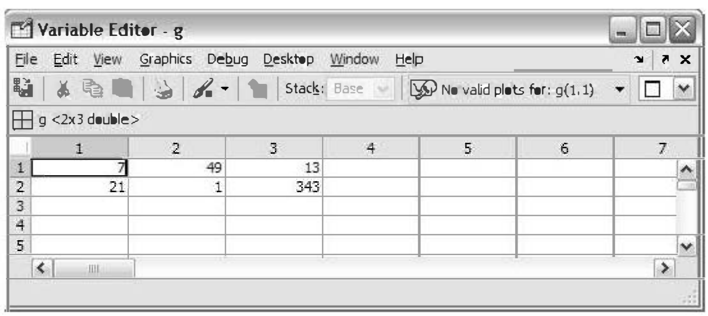


Figure 4-2: The Variable Editor Window.

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

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 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];
>> disp(abc)
     5     9     1
     7     2     4

>> disp('The problem has no solution.')

The problem has no solution.
>>
```

A 2×3 array is assigned to variable `abc`.

The `disp` command is used to display the `abc` array.

The array is displayed without its name.

The `disp` command is used to display a message.

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(' ')
disp('The average of points scored in a game is:')
disp(' ')
disp(ave_points)
```

`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          The tableYP array is displayed.
            1990          145
            1992          158
            1994          178
            1996          211
```

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.

4.3.2 The `fprintf` *Command*

The `fprintf` command can be used to display output (text and data) on the screen or to save it to a file. With this command (unlike with the `disp` command) the output can be formatted. For example, text and numerical values of variables can be intermixed and displayed in the same line. In addition, the format of the numbers can be controlled.

With many available options, the `fprintf` command can be long and complicated. To avoid confusion, the command is presented gradually. First, this section shows how to use the command to display text messages, then how to mix numerical data and text, next how to format the display of numbers, and finally how to save the output to a file.

Using the `fprintf` command to display text:

To display text, the `fprintf` command has the form:

```
fprintf('text typed in as a string')
```

For example:

```
fprintf('The problem, as entered, has no solution. Please check the
input data.')
```

If this line is part of a script file, then when the line is executed, the following is displayed in the Command Window:

```
The problem, as entered, has no solution. Please check the input data.
```

With the `fprintf` command it is possible to start a new line in the middle of the string. This is done by inserting `\n` before the character that will start the new line. For example, inserting `\n` after the first sentence in the previous example gives:

```
fprintf('The problem, as entered, has no solution.\nPlease
check the input data.')
```

When this line executes, the display in the Command Window is:

```
The problem, as entered, has no solution.
Please check the input data.
```

The `\n` is called an escape character. It is used to control the display. Other escape characters that can be inserted within the string are:

`\b` Backspace.
`\t` Horizontal tab.

When a program has more than one `fprintf` command, the display generated is continuous (the `fprintf` command does not automatically start a new line). This is true even if there are other commands between the `fprintf` commands. An example is the following script file:

```
fprintf('The problem, as entered, has no solution. Please check the
input data.')
x = 6; d = 19 + 5*x;
fprintf('Try to run the program later.')
y = d + x;
fprintf('Use different input values.')
```

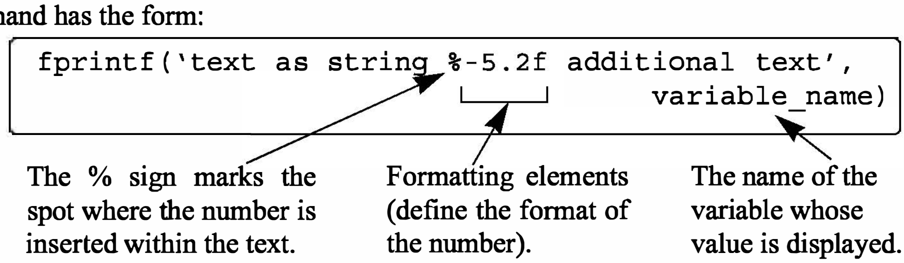
When this file is executed the display in the Command Window is:

```
The problem, as entered, has no solution. Please check the
input data.Try to run the program later.Use different input
values.
```

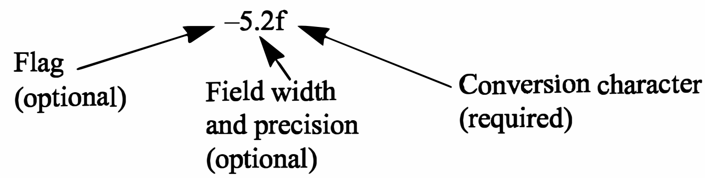
To start a new line with the `fprintf` command, `\n` must be typed at the start of the string.

Using the `fprintf` command to display a mix of text and numerical data:

To display a mix of text and a number (value of a variable), the `fprintf` command has the form:



The formatting elements are:



The flag, which is optional, can be one of the following three characters:

Character used for flag	Description
– (minus sign)	Left-justifies the number within the field.
+ (plus sign)	Prints a sign character (+ or –) in front of the number.
0 (zero)	Adds zeros if the number is shorter than the field.

The field width and precision (5.2 in the previous example) are optional. The first number (5 in the example) is the field width, which specifies the minimum number of digits in the display. If the number to be displayed is shorter than the field width, spaces or zeros are added in front of the number. The precision is the second number (2 in the example). It specifies the number of digits to be displayed to the right of the decimal point.

The last element in the formatting elements, which is required, is the conversion character, which specifies the notation in which the number is displayed. Some of the common notations are:

- e Exponential notation using lowercase e (e.g., 1.709098e+001).
- E Exponential notation using uppercase E (e.g., 1.709098E+001).
- f Fixed-point notation (e.g., 17.090980).
- g The shorter of e or f notations.
- G The shorter of E or f notations.
- i Integer.

Information about additional notation is available in the help menu of MATLAB. As an example, the `fprintf` command with a mix of text and a number is used 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 values are assigned to the variables by using the input command.
% The fprintf command is used to display the output.
game(1) = input('Enter the points scored in the first game    ');
game(2) = input('Enter the points scored in the second game   ');
game(3) = input('Enter the points scored in the third game    ');
ave_points = mean(game);
```

```
fprintf('An average of %f points was scored in the three games.',ave_points)
```

Text

% marks the position of the number.

Additional text.

The name of the variable whose value is displayed.

Notice that, besides using the `fprintf` command, this file differs from the ones shown earlier in the chapter in that the scores are stored in the first three elements of a vector named `game`, and the average of the scores is calculated by using the `mean` function. The Command Window where the script file above (saved as Chapter4Example6) was run is shown below.

```
>> Chapter4Example6
Enter the points scored in the first game    75
Enter the points scored in the second game   60
Enter the points scored in the third game    81
An average of 72.000000 points was scored in the three games.
>>
```

The display generated by the `fprintf` command combines text and a number (value of a variable).

With the `fprintf` command it is possible to insert more than one number (value of a variable) within the text. This is done by typing %g (or % followed by any formatting elements) at the places in the text where the numbers are to be inserted. Then, after the string argument of the command (following the comma), the names of the variables are typed in the order in which they are inserted in the text. In general the command looks like:

```
fprintf('..text...%g...%g...%f...',variable1,variable2,variable3)
```

An example is shown in the following script file:

```
% This program calculates the distance a projectile flies,
% given its initial velocity and the angle at which it is shot.
% the fprintf command is used to display a mix of text and numbers.

v=1584;   % Initial velocity (km/h)
theta=30;   % Angle (degrees)
vms=v*1000/3600;                    Changing velocity units to m/s.
t=vms*sind(30)/9.81;                Calculating the time to highest point.
d=vms*cosd(30)*2*t/1000;            Calculating max distance.
```

```
fprintf('A projectile shot at %3.2f degrees with a velocity
of %4.2f km/h will travel a distance of %g km.\n',theta,v,d)
```

When this script file (saved as Chapter4Example7) is executed, the display in the Command Window is:

```
>> Chapter4Example7
A projectile shot at 30.00 degrees with a velocity of
1584.00 km/h will travel a distance of 17.091 km.
>>
```

Additional remarks about the `fprintf` command:

- To place a single quotation mark in the displayed text, type two single quotation marks in the string inside the command.
- The `fprintf` command is vectorized. This means that when a variable that is a vector or a matrix is included in the command, the command repeats itself until all the elements are displayed. If the variable is a matrix, the data is used column by column.

For example, the script file below creates a 2×5 matrix `T` in which the first row contains the numbers 1 through 5, and the second row shows the corresponding square roots.

```
x=1:5;                 Create a vector x.
y=sqrt(x);             Create a vector y.
T=[x; y]               Create 2 × 5 matrix T, first row is x, second row is y.
fprintf('If the number is: %i, its square root is: %f\n',T)
                       The fprintf command displays two numbers from T in every line.
```

When this script file is executed, the display in the Command Window is:

```
T =
    1.0000    2.0000    3.0000    4.0000    5.0000     The 2 × 5 matrix T.
    1.0000    1.4142    1.7321    2.0000    2.2361
If the number is: 1, its square root is: 1.000000    The fprintf
If the number is: 2, its square root is: 1.414214    command repeats
If the number is: 3, its square root is: 1.732051    five times, using
If the number is: 4, its square root is: 2.000000    the numbers from
If the number is: 5, its square root is: 2.236068    the matrix T col-
                                                     umn after column.
```

Using the `fprintf` command to save output to a file:

In addition to displaying output in the Command Window, the `fprintf` command can be used for writing the output to a file when it is necessary to save the output. The data that is saved can subsequently be displayed or used in MATLAB and in other applications.

Writing output to a file requires three steps:

a) Opening a file using the `fopen` command.
b) Writing the output to the open file using the `fprintf` command.
c) Closing the file using the `fclose` command.

Step *a*:

Before data can be written to a file, the file must be opened. This is done with the `fopen` command, which creates a new file or opens an existing file. The `fopen` command has the form:

```
fid = fopen('file_name','permission')
```

`fid` is a variable called the file identifier. A scalar value is assigned to `fid` when `fopen` is executed. The file name is written (including its extension) within single quotes as a string. The permission is a code (also written as a string) that tells how the file is opened. Some of the more common permission codes are:

`'r'`	Open file for reading (default).
`'w'`	Open file for writing. If the file already exists, its content is deleted. If the file does not exist, a new file is created.
`'a'`	Same as `'w'`, except that if the file exists the written data is appended to the end of the file.
`'r+'`	Open (do not create) file for reading and writing.
`'w+'`	Open file for reading and writing. If the file already exists, its content is deleted. If the file does not exist, a new file is created.
`'a+'`	Same as `'w+'`, except that if the file exists the written data is appended to the end of the file.

If a permission code is not included in the command, the file opens with the default code `'r'`. Additional permission codes are described in the help menu.

Step *b*:

Once the file is open, the `fprintf` command can be used to write output to the file. The `fprintf` command is used in exactly the same way as it is used to display output in the Command Window, except that the variable `fid` is inserted inside the command. The `fprintf` command then has the form:

```
fprintf(fid,'text %-5.2f additional text',vari
                                        able_name)
```

`fid` is added to the `fprintf` command.

Step *c*:

When the writing of data to the file is complete, the file is closed using the `fclose` command. The `fclose` command has the form:

```
fclose(fid)
```

Additional notes on using the `fprintf` command for saving output to a file:

- The created file is saved in the current directory.
- It is possible to use the `fprintf` command to write to several different files. This is done by first opening the files, assigning a different `fid` to each (e.g. `fid1`, `fid2`, `fid3`, etc.), and then using the `fid` of a specific file in the `fprintf` command to write to that file.

An example of using `fprintf` commands for saving output to two files is shown in the following script file. The program in the file generates two unit conversion tables. One table converts velocity units from miles per hour to kilometers per hour, and the other table converts force units from pounds to newtons. Each conversion table is saved to a different text file (extension .txt).

```
% Script file in which fprintf is used to write output to files.
% Two conversion tables are created and saved to two different files.
% One converts mi/h to km/h, the other converts lb to N.
clear all
Vmph=10:10:100;                       Creating a vector of velocities in mi/h.
Vkmh=Vmph.*1.609;                     Converting mph to km/h.
TBL1=[Vmph; Vkmh];                    Creating a table (matrix) with two rows.
Flb=200:200:2000;                     Creating a vector of forces in lb.
FN=Flb.*4.448;                        Converting lb to N.
TBL2=[Flb; FN];                       Creating a table (matrix) with two rows.
fid1=fopen('VmphtoVkm.txt','w');      Open a .txt file named VmphtoVkm.
fid2=fopen('FlbtoFN.txt','w');        Open a .txt file named FlbtoFN.
fprintf(fid1,'Velocity Conversion Table\n \n');
                                      Writing a title and an empty line to the file fid1.
fprintf(fid1,'      mi/h          km/h    \n');
                                      Writing two column headings to the file fid1.
fprintf(fid1,'   %8.2f       %8.2f\n',TBL1);
                                      Writing the data from the variable TBL1 to the file fid1.
```

```
fprintf(fid2,'Force Conversion Table\n \n');
fprintf(fid2,'     Pounds        Newtons     \n');
fprintf(fid2,'   %8.2f       %8.2f\n',TBL2);
fclose(fid1);
fclose(fid2);
```

Writing the force conversion table (data in variable TBL2) to the file fid2.

Closing the files fid1 and fid2.

When the script file above is executed two new .txt files, named VmphtoVkm and FlbtoFN, are created and saved in the current directory. These files can be opened with any application that can read .txt files. Figures 4-3 and 4-4 show how the two files appear when they are opened with Microsoft Word.

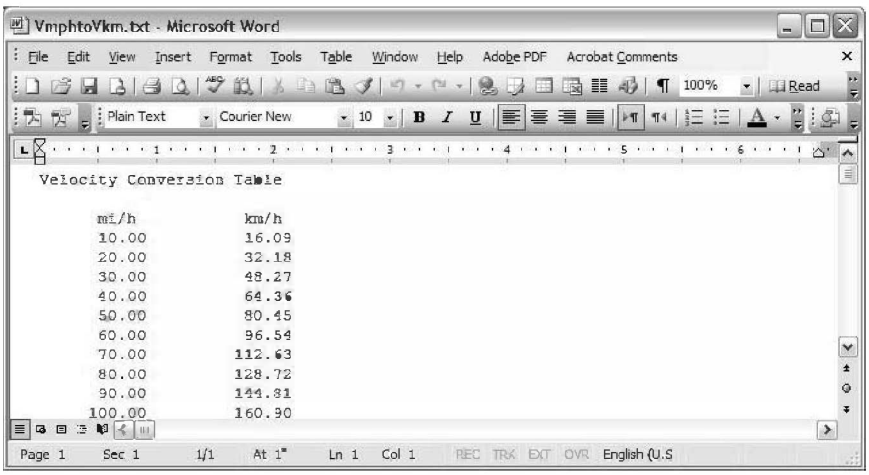


Figure 4-3: The VmphtoVkm.txt file opened in Word.

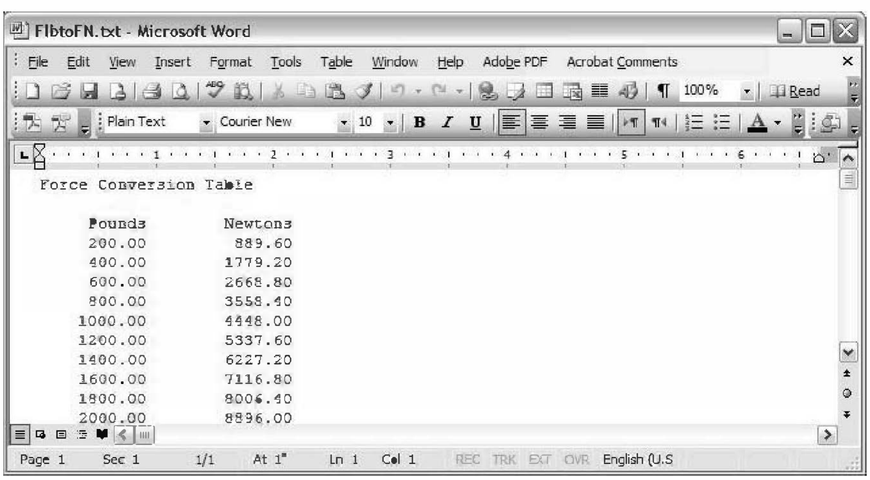


Figure 4-4: The FlbtoFN.txt file opened in Word.

4.4 THE save AND load COMMANDS

The save and load commands are most useful for saving and retrieving data for use in MATLAB. The save command can be used for saving the variables that are currently in the workspace, and the load command is used for retrieving variables that have been previously saved, to the workspace. The workspace can be saved when MATLAB is used in one type of platform (e.g., PC), and retrieved for use in MATLAB in another platform (e.g., Mac). The save and load commands can also be used for exchanging data with applications outside MATLAB. Additional commands that can be used for this purpose are presented in Section 4.5.

4.4.1 The save Command

The save command is used for saving the variables (all or some of them) that are stored in the workspace. The two simplest forms of the save command are:

```
save file_name
```

and

```
save('file_name')
```

When either one of these commands is executed, all of the variables currently in the workspace are saved in a file named file_name.mat that is created in the current directory. In mat files, which are written in a binary format, each variable preserves its name, type, size, and value. These files cannot be read by other applications. The save command can also be used for saving only some of the variables that are in the workspace. For example, to save two variables named var1 and var2, the command is:

```
save file_name var1 var2
```

or

```
save('file_name','var1','var2')
```

The save command can also be used for saving in ASCII format, which can be read by applications outside MATLAB. Saving in ASCII format is done by adding the argument -ascii in the command (for example, save file_name -ascii). In the ASCII format the variable's name, type, and size are not preserved. The data is saved as characters separated by spaces but without the variable names. For example, the following shows how two variables (a 1×4 vector and a 2×3 matrix) are defined in the Command Window and then saved in ASCII format to a file named DatSavAsci:

```
>> V=[3 16 -4 7.3];                      Create a 1×4 vector V.
>> A=[6 -2.1 15.5; -6.1 8 11];           Create a 2×3 matrix A.
>> save -ascii DatSavAsci                Save variables to a file named DatSavAsci.
```