Note that `C` is an ASCII string in a column vector. To change it to a row vector in characters we transpose it and then use `setstr` as in:

```
>> C = setstr(C')
   C =
   MATLAB
```

Finally, we close the file:

```
>>  fclose(fid_bin)
   ans = 0
```

This value indicates that the file was successfully closed.

Passing Data Between MATLAB and Excel

A software package used in engineering, science, and finance is Excel. Excel and MATLAB can read and write data to files. In this section we show how such files can be used by any of the packages. For example, MATLAB can write data separated by commas in files with extension `csv`, for comma separated values, and then read by Excel, and vice versa.

Exporting Data to Excel

To show how we can export data from MATLAB to Excel we have the following example.

Example Exporting data to Excel from MATLAB
Let us consider the following data about the five countries with the largest territories in square miles in the American continent together with their capital cities:

```
Canada, Ottawa, 3849660
United States of America, Washington D.C., 3787319
Brazil, Brasilia, 3300410
Argentina, Buenos Aires, 1073596
Mexico, Mexico D.F., 759589
```

This data is written by MATLAB to file `countries.csv`. The following file opens the file `countries.csv`, writes the data, and closes the file. Each country name must have the same number of characters. Each capital name must have ten characters, including blank spaces.

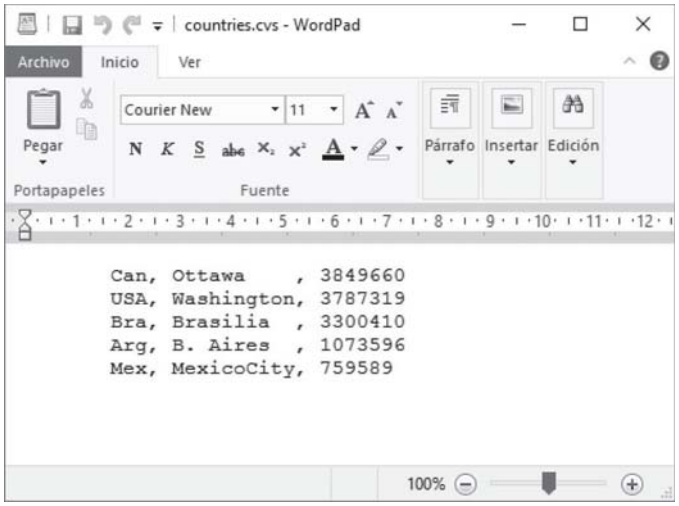

FIGURE Data in the file countries.csv.

```
% File Example .m
Country = [‘Can’; ‘USA’; ‘Bra’; ‘Arg’; ‘Mex’]
Capital=[‘Ottawa ’; ‘Washington’; ‘Brasilia ’;‘B. Aires ’;‘MexicoCity’]
Size = [3849660; 3787319; 3300410; 1073596; 759589]
handle = fopen(‘countries.csv’, ‘w’)
for i = 1: 5
   fprintf(handle, ‘%10s, %10s, %7d \n’,...
      Country(i, :), Capital(i, :), Size(i, :))
end
fclose(handle)
```

Now we open the file `countries.csv` with the Wordpad and we see the contents shown in Figure 6.3. We note that it is in comma-separated-values format. The commas are called separators or delimiters. Now we proceed to open the file with Excel. Since the file was not created by Excel, it has to be imported to Excel. The `Import Wizard` is automatically opened. It consists of three windows. The first window is shown in Figure 6.4. Here we indicate that the data is delimited by commas as indicated. After pressing the `Next` button the second window for the `Import Wizard` opens and here we indicate that the data is delimited by commas as shown in Figure 6.5. Finally, when we press the `Next` button we get to the third window in the `Import Wizard`. Here we select the columns we want to import and set the data format, as shown in Figure 6.6. Finally, the data is shown in Excel as can be seen in Figure 6.7.

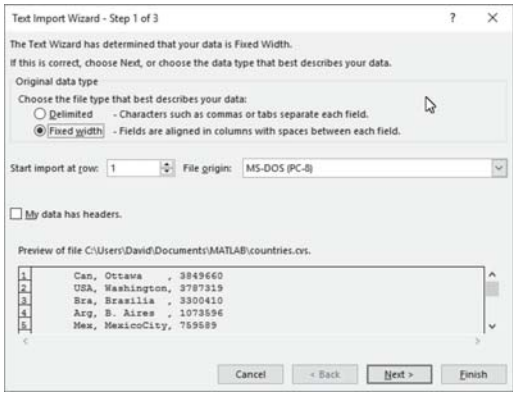

FIGURE 6.4: Part 1 of the `Import Wizard`. Here we indicate that the data is separated by commas or tabs.

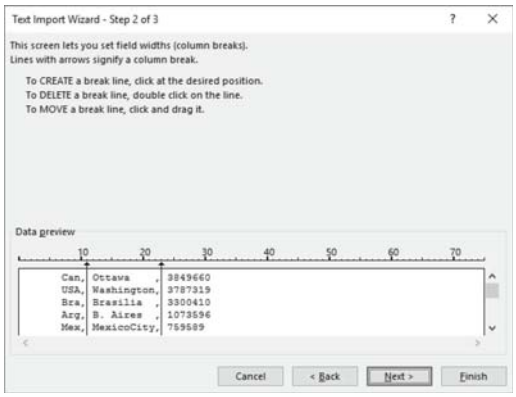

FIGURE 6.5: Part 2 of the `Import Wizard`. Here we indicate that the data is separated by commas.

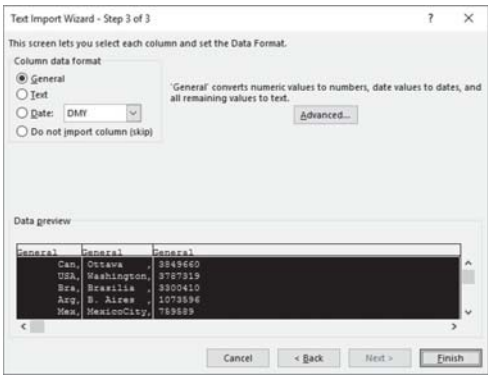

FIGURE 6.6: Part 3 of the `Import Wizard`. Here we select each column and set the data format.

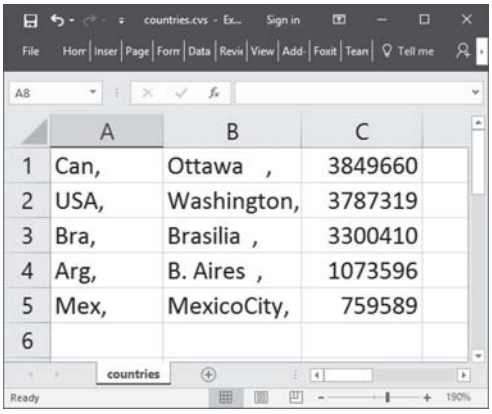

FIGURE 6.7: Data from `countries.csv` in Excel.

Another instruction used in MATLAB to write only numerical data to a file and read it by Excel is the instruction `csvwrite('file_name', m)`. For example, for the matrix `A` given by

```
>> A = [1957 10 5; 1950 10 8;1989 5 10 ]

A =
   1957  10  5
   1950  10  8
   1989  5  10
```

We can write this matrix to a file `list.csv` with

```
>> csvwrite ('list.csv', A )
```

The file `list.csv` can be readily opened with Excel.

Exporting Excel Files to MATLAB

The instruction `csv` also allows numerical data transfer from Excel to MATLAB. To show how this can be done, in Excel define the matrix `A` given by

$$A = \begin{bmatrix} 1 & 0 \\ 100 & 1 \\ 2 & 4 \end{bmatrix}$$

The matrix in Excel is shown in Figure 6.8. We save it in a file `Numbers.csv`.

Now, from the MATLAB `Command Window` we use the instruction `csvread` as `csvread('Numbers.csv')` to obtain the data in MATLAB as follows:

```
>>  csvread('Numbers.csv')
```

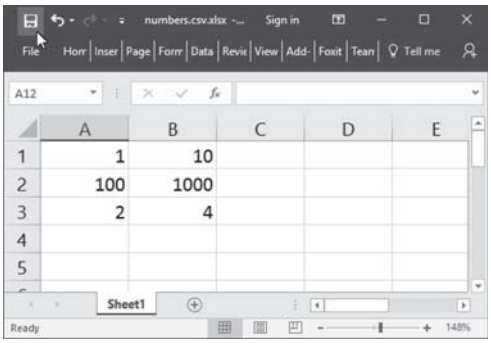

FIGURE 6.8: Data in Excel for `Numbers.csv`.

```
ans =
   1  10
  100  1000
   2  4
```

Reading Data from Excel Files

MATLAB can also read data from files with either extension `xls` or `xlsx`. To describe the procedure we use the Excel data shown in Figure 6.9. This data is saved in the file `years.xlsx` in the current directory for the MATLAB session. Now, in MATLAB we look at the `Current Directory` window and locate the file `years.xlsx`. We just double click on this file and then the `Import Wizard` opens requesting the variables to be imported. Selecting the variable to be imported (see Figure 6.10), it is displayed in the right-hand window and then we click on the `Finish` button to end the importing. In the `Workspace` window appears the variable which we can now use as any other variable created in MATLAB. This is shown in Figure 6.11.

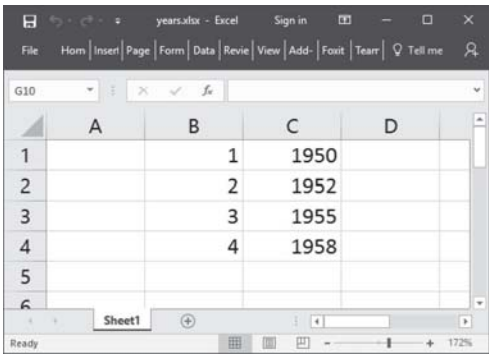

FIGURE 6.9: Data in Excel for `years.xlsx`.

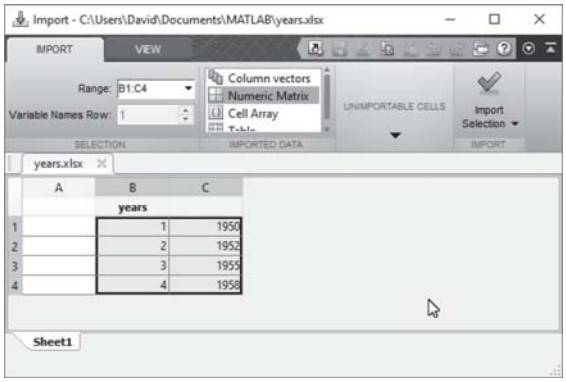

FIGURE 6.10: Import wizard for Excel files.

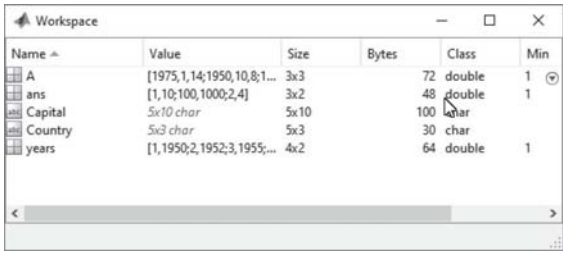

FIGURE 6.11: Import wizard for variables.

Publishing m-files from MATLAB

A very important part in programming is documentation. Sometimes this may be either a very tedious part in the programming process or a very easy one depending upon the programmer's style. Fortunately, in the case of m-language programming, MATLAB has a tool to create documentation once a program has been finished. This is known as publishing and the end result is a document that can be created in Word, HTML, XML, LaTeX, or Power Point. Furthermore, it is possible to run the program documented from the published file. We show the procedure with an example. In order to learn how an m-file has to be structured to it, first we have to describe cell programming.

Cell Programming

The MATLAB editor has the option to create cells. These cells are useful to run portions of the m-file and to create sections in the publishing process. A cell is a portion of an m-file having certain characteristics. A cell is composed of the following parts:

1. A beginning row which starts with a double percent sign followed by a space and a title text.

2. Comment lines that start with a percent sign followed by a space and text which is the body of the documentation.

3. Equations written in LaTeX style, and finally:

4. MATLAB instructions. We show the procedure with an example.

Example Plotting of a sine function with cells
Let us suppose that we want to plot the sine function from 0 to 2π and then we want to modify the plot. To plot the function we use

```
x = 0: 0.01: 2*pi;
y = sin(x);
plot(x, y)
```

Once the function is plotted we add title, legends to the axes and a label with:

```
xlabel('x-axis')
ylabel('sine wave')
title('Plot of sin x')
```

Now we create the m-file using cells. In the first cell we place the first part of the m-file and in the second cell the m-file where we add the text part. We add comments to the m-file to make it self-explanatory. In the comment lines we leave a blank space between the percent sign % and the beginning of the text. The following m-file is in cell form:

```
%% Example of m-file using cells
%% Plot of a sine wave
% Here we plot a sine wave going from 0 to 2π
% As we know we have to create a vector for x values
% and then a vector for y values. The vector y has the
% information for the sine wave values
x = 0: 0.01: 2*pi;
y = sin(x);
plot(x, y)
%% Adding text information to a plot
% We add a title with title.
% We add a text to the x axis with xlabel.
% Finally, we add a text to y axis with yaxis.
%
xlabel('x-axis')
ylabel('sine wave')
title('Plot of sin x')
```

FIGURE 6.12: m-file divided in cells.

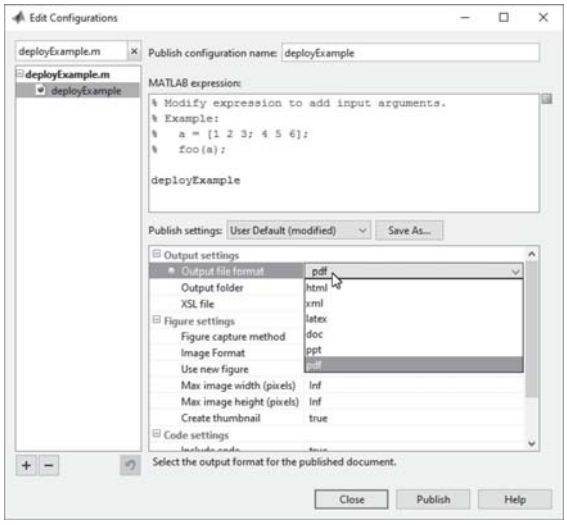

FIGURE 6.13: Publishing preferences.

The editor window is shown in Figure 6.12. We see that each cell is separated by a line and that each cell has different background color. To run the file in this mode, from the main menu we select the icon for `Run` and `Advance` and this runs the m-file a cell at a time. After running each cell we see the result of the second cell is the sine function plot, and the last cell result is the same plot with a title and with axes labels. The first cell does not display anything because there are no instructions in that cell. We now proceed to publish this m-file.

Publishing m-files

Now that we have the m-file in cell mode we can proceed to publish it. The result is a document in the format selected. The first step is to choose the `Publish` tab in the MATLAB m-file editor. In the `Publish` icon we can choose

publishing in Word, HTML, XML, LaTeX, or Power Point. The default option is the HTML format. If we wish to change to any of the other formats we can do so in the `Publish` icon which displays the preferences window for publishing as shown in Figure 6.13. For the example we choose the default pdf format. In this window we can choose the button `Publish` . We can also do it with the icon `Publish` after we close the window for the preferences. This will start the publishing process. After a few seconds we get the pdf document shown in Figure 6.14. We see in the pdf document that it has

1. A title,
2. A table of contents,
3. MATLAB instructions,
4. Text explaining the instructions, and
5. Plots.

We now describe each part of the document:

1. The title "`Example of m-file using cells`" is the first line of the first cell (the first cell has only the title of the document).

   %% `Example of m-file using cells`

2. The table of contents is formed by the first line of each cell. That is, the lines with a double percent sign,

   %% `Adding text information to a plot`

   %% `Plot of a sine wave`

3. The MATLAB instructions are, for the second cell,

   ```
   x = 0: 0.01: 2*pi;
   y = sin(x);
   plot(x, y)
   ```

   And for the third cell

   ```
   xlabel('x-axis')
   ylabel('sine wave')
   title('Plot of sin x')
   ```

4. The text for each cell is the commented lines. For the second cell:

   % `As we know we have to create a vector for x values`

   % `and then a vector for y values.`

TABLE 6.6: Commands for MATLAB LaTeX

Traditional Equation	MATLAB LaTeX $$equation$$
a/b	\frac{ a }{ b }
a^2	a ∧ 2
$a_{k,n}$	a_{ k, n }
α^2	\alpha ∧ 2
$\sqrt{a+b}$	\sqrt{ a + b }
$\int (a+b)dt$	\int { (a + b)dt }
$a \leq b$	a \leq b
$a \geq b$	a\geq b
\	\textbackslash

% The vector y has the information

% for the sine wave values.

For the third cell:

% We add a title with the instruction title.

% We add a text to the x axis with xlabel.

% We add a text to y axis with yaxis.

5. Finally, the plots are also displayed in the pdf document.

In the published document we may also include equations. They have to be written in the LaTeX format. Table 6.6 lists some of the more used MATLAB LaTeX formats to write equations. For example, to write

$$\int \sqrt{\alpha \sin(t)}\, dt$$

We use:

% $$ \int \sqrt {\alpha sin(t)}\, dt $$

Some rules have to be followed. These rules are:

1. If a row has an equation, there must be an empty comment row above and below.
2. There must be at least a blank space between the percent sign and the double dollar sign.

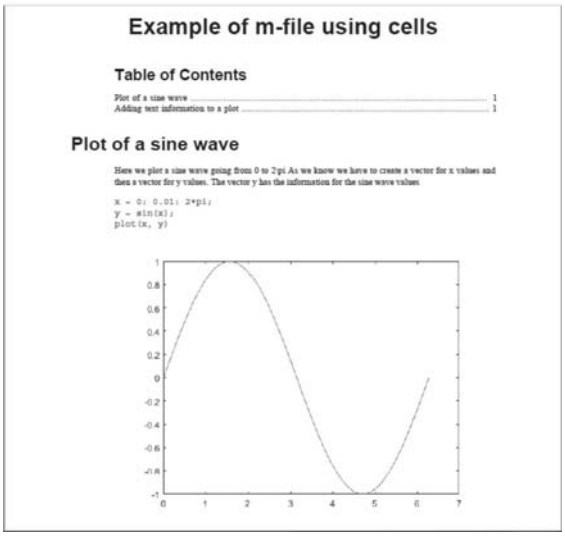

FIGURE 6.14: Top part of the deployed pdf document.

3. After a row with executable MATLAB instructions, the row has to start with a double percent sign, that is, a new cell has to start.

We now show an example to solve a quadratic equation.

Example Publishing an m-file with equations
To solve the quadratic equation

$$ax^2 + bx + c = 0$$

We have the solutions:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

To publish these equations we have to write the m-file as

```
%% Solution of a second order equation
%% Introduction
% A second order equation of the form
%
% $$ ax^2  + bx + c = 0 $$
%
% has the solutions
%
% $$ x_{1} = \frac { -b - \sqrt{b^2 - 4ac}}{2a}$$
%
% $$ x_{2} = \frac { -b + \sqrt{b^2 - 4ac}}{2a}$$
%
%% Example
```

```
% As an example we solve the equation
%
% $$ 3x^2 + 6x - 9 = 0 $$
%
% The data are then
a = 3; b = 6; c = -9;
%% Solutions
% The solutions are
%
x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
%% Results
% Finally, we display the values of the roots. %
fprintf ('x1 is a root of the second order equation %g\n', x1)
fprintf ('x2 is a root of the second order equation %g\n', x2)
```

We publish to HTML and we get the document shown in Figure 6.15.

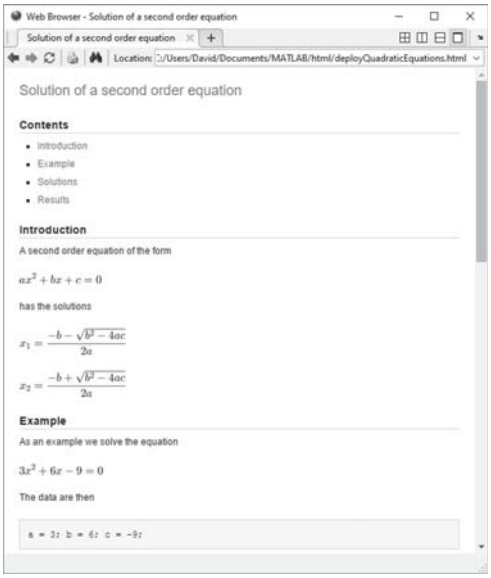

FIGURE 6.15: Top part of the published file in HTML format.

Concluding Remarks

MATLAB has integrated a powerful programming language called m-language. This language allows users to produce complex programs in a very short time when we compare it with other programming languages such as C, C++, Visual Basic, FORTRAN, among others. In the chapter we treated in detail several of the instructions needed to write a program in the m-language. The process was carried out through examples going from very simple to more

complex examples containing programs. A treatment in input/output instructions was given, in particular, the case of reading to/from a file. Also the case of transferring information between MATLAB and Excel was seen. More advanced topics such as deployment of m-files to users not having a MATLAB license was also discussed and examples provided a good understanding of the topic. Finally, since program documenting, also known as publishing, is an important part of programming, MATLAB does also provide a tool for documenting m-files. The process can be used for publishing to other different formats, but only the process for an HTML document was used.