

# 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'
```

Typing a string.

```
ans =
Variables in memory
```

The string is assigned to ans.

```
>> a = 7;
>> E = 3;
>> d = [5, a+E, 4, E^2]
```

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

```
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
```

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

```
>> 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 |            |

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

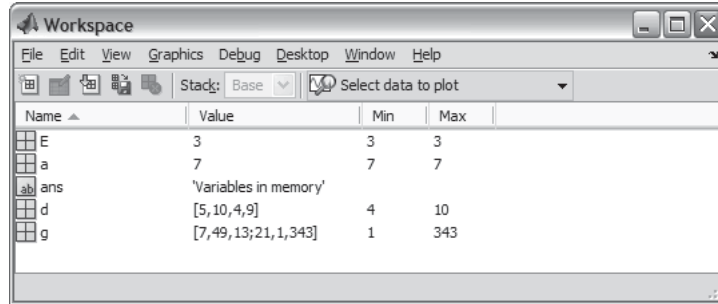


Figure 4-1: The Workspace Window.

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.

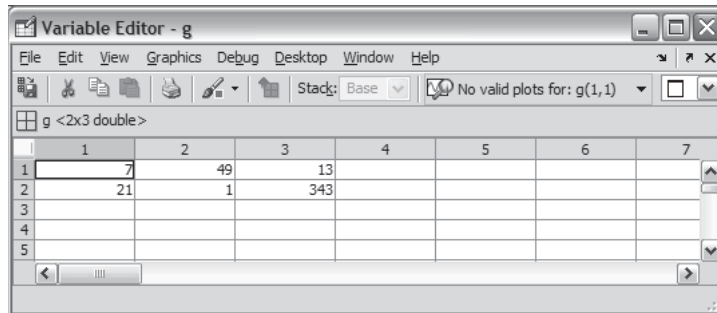


Figure 4-2: The Variable Editor Window.

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

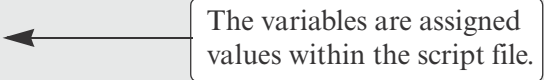
When a script file is executed, the variables that are used in the calculations within the file must have assigned values. In other words, the variables must be in the workspace. The assignment of a value to a variable can be done in three ways, depending on where and how the variable is defined.

### 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
```



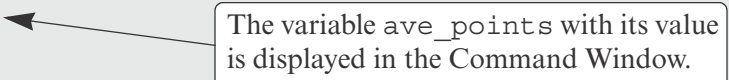
The variables are assigned values within the script file.

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-

dow, 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 problem has no solution.
>>
```

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)
```

Display empty line.

Display text.

Display empty line.

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
The average of points scored in a game is:
77
```

An empty line is displayed.

The text line is displayed.

An empty line is displayed.

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];
pop=[127 130 136 145 158 178 211];
tableYP(:,1)=yr;
tableYP(:,2)=pop;
disp('      YEAR      POPULATION')
disp('      (MILLIONS) ')
disp(' ')
disp(tableYP)
```

The population data is entered in two row vectors.

yr is entered as the first column in the array tableYP.

pop is entered as the second column in the array tableYP.

Display heading (first line).

Display heading (second line).

Display an empty line.

Display the array tableYP.

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

```
>> PopTable
      YEAR      POPULATION
      (MILLIONS)
1984      127
```

Headings are displayed.

An empty line is displayed.



|      |     |                                 |
|------|-----|---------------------------------|
| 1986 | 130 | The tableYP array is displayed. |
| 1988 | 136 |                                 |
| 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:

|                 |                 |
|-----------------|-----------------|
| <code>\b</code> | Backspace.      |
| <code>\t</code> | 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:

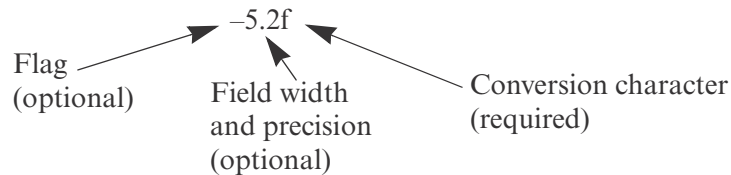
```
fprintf('text as string %-5.2f additional text',  
                                variable_name)
```

The `%` sign marks the spot where the number is inserted within the text.

Formatting elements (define the format of the number).

The name of the variable whose value is displayed.

The formatting elements are:



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

| <u>Character used<br/>for flag</u> | <u>Description</u>                                       |
|------------------------------------|--|
| – (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;
```

```
t=vms*sind(30)/9.81;
```

```
d=vms*cosd(30)*2*t/1000;
```

Changing velocity units to m/s.

Calculating the time to highest point.

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.
- To display the % character in the text, type %%.
- 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  $2 \times 5$  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;
y=sqrt(x);
T=[x; y]
fprintf('If the number is: %i, its square root is: %f\n',T)
```

Create a vector x.

Create a vector y.

Create  $2 \times 5$  matrix T, first row is x, second row is y.

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
    1.0000    1.4142    1.7321    2.0000    2.2361
If the number is: 1, its square root is: 1.000000
If the number is: 2, its square root is: 1.414214
If the number is: 3, its square root is: 1.732051
If the number is: 4, its square root is: 2.000000
If the number is: 5, its square root is: 2.236068
```

The  $2 \times 5$  matrix T.

The fprintf command repeats five times, using the numbers from the matrix T column 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:


|                   |   |
|-------------------|---|
| <code>'r'</code>  | Open file for reading (default).  |
| <code>'w'</code>  | Open file for writing. If the file already exists, its content is deleted. If the file does not exist, a new file is created.             |
| <code>'a'</code>  | Same as <code>'w'</code> , except that if the file exists the written data is appended to the end of the file.                            |
| <code>'r+'</code> | Open (do not create) file for reading and writing.  |
| <code>'w+'</code> | 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. |
| <code>'a+'</code> | Same as <code>'w+'</code> , 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', variable_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;
Vknh=Vmph.*1.609;
TBL1=[Vmph; Vknh];
Flb=200:200:2000;
FN=Flb.*4.448;
TBL2=[Flb; FN];
fid1=fopen('VmphtoVkm.txt','w');
fid2=fopen('FlbtoFN.txt','w');
fprintf(fid1,'Velocity Conversion Table\n \n');
fprintf(fid1,'      mi/h      km/h      \n');
fprintf(fid1,'      %8.2f      %8.2f\n',TBL1);
```

Creating a vector of velocities in mi/h.

Converting mph to km/h.

Creating a table (matrix) with two rows.

Creating a vector of forces in lb.

Converting lb to N.

Creating a table (matrix) with two rows.

Open a .txt file named VmphtoVkm.

Open a .txt file named FlbtoFN.

Writing a title and an empty line to the file fid1.

Writing two column headings to the file fid1.

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.

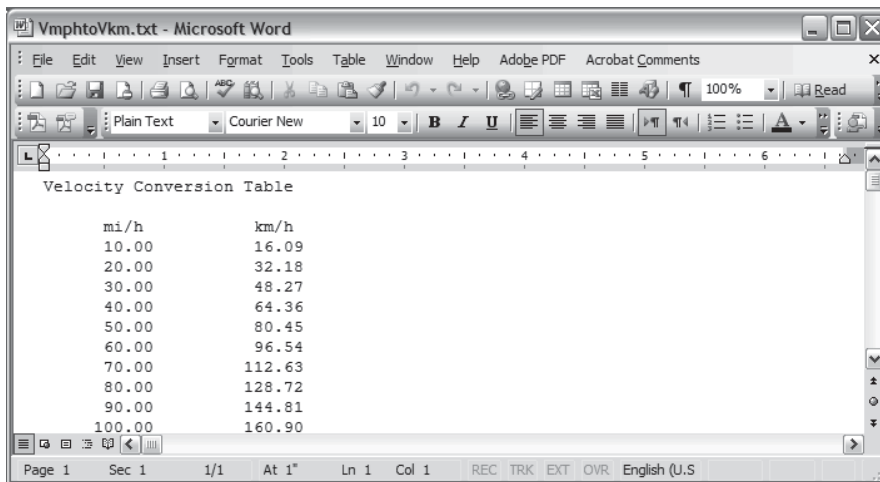


Figure 4-3: The VmphtoVkm.txt file opened in Word.

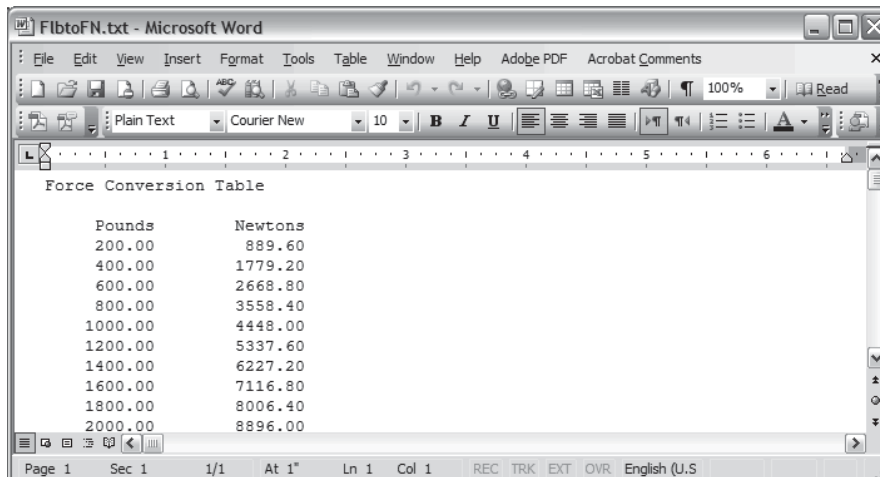


Figure 4-4: The FlbtoFN.txt file opened in Word.



#### 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  $1 \times 4$  vector and a  $2 \times 3$  matrix) are defined in the Command Window and then saved in ASCII format to a file named `DatSavAscii`:

```
>> V=[3 16 -4 7.3];
```

```
Create a 1×4 vector V.
```

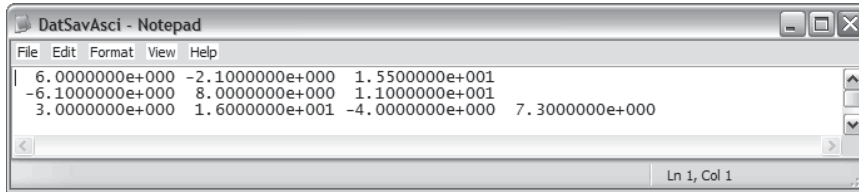
```
>> A=[6 -2.1 15.5; -6.1 8 11];
```

```
Create a 2×3 matrix A.
```

```
>> save -ascii DatSavAscii
```

```
Save variables to a file named DatSavAscii.
```

Once saved, the file can be opened by any application that can read ASCII files. For example, Figure 4-5 shows the data when the file is opened with Notepad.



**Figure 4-5: Data saved in ASCII format.**

Note that the file does not include the names of the variables, just the numerical values of the variables (first A and then V) are listed.

#### 4.4.2 The *load* Command

The `load` command can be used for retrieving variables that were saved with the `save` command back to the workspace, and for importing data that was created with other applications and saved in ASCII format or in text (.txt) files. Variables that were saved with the `save` command in .mat files can be retrieved with the command:

```
load file_name
```

or

```
load('file_name')
```

When the command is executed, all the variables in the file (with the name, type, size, and values as were saved) are added (loaded back) to the workspace. If the workspace already has a variable with the same name as a variable that is retrieved with the `load` command, then the variable that is retrieved replaces the existing variable. The `load` command can also be used for retrieving only some of the variables that are in the saved .mat file. For example, to retrieve two variables named `var1` and `var2`, the command is:

```
load file_name var1 var2
```

or

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

The `load` command can also be used to import data that is saved in ASCII or text (.txt) to the workspace. This is possible, however, only if the data in the file is in the form of a variable in MATLAB. Thus, the file can have one number (scalar), a row or a column of numbers (vector), or rows with the same number of numbers in each (matrix). For example, the data shown in Figure 4-5 cannot be loaded with the `load` command (even though it was saved in ASCII format with the `save` command), because the number of elements is not the same in all rows. (Recall that this file was created by saving two different variables.)

When data is loaded from an ASCII or text file into the workspace, it has to be assigned to a variable name. Data in ASCII format can be loaded with either of the following two forms of the load command:

```
load file_name
```

or

```
VarName=load('file_name')
```

If the data is in a text file, the extension .txt has to be added to the file name. The form of the load command is then:

```
load file_name.txt
```

or

```
VarName=load('file_name.txt')
```

In the first form of the command the data is assigned to a variable that has the name of the file. In the second form the data is assigned to a variable named VarName.

For example, the data shown in Figure 4-6 (a  $3 \times 2$  matrix) is typed in Notepad, and then saved as DataFromText.txt.

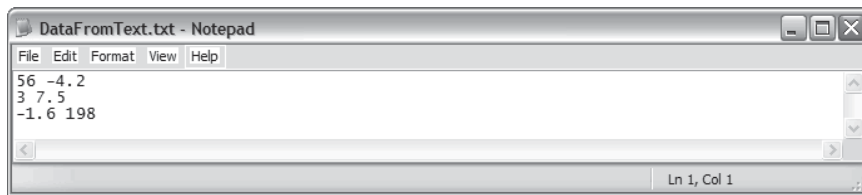


Figure 4-6: Data saved as .txt file.

Next, two forms of the load command are used to import the data in the text file to the Workspace of MATLAB. In the first command the data is assigned to a variable named DfT. In the second command the data is automatically assigned to a variable named DataFromText, which is the name of the text file where the data was saved.

```
>> DfT=load('DataFromText.txt')
```

```
DfT =  
56.0000    -4.2000  
3.0000     7.5000  
-1.6000   198.0000
```

```
>> load DataFromText.txt
```

```
>> DataFromText  
DataFromText =  
56.0000    -4.2000  
3.0000     7.5000  
-1.6000   198.0000
```

Load the file DataFromText and assign the loaded data to the variable DfT.

Use the load command with the file DataFromText.

The data is assigned to a variable named DataFromText.

Importing data to (or exporting from) other applications can also be done, with MATLAB commands that are presented in the next section.

## 4.5 IMPORTING AND EXPORTING DATA

MATLAB is often used for analyzing data that was recorded in experiments or generated by other computer programs. This can be done by first importing the data into MATLAB. Similarly, data that is produced by MATLAB sometimes needs to be transferred to other computer applications. There are various types of data (numerical, text, audio, graphics, and images). This section describes only how to import and export numerical data, which is probably the most common type of data that needs to be transferred by new users of MATLAB. For other types of data transfer, look in the Help Window under File I/O.

Importing data can be done either by using commands or by using the Import Wizard. Commands are useful when the format of the data being imported is known. MATLAB has several commands that can be used for importing various types of data. Importing commands can also be included in a script file such that the data is imported when the script is executed. The Import Wizard is useful when the format of the data (or the command that is applicable for importing the data) is not known. The Import Wizard determines the format of the data and automatically imports it.

### 4.5.1 Commands for Importing and Exporting Data

This section describes—in detail—how to transfer data into and out of Excel spreadsheets. Microsoft Excel is commonly used for storing data, and Excel is compatible with many data recording devices and computer applications. Many people are also capable of importing and exporting various data formats into and from Excel. MATLAB also has commands for transferring data directly to and from formats such as csv and ASCII, as well as to the spreadsheet program Lotus 123. Details of these and many other commands can be found in the Help Window under File I/O

#### **Importing and exporting data into and from Excel:**

Importing data from Excel is done with the `xlsread` command. When the command is executed, the data from the spreadsheet is assigned as an array to a variable. The simplest form of the `xlsread` command is:

```
variable_name = xlsread('filename')
```

- 'filename' (typed as a string) is the name of the Excel file. The directory of the Excel file must be either the current directory or listed in the search path.
- If the Excel file has more than one sheet, the data will be imported from the first sheet.

When an Excel file has several sheets, the `xlsread` command can be used to import data from a specified sheet. The form of the command is then:

```
variable_name = xlsread('filename', 'sheet_name')
```

- The name of the sheet is typed as a string.

Another option is to import only a portion of the data that is in the spreadsheet. This is done by typing an additional argument in the command:

```
variable_name = xlsread('filename', 'sheet_name', 'range')
```

- The `'range'` (typed as a string) is a rectangular region of the spreadsheet defined by the addresses (in Excel notation) of the cells at opposite corners of the region. For example, `'C2:E5'` is a  $4 \times 3$  region of rows 2, 3, 4, and 5 and columns *C*, *D*, and *E*.

Exporting data from MATLAB to an Excel spreadsheet is done by using the `xlswrite` command. The simplest form of the command is:

```
xlswrite('filename', variable_name)
```

- `'filename'` (typed as a string) is the name of the Excel file to which the data is exported. The file must be in the current directory. If the file does not exist, a new Excel file with the specified name will be created.
- `variable_name` is the name of the variable in MATLAB with the assigned data that is being exported.
- The arguments `'sheet_name'` and `'range'` can be added to the `xlswrite` command to export to a specified sheet and to a specified range of cells, respectively.

As an example, the data from the Excel spreadsheet shown in Figure 4-7 is imported into MATLAB by using the `xlsread` command.

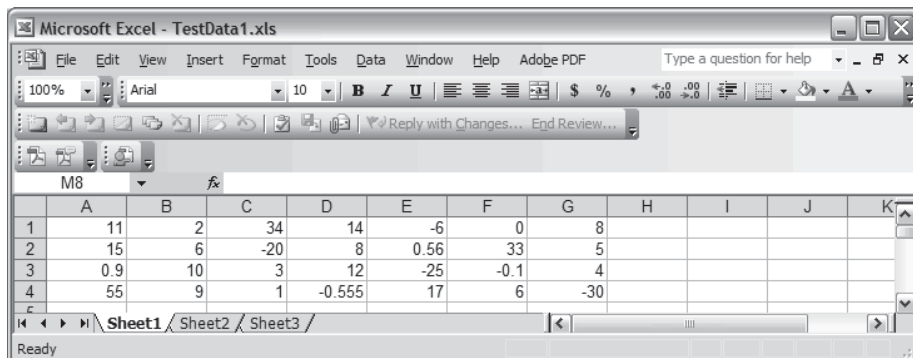


Figure 4-7: Excel spreadsheet with data.

The spreadsheet is saved in a file named `TestData1` in a disk in drive A. After the Current Directory is changed to drive A, the data is imported into MATLAB by assigning it to the variable `DATA`:

```
>> DATA = xlsread('TestData1')

DATA =
    11.0000    2.0000   34.0000   14.0000   -6.0000         0    8.0000
    15.0000    6.0000  -20.0000    8.0000    0.5600   33.0000    5.0000
     0.9000   10.0000    3.0000   12.0000  -25.0000   -0.1000    4.0000
    55.0000    9.0000    1.0000  -0.5550   17.0000    6.0000  -30.0000
```

### 4.5.2 Using the Import Wizard

Using the Import Wizard is probably the easiest way to import data into MATLAB since the user does not have to know, or to specify, the format of the data. The Import Wizard is activated by selecting **Import Data** in the **File** menu of the Command Window. (It can also be started by typing the command `uiimport`.) The Import Wizard starts by displaying a file selection box that shows all the data files recognized by the Wizard. The user then selects the file that contains the data to be imported, and clicks **Open**. The Import Wizard opens the file and displays a portion of the data in a preview box so that the user can verify that the data is the correct choice. The Import Wizard tries to process the data, and if the wizard is successful, it displays the variables it has created with a portion of the data. The user clicks **next** and the wizard shows the Column Separator that was used. If the variable has the correct data, the user can proceed with the wizard (click **next**); otherwise the user can choose a different Column Separator. In the next window the wizard shows the name and size of the variable to be created in MATLAB. (When the data is all numerical, the variable in MATLAB has the same name as the file from which the data was imported.) When the wizard ends (click **finish**), the data is imported to MATLAB.

As an example, the Import Wizard is used to import numerical ASCII data saved in a `.txt` file. The data saved with the file name `TestData2` is shown in Figure 4-8.

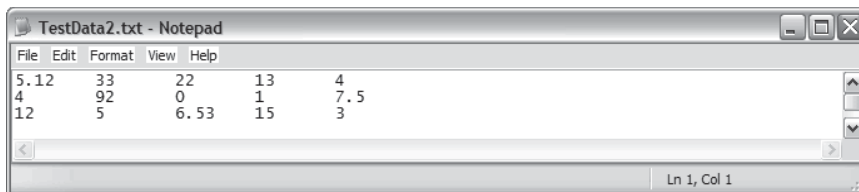


Figure 4-8: Numerical ASCII data.

The display of the Import Wizard during the import process for the TestData2 file is shown in Figures 4-9 and 4-10. Figure 4-10 shows that the name of the variable in MATLAB is TestData2 and its size is  $3 \times 5$ .

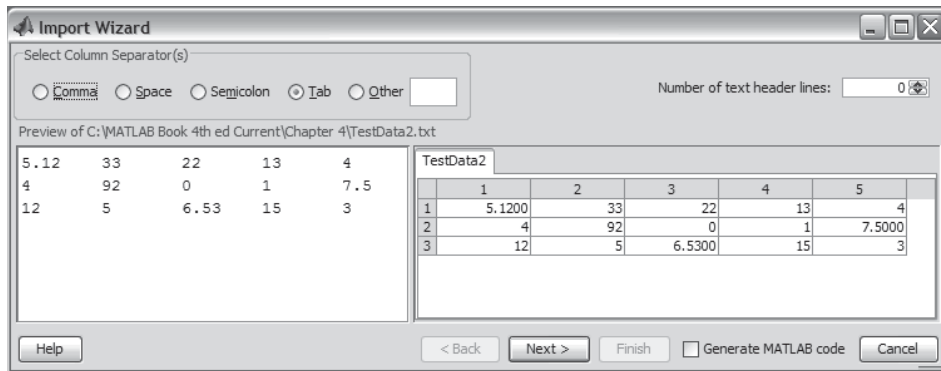


Figure 4-9: Import Wizard, first display.

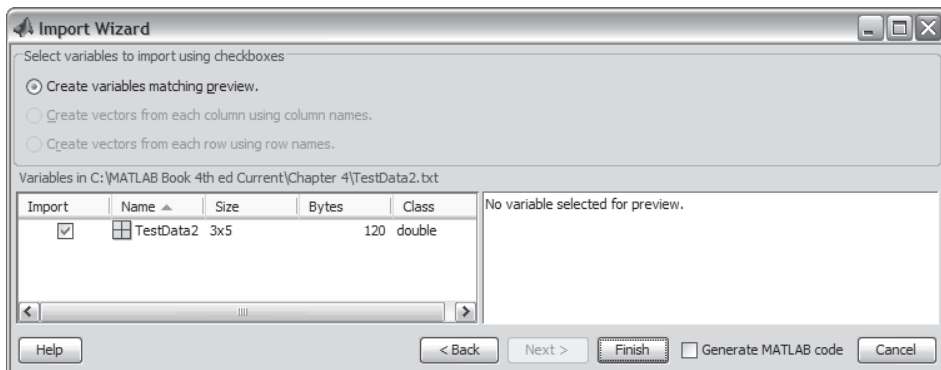


Figure 4-10: Import Wizard, second display.

In the Command Window of MATLAB, the imported data can be displayed by typing the name of the variable.

```
>> TestData2
TestData2 =
    5.1200    33.0000    22.0000    13.0000    4.0000
    4.0000    92.0000         0     1.0000    7.5000
   12.0000     5.0000    6.5300   15.0000    3.0000
```

## 4.6 EXAMPLES OF MATLAB APPLICATIONS

### Sample Problem 4-1: Height and surface area of a silo

A cylindrical silo with radius  $r$  has a spherical cap roof with radius  $R$ . The height of the cylindrical portion is  $H$ . Write a program in a script file that determines the height  $H$  for given values of  $r$ ,  $R$ , and the volume  $V$ . In addition, the program calculates the surface area of the silo.

Use the program to calculate the height and surface area of a silo with  $r = 30$  ft,  $R = 45$  ft, and a volume of 200,000 ft<sup>3</sup>. Assign values for  $r$ ,  $R$ , and  $V$  in the Command Window.

#### Solution

The total volume of the silo is obtained by adding the volume of the cylindrical part and the volume of the spherical cap. The volume of the cylinder is given by

$$V_{cyl} = \pi r^2 H$$

and the volume of the spherical cap is given by:

$$V_{cap} = \frac{1}{3}\pi h^2(3R - h)$$

where  $h = R - R\cos\theta = R(1 - \cos\theta)$ ,

and  $\theta$  is calculated from  $\sin\theta = \frac{r}{R}$ .

Using the equations above, the height,  $H$ , of the cylindrical part can be expressed by

$$H = \frac{V - V_{cap}}{\pi r^2}$$

The surface area of the silo is obtained by adding the surface areas of the cylindrical part and the spherical cap.

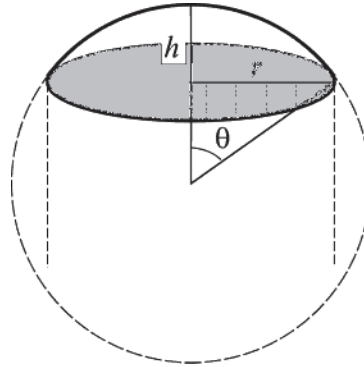
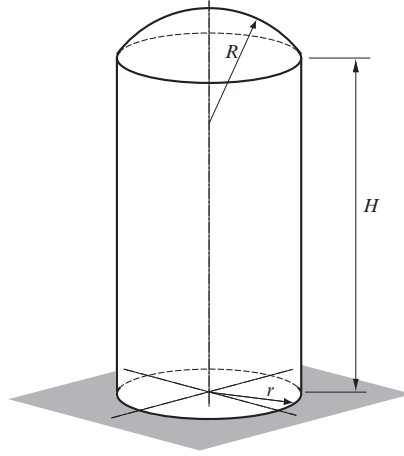
$$S = S_{cyl} + S_{cap} = 2\pi rH + 2\pi Rh$$

A program in a script file that solves the problem is presented below:

```
theta=asin(r/R);  
h=R*(1-cos(theta));
```

Calculating  $\theta$ .

Calculating  $h$ .





```
Vcap=pi*h^2*(3*R-h)/3;
H=(V-Vcap)/(pi*r^2);
S=2*pi*(r*H + R*h);
fprintf('The height H is: %f ft.',H)
fprintf('\nThe surface area of the silo is: %f square ft.',S)
```

Calculating the volume of the cap.

Calculating  $H$ .

Calculating the surface area

The Command Window where the script file, named silo, was executed is:

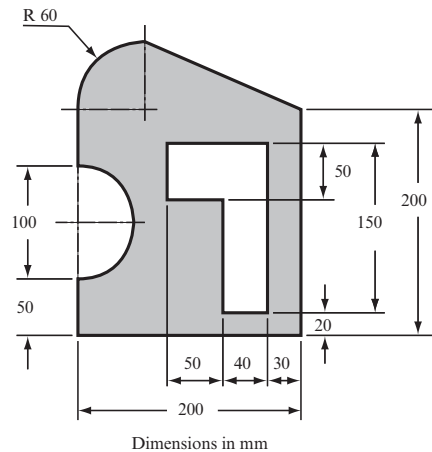
```
>> r=30; R=45; V=200000;
>> silo
The height H is: 64.727400 ft.
The surface area of the silo is: 15440.777753 square ft.
```

Assigning values to  $r$ ,  $R$ , and  $V$ .

Running the script file named

### Sample Problem 4-2: Centroid of a composite area

Write a program in a script file that calculates the coordinates of the centroid of a composite area. (A composite area can easily be divided into sections whose centroids are known.) The user needs to divide the area into sections and know the coordinates of the centroid (two numbers) and the area of each section (one number). When the script file is executed, it asks the user to enter the three numbers as a row in a matrix. The user enters as many rows as there are sections. A section that represents a hole is taken to have a negative area. For output, the program displays the coordinates of the centroid of the composite area. Use the program to calculate the centroid of the area shown in the figure.



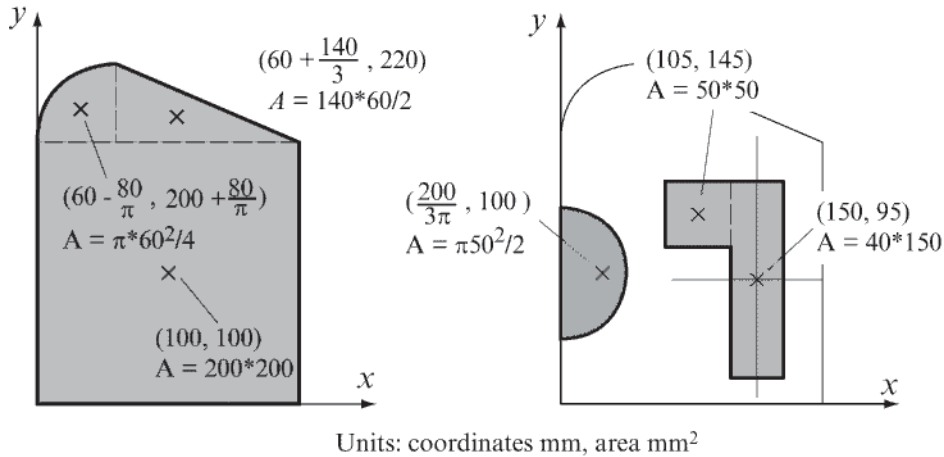
### Solution

The area is divided into six sections as shown in the following figure. The total area is calculated by adding the three sections on the left and subtracting the three sections on the right. The location and coordinates of the centroid of each section are marked in the figure, as well as the area of each section.

The coordinates  $\bar{X}$  and  $\bar{Y}$  of the centroid of the total area are given by  $\bar{X} = \frac{\sum A\bar{x}}{\sum A}$

and  $\bar{Y} = \frac{\sum A\bar{y}}{\sum A}$ , where  $\bar{x}$ ,  $\bar{y}$ , and  $A$  are the coordinates of the centroid and area of each section, respectively.

A script file with a program for calculating the coordinates of the centroid



of a composite area is provided below.

```
% The program calculates the coordinates of the centroid
% of a composite area.
clear C xs ys As
C=input('Enter a matrix in which each row has three ele-
ments.\nIn each row enter the x and y coordinates of the
centroid and the area of a section.\n');
xs=C(:,1)';
ys=C(:,2)';
As=C(:,3)';
A=sum(As);
x=sum(As.*xs)/A;
y=sum(As.*ys)/A;
fprintf('The coordinates of the centroid are: ( %f, %f
)\n',x,y)
```

Creating a row vector for the x coordinate of each section (first column of C).

Creating a row vector for the y coordinate of each section (second column of C).

Creating a row vector for the area of each section (third column of C).

Calculating the total area.

Calculating the coordinates of the centroid of the composite area.

The script file was saved with the name Centroid. The following shows the Command Window where the script file was executed.

```
>> Centroid
Enter a matrix in which each row has three elements.
In each row enter the x and y coordinates of the centroid
and the area of a section.
```

```
[100 100 200*200
60-80/pi 200+80/pi pi*60^2/4
60+140/3 220 140*60/2
200/(3*pi) 100 -pi*50^2/2
105 145 -50*50
150 95 -40*150]
```

The coordinates of the centroid are: ( 85.387547 , 131.211809 )

Entering the data for matrix C.  
Each row has three elements: the  
 $x$ ,  $y$ , and  $A$  of a section.

### Sample Problem 4-3: Voltage divider

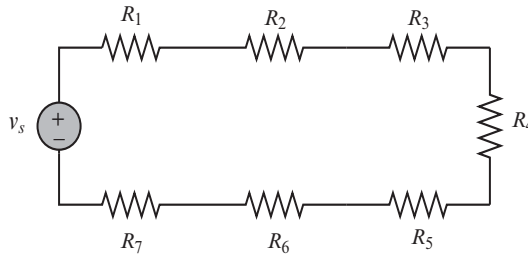
When several resistors are connected in an electrical circuit in series, the voltage across each of them is given by the voltage divider rule:

$$v_n = \frac{R_n}{R_{eq}} v_s$$

where  $v_n$  and  $R_n$  are the voltage across resistor  $n$  and its resistance, respectively,  $R_{eq} = \sum R_{eq}$  is the equivalent resistance, and  $v_s$  is the source voltage. The power dissipated in each resistor is given by:

$$P_n = \frac{R_n}{R_{eq}^2} v_s^2$$

The figure below shows a circuit with seven resistors connected in series.



Write a program in a script file that calculates the voltage across each resistor, and the power dissipated in each resistor, in a circuit that has resistors connected in series. When the script file is executed, it requests that the user first enter the source voltage and then to enter the resistances of the resistors in a vector. The program displays a table with the resistance listed in the first column, the voltage across the resistor in the second column, and the power dissipated in the resistor in the third column. Following the table, the program displays the current in the circuit and the total power.

Execute the file and enter the following data for  $v_s$  and the  $R$ 's.

$v_s = 24\text{V}$ ,  $R_1 = 20\Omega$ ,  $R_2 = 14\Omega$ ,  $R_3 = 12\Omega$ ,  $R_4 = 18\Omega$ ,  $R_5 = 8\Omega$ ,  $R_6 = 15\Omega$ ,  $R_7 = 10\Omega$ .

**Solution**

A script file that solves the problem is shown below.

```
% The program calculates the voltage across each resistor
% in a circuit that has resistors connected in series.
vs=input('Please enter the source voltage ');
Rn=input('Enter the values of the resistors as elements in a
row vector\n');
Req=sum(Rn);
vn=Rn*vs/Req;
Pn=Rn*vs^2/Req^2;
i = vs/Req;
Ptotal = vs*i;
Table = [Rn', vn', Pn'];
disp(' ')
disp(' Resistance Voltage   Power')
disp('      (Ohms)      (Volts)   (Watts)')
disp(' ')
disp(Table)
disp(' ')
fprintf('The current in the circuit is %f Amps.',i)
fprintf('\nThe total power dissipated in the circuit is %f
Watts.',Ptotal)
```

Calculate the equivalent resistance.

Apply the voltage divider rule.

Calculate the power in each resistor.

Calculate the current in the circuit.

Calculate the total power in the circuit.

Create a variable table with the vectors Rn, vn, and Pn as columns.

Display headings for the columns.

Display an empty line.

Display the variable Table.

The Command Window where the script file was executed is:

```
>> VoltageDivider
Please enter the source voltage 24
Enter the value of the resistors as elements in a row vector
[20 14 12 18 8 15 10]
Resistance Voltage Power
      (Ohms)      (Volts)   (Watts)
20.0000      4.9485      1.2244
14.0000      3.4639      0.8571
12.0000      2.9691      0.7346
18.0000      4.4536      1.1019
8.0000       1.9794      0.4897
```

Name of the script file.

Voltage entered by the user.

Resistor values entered as a vector.

|         |        |        |
|---------|--------|--------|
| 15.0000 | 3.7113 | 0.9183 |
| 10.0000 | 2.4742 | 0.6122 |

The current in the circuit is 0.247423 Amps.

The total power dissipated in the circuit is 5.938144 Watts.

## 4.7 PROBLEMS

Solve the following problems by first writing a program in a script file and then executing the program.

1. Body mass index (BMI) of a person is a measure of body fat based on height and weight. In U.S. customary units it is calculated by:

$$BMI = 703 \frac{W}{h^2}$$

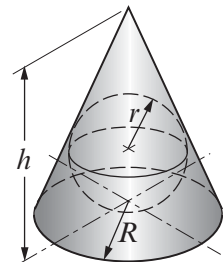
where  $W$  is the person's weight in pounds and  $h$  is the heights in inches. Write a MATLAB program in a script file that calculates the BMI. For input the program asks the user to enter his/her weight and height. The program then calculates the BMI rounded to the nearest tenth. For output the program displays the message: "The BMI is: XX." where XX is the value of the BMI. Determine the BMI of a 68-in.-tall person that weigh 162 lb.

2. The altitude,  $h$ , as a function of air pressure can be calculated by:

$$h = 145366.45 \left[ 1 - \left( \frac{p}{1013.25} \right)^{0.190289} \right]$$

where  $h$  is in units of feet and the pressure  $p$  in units of millibars (mb). Write a MATLAB program in a script file that calculates the  $h$  for a given  $p$ . For input the program asks the user to enter the pressure in units of millibars. The program then calculates the altitude rounded to the nearest integer. For output the program displays the message: "The altitude is: XX ft." where XX is the calculated value of  $h$ . Determine the altitude if the pressure is 394 mb.

3. Write a MATLAB program that determines the radius,  $r$ , of the largest sphere that can be inscribed inside a cone with base radius  $R$  and height  $h$ . For input the program asks the user to enter values for  $R$  and  $h$ . The program then calculates  $r$  rounded to the nearest tenth. For output the program displays the message: "The radius of the largest sphere that can be inscribed inside a cone with a base radius of XX in. and height of XX in., is: XX in." where XX are the corresponding numerical values. Use the



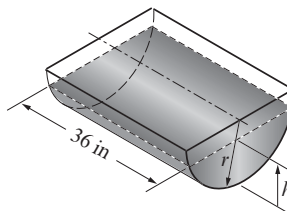
program to determine  $r$  for a cone with  $R = 8$  in. and  $h = 22$  in.

4. Radioactive decay can be modeled by the equation

$$A = A_0 e^{-kt}$$

where  $A$  is the amount at time  $t$ ,  $A_0$  is the amount at time  $t = 0$ , and  $k$  is a constant. Write a MATLAB program that calculates the amount of a radioactive material. When executed, the program asks the user to enter the half-life of the material (in years), the current amount of the material (in lb), and the number of years  $t$  from now for which the amount should be calculated. From this information the program first calculates the constant  $k$  and then the amount at  $t$  years. For output the program displays the message: “The amount of the material after XX years is XX kg” where XX are the corresponding numerical values. Use the program to determine how much plutonium-239 (half-life 24,110 years) will be left from 50 lb after 500 years.

5. A fuel tank is made of a half cylinder ( $r = 14$  in.) as shown. Derive an expression for the amount of fuel in gallons as a function of  $h$ . Create a vector for  $h$  ranging from 0 to 14 in. with increments of 2 in. Then calculate the corresponding volume rounded to the nearest tenth of a gallon. Display the results in a two-column table where in the first column are the values of  $h$  and in the second column the associated values of volume (in gallons).



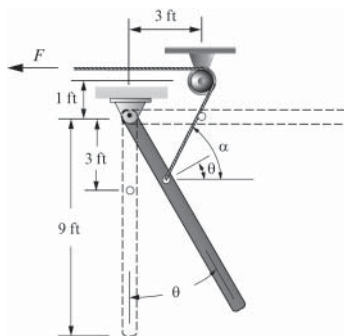
6. A 300-lb garage door is being opened by pulling on the cable as shown. As the door is lifted the force,  $F$ , in the cable, as a function of the angle  $\theta$ , is given by:

$$F = \frac{300 \cdot 4.5 \sin \theta}{3 \cos(\alpha - \theta)}$$

where

$$\sin \alpha = \frac{1 + 3 \cos \theta}{\sqrt{(1 + 3 \cos \theta)^2 + (3 - 3 \sin \theta)^2}}$$

Calculate  $F$  for  $\theta = 0^\circ$  through  $90^\circ$  with increments of  $10^\circ$ . Display the results in a two-column table.



7. Write a MATLAB program in a script file that calculates the average and the standard deviation of a list of grades as well as the number of grades on the list. The program asks the user (input command) to enter the grades as elements of a vector. The program then calculates the required quantities using MATLAB's built-in functions `length`, `mean`, and `std`. The results are displayed in the Command Window in the following format: “There are XX grades.” where XX is the numerical value.

“The average grade is XX.” where XX is the numerical value rounded to the nearest tenth.

“The standard deviation is XX.” where XX is the numerical value rounded to the nearest tenth.

Execute the program and enter the following grades: 93, 77, 51, 62, 99, 41, 82, 77, 71, 68, 100, 46, 78, 80, and 83.

8. The reduction of the amount of medication in the body can be modeled by the equation  $A = A_0 e^{kt}$ , where  $A$  is the amount at time  $t$ ,  $A_0$  is the amount at  $t = 0$ , and  $k$  is the decay constant ( $k < 0$ ). The half-life time of a certain medication is 3.5 h. A person takes 400 mg of the medication at  $t=0$ , and then additional 400 mg every 4 h. Determine the amount of the medication in a patient's body 23 h after taking the first dose.

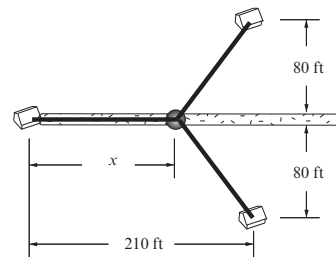
After determining the value of  $k$ , define a vector  $t = [23, 19, 15, 11, 7, 3]$  (the time since taking each dose) and calculate the corresponding values of  $A$ . Then use MATLAB's built-in function `sum` to determine the total amount.

9. The value of a saving account,  $V$ , after  $t$  years is given by:

$$V = P \left( 1 + \frac{r/100}{n} \right)^{nt}$$

where  $P$  is the initial investment,  $r$  is the yearly interest rate in % (e.g., 7.5% entered as 7.5), and  $n$  is the number of times per year that the interest is compounded. Write a MATLAB program in a script file that calculates  $V$ . When the program is executed, it asks the user to enter the amount of the initial investment, the number of years, the interest rate, and the number of times per year that the interest is compounded. The output is displayed in the following format: “The value of a \$XX investment at a yearly interest rate of X.X% compounded X times per year, after XX years is \$XXXX.XX”, where XXX stands for the corresponding quantities. Use the program to determine the value of a \$20,000 investment after 18 years if the yearly interest rate is 3.5% compounded 6 times a year.

10. The electricity supply cables of the three houses shown are connected to a pole as shown. Write a MATLAB program that determines the location of the pole (distance  $x$ ) that minimizes the total length of the cables needed. In the program define a vector  $x$  with values ranging from 50 to 200 with increments of 0.1. Use this vector to calculate the corresponding values of total length of the cables. Then use MATLAB's built-in function `min` to find the value of  $x$  that corresponds to the shortest length of cables.

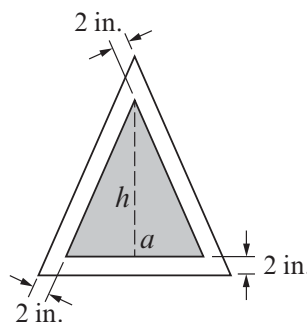


11. Early explorers often estimated altitude by measuring the temperature of boiling water. Use the following two equations to make a table that modern-day hikers could use for the same purpose.

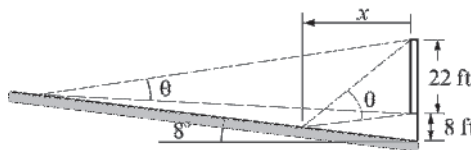
$$p = 29.921(1 - 6.8753 \times 10^{-6}h), \quad T_b = 49.161 \ln p + 44.932$$

where  $p$  is atmospheric pressure in inches of mercury,  $T_b$  is boiling temperature in  $^{\circ}\text{F}$ , and  $h$  is altitude in feet. The table should have two columns, the first altitude and the second boiling temperature. The altitude should range between  $-500$  ft and  $10,000$  ft at increments of  $500$  ft.

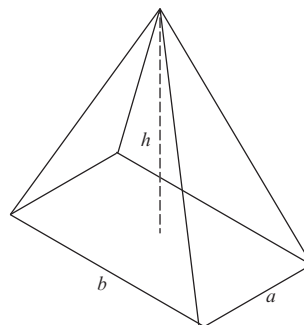
12. An isosceles triangle sign is designed to have a triangular printed area of  $600 \text{ in.}^2$  (shaded area with a base length of  $a$  and height of  $h$  in the figure). As shown in the figure, there is a 2-in. gap between the sides of the triangles. Write a MATLAB program that determine the dimensions  $a$  and  $h$  such that the overall area of the sign will be as small as possible. In the program define a vector  $a$  with values ranging from 10 to 120 with increments of 0.1. Use this vector for calculating the corresponding values of  $h$  and the overall area of the sign. Then use MATLAB's built-in function `min` to find the dimensions of the smallest sign.



13. The angle  $\theta$  at which a viewer sees the picture on the screen in a movie theater depends on the distance  $x$  from the screen. Write a MATLAB program that determines the angle  $\theta$  (in degrees) for viewers setting at distances of 20, 26, 32, 38, 44, 50, 56, 62, and 68 ft. Display the results in a two-column table.

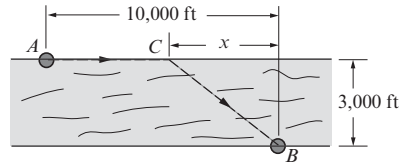


14. A 12-ft (144-in.) wire is cut into eight pieces which are welded together to form a pyramid as shown, such that in the rectangular base  $b = 1.9a$ . Write a MATLAB program that determines the dimensions  $a$  and  $b$  such that the volume of the pyramid will be as large as possible. In the program define a vector  $a$  with values ranging from 4 to 14 in. with increments of 0.01 in. Use this vector for calculating the corresponding values of  $b$ ,  $h$  and the volume. Then use MATLAB's built-in function `max` to find the dimensions of  $a$  and  $b$  that correspond to the pyramid with the largest volume.





15. A person at point  $A$  spots a child in trouble at point  $B$  across the river. The person can run at a speed of 8.6 ft/s and can swim at a speed of 3.9 ft/s. In order to reach the child in the shortest time the person runs to point  $C$  and then swims to point  $B$ , as shown. Write a MATLAB program that determines the distance  $x$  to point  $C$  that minimizes the time the person can reach the child. In the program define a vector  $x$  with values ranging from 0 to 5,000 with increments of 1. Use this vector to calculate the corresponding values of  $x$ . Then use MATLAB's built-in function `min` to find the value of  $x$  that corresponds to the shortest time.

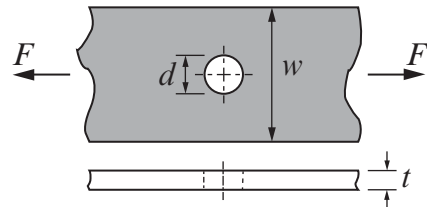


16. The maximum stress  $\sigma_{\max}$  at the edge of a hole (diameter  $d$ ) in a thin plate, with width  $w$  and thickness  $t$ , loaded by a tensile force  $F$  as shown is given by:

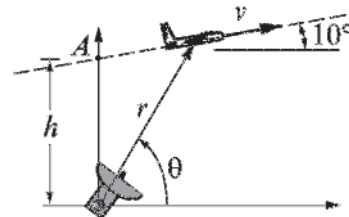
$$\sigma_{\max} = K_t \sigma_{\text{nom}}$$

where  $\sigma_{\text{nom}} = \frac{F}{t(w-d)}$  and  $K_t = 3 - 3.14(d/w) + 3.667(d/w)^2 - 1.527(d/w)^3$ .

Write a program in a script file that calculates  $\sigma_{\max}$ . The program should read the values of  $F$ ,  $w$ ,  $d$ , and  $t$  from an ASCII text file using the `load` command. The output should be in the form of a paragraph combining text and numbers—i.e., something like: “The maximum stress in a plate with a width of XX in. and thickness of XX in. and a hole of XX in. in diameter, due to a tensile force of XXX lb is XXXX psi, where XX stands for numerical values.” The stress should be rounded to the nearest integer. Use the program to calculate  $\sigma_{\max}$  when  $w = 2.5$  in.,  $d = 1.375$  in.,  $t = 0.1875$  in., and  $F = 8000$  lb.



17. The airplane shown is flying at a constant speed of  $v = 350$  mi/h along a straight path as shown. The airplane is being tracked by a radar station positioned a distance  $h = 1500$  ft below point  $A$ . The airplane is at point  $A$  at  $t = 0$ . Write a MATLAB program that calculates  $\theta$  and  $r$  as functions of time for  $t = 0, 0.5, 1, 1.5, \dots, 6$  s. Display the results in a three-column table where the first column is  $t$ , the second is the angle  $\theta$  in degrees, and the third is the corresponding value of  $r$ .



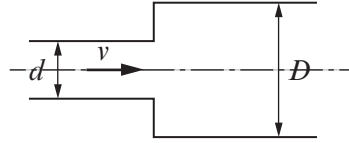
18. The intrinsic electrical conductivity  $\sigma$  of a semiconductor can be approximated by:

$$\sigma = e^{(C - \frac{E_g}{2kT})}$$

where  $\sigma$  is measured in  $(\Omega - m)^{-1}$ ,  $E_g$  is the band gap energy,  $k$  is Boltzmann's constant ( $8.62 \times 10^{-5}$  eV/K), and  $T$  is temperature in kelvins. For germanium,  $C = 13.83$  and  $E_g = 0.67$  eV. Write a program in a script file that calculates the intrinsic electrical conductivity for germanium for various temperatures. The values of the temperature should be read from an xls spreadsheet using the `xlsread` command. The output should be presented as a table where the first column is the temperature and the second column is the intrinsic electrical conductivity. Use the following values for temperature: 400, 435, 475, 500, 520, and 545 K.

19. The pressure drop  $\Delta p$  in pascals (Pa) for a fluid flowing in a pipe with a sudden increase in diameter is given by:

$$\Delta p = \frac{1}{2} \left[ 1 - \left( \frac{d}{D} \right)^2 \right]^2 \rho v^2$$



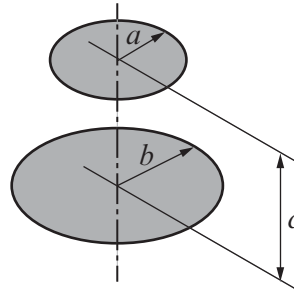
where  $\rho$  is the density of the fluid,  $v$ , the velocity of the flow, and  $d$  and  $D$  are defined in the figure. Write a program in a script file that calculates the pressure drop  $\Delta p$ . When the script file is executed, it requests the user to input the density in  $\text{kg/m}^3$ , the velocity in  $\text{m/s}$ , and values of the nondimensional ratio  $d/D$  as a vector. The program displays the inputted values of  $\rho$  and  $v$  followed by a table with the values of  $d/D$  in the first column and the corresponding values of  $\Delta p$  in the second column.

Execute the program assuming flow of gasoline ( $\rho = 737 \text{ kg/m}^3$ ) at  $v = 5 \text{ m/s}$  and the following ratios of diameters  $d/D = 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3$ .

20. The net heat exchange by radiation from plate 1 with radius  $b$  to plate 2 with radius  $a$  that are separated by a distance  $c$  is given by:

$$q = \sigma \pi b^2 F_{1-2} (T_1^4 - T_2^4)$$

where  $T_1$  and  $T_2$  are the absolute temperatures of the plates,  $\sigma = 5.669 \times 10^{-8} \text{ W/(m}^2 \cdot \text{K}^4)$  is the Stefan-Boltzmann constant, and  $F_{1-2}$  is a shape factor which, for the arrangement in the figure, is given by:



$$F_{1-2} = \frac{1}{2} \left[ Z - \sqrt{Z^2 - 4X^2Y^2} \right]$$

where  $X = a/c$ ,  $Y = c/b$ , and  $Z = 1 + (1 + X^2)Y^2$ . Write a script file that calculates the heat exchange  $q$ . For input the program asks the user to enter values for  $T_1$ ,  $T_2$ ,  $a$ ,  $b$ , and  $c$ . For output the program prints a summary of the geometry and temperatures and then prints the value of  $q$ . Use the script to calculate the results for  $T_1 = 400\text{ K}$ ,  $T_2 = 600\text{ K}$ ,  $a = 1\text{ m}$ ,  $b = 2\text{ m}$ , and  $c = 0.1, 1$ , and  $10\text{ m}$ .

21. The equation of a circle in a plane with radius  $R$  and a center at point  $(x_0, y_0)$  is given by:

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

The equation can also be written in the form:

$$-2x_0x - 2y_0y + c = -(x^2 + y^2) \text{ where } c = x_0^2 + y_0^2 - R^2$$

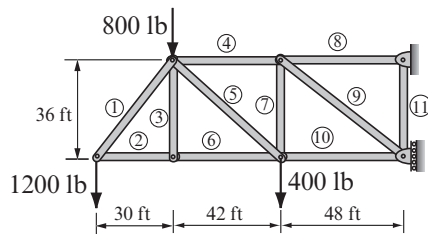
Given the coordinates of three points  $(x_1, y_1)$ ,  $(x_2, y_2)$ , and  $(x_3, y_3)$  it is possible to determine the radius and the coordinates of the center of the circle that passes through the three points. This is done by substituting the coordinate of each of the points in the equation and solving the system of three linear equations for  $x_0$ ,  $y_0$ , and  $c$ .

Write a program in a script file that calculates the coordinates of the center and the radius of a circle that passes through three given points. When executed the program asks the user to enter the coordinates of the three points. The program then calculates the center and radius and displays the results in the following format: "The coordinates of the center are (xx.x, xx.x) and the radius is xx.x.", where xx.x stands for the calculated quantities rounded to the nearest tenth. Execute the program entering the following three points: (11.5, 5), (3.2, 8.6), and  $(-4.5, -6.8)$ .

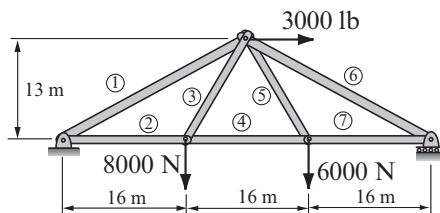
22. A truss is a structure made of members joined at their ends. For the truss shown in the figure, the forces in the 11 members are determined by solving the following system of 11 equations:

$$\begin{aligned} F_1 \cos 50^\circ + F_2 &= 0, & F_1 \sin 50^\circ - 400 &= 0, & 1200 \text{ lb} \\ -F_2 + F_6 &= 0 \\ -F_1 \cos 50^\circ + F_4 + F_5 \cos 41^\circ &= 0, \\ -F_1 \sin 50^\circ + F_3 + F_5 \sin 41^\circ - 800 &= 0, & -F_5 \cos 41^\circ - F_6 + F_{10} &= 0, \\ F_5 \sin 41^\circ + F_7 - 1200 &= 0, & -F_4 + F_8 + F_9 \cos 37^\circ &= 0, & -F_7 - F_9 \sin 37^\circ &= 0, \\ -F_9 \cos 37^\circ - F_{10} - 4933 &= 0, & F_9 \sin 37^\circ + F_{11} &= 0 \end{aligned}$$

Write the equations in matrix form and use MATLAB to determine the forces in the members. A positive force means tensile force and a negative force means compressive force. Display the results in a table where the first column displays the member number and the second column displays the corresponding force.



23. A truss is a structure made of members joined at their ends. For the truss shown in the figure, the forces in the seven members are determined by solving the following system of seven equations.



$$F_1 \cos 28.5^\circ + F_2 - 3000 = 0,$$

$$F_1 \sin 28.5^\circ + 6521 = 0,$$

$$-F_1 \cos 28.5^\circ - F_3 \cos 58.4^\circ + F_5 \cos 58.4^\circ + F_6 \cos 28.5^\circ + 3000 = 0,$$

$$-F_1 \sin 28.5^\circ - F_3 \sin 58.4^\circ - F_5 \sin 58.4^\circ - F_6 \sin 28.5^\circ = 0$$

$$-F_4 - F_5 \cos 58.4^\circ + F_7 = 0, \quad F_6 \sin 28.5^\circ + 7479 = 0 \quad -F_7 - F_6 \cos 28.5^\circ = 0$$

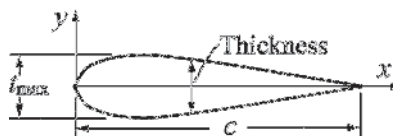
Write the equations in matrix form and use MATLAB to determine the forces in the members. A positive force means tensile force and a negative force means compressive force. Display the results in a table where the first column displays the member number and the second column displays the corresponding force.

24. The graph of the function  $f(x) = ax^3 + bx^2 + cx + d$  passes through the points  $(-1.2, 18.8)$ ,  $(0.2, 5)$ ,  $(2, 16)$ , and  $(3.5, 15)$ . Determine the constants  $a$ ,  $b$ ,  $c$ , and  $d$ . (Write a system of four equations with four unknowns, and use MATLAB to solve the equations.)

25. The graph of the function  $f(x) = ax^4 + bx^3 + cx^2 + dx + e$  passes through the points  $(-2.5, -62)$ ,  $(-1.5, -7.2)$ ,  $(-0.5, 8.3)$ ,  $(1, 3.7)$ , and  $(3, 45.7)$ . Determine the constants  $a$ ,  $b$ ,  $c$ ,  $d$ , and  $e$ . (Write a system of five equations with four unknowns, and use MATLAB to solve the equations.)

26. The surface of many airfoils can be described with an equation of the form

$$y = \pm \frac{tc}{0.2} \left[ a_0 \sqrt{x/c} + a_1(x/c) + a_2(x/c)^2 + a_3(x/c)^3 + a_4(x/c)^4 \right]$$



where  $t$  is the maximum thickness as a fraction of the chord length  $c$  (e.g.,  $t_{\max} = ct$ ). Given that  $c = 1$  m and  $t = 0.2$  m, the following values for  $y$  have been measured for a particular airfoil:

|         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|
| $x$ (m) | 0.15    | 0.35    | 0.5     | 0.7     | 0.85    |
| $y$ (m) | 0.08909 | 0.09914 | 0.08823 | 0.06107 | 0.03421 |

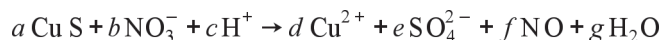
Determine the constants  $a_0$ ,  $a_1$ ,  $a_2$ ,  $a_3$ , and  $a_4$ . (Write a system of five equations and five unknowns, and use MATLAB to solve the equations.)

27. During a golf match, a certain number of points are awarded for each eagle and a different number for each birdie. No points are awarded for par, and a certain number of points are deducted for each bogey and a different number deducted for each double bogey (or worse). The newspaper report of an important match neglected to mention what these point values were, but did provide the following table of the results:

| Golfer | Eagles | Birdies | Pars | Bogeys | Doubles | Points |
|--------|--------|---------|------|--------|---------|--------|
| A      | 1      | 2       | 10   | 1      | 1       | 5      |
| B      | 2      | 3       | 11   | 0      | 1       | 12     |
| C      | 1      | 4       | 10   | 1      | 0       | 11     |
| D      | 1      | 3       | 10   | 2      | 0       | 8      |

From the information in the table write four equations in terms of four unknowns. Solve the equations for the unknown points awarded for eagles and birdies and points deducted for bogeys and double bogeys.

28. The dissolution of copper sulfide in aqueous nitric acid is described by the following chemical equation:



where the coefficients  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ ,  $f$ , and  $g$  are the numbers of the various molecules participating in the reaction and are unknown. The unknown coefficients are determined by balancing each atom on left and right and then balancing the ionic charge. The resulting equations are:

$$a = d, \quad a = e, \quad b = f, \quad 3b = 4e + f + g, \quad c = 2g, \quad -b + c = 2d - 2e$$

There are seven unknowns and only six equations. A solution can still be obtained, however, by taking advantage of the fact that all the coefficients must be positive integers. Add a seventh equation by guessing  $a = 1$  and solve the system of equations. The solution is valid if all the coefficients are positive integers. If this is not the case, take  $a = 2$  and repeat the solution. Continue the process until all the coefficients in the solution are positive integers.

29. The heat index  $HI$ , calculated from the air temperature and relative humidity, is the apparent temperature felt by the body. An equation used by the National Weather Service for calculating the  $HI$  is given by:

$$HI = -42.379 + 2.04901523T + 10.14333127R - 0.22475541TR - 6.83783 \times 10^{-3}T^2 - 5.481717 \times 10^{-2}R^2 + 1.22874 \times 10^{-3}T^2R + 8.5282 \times 10^{-4}TR^2 - 1.99 \times 10^{-6}T^2R^2$$

where  $T$  is the temperature in  $^{\circ}\text{F}$ , and  $R$  is the relative humidity in integer percentage. Write a MATLAB program in a script file that displays the following chart of heat index for given air temperature and relative humidity in

the Command Window:

| Relative<br>Humidity<br>(%) | Temperature (F) |    |    |    |     |     |     |     |
|-----------------------------|-----------------|----|----|----|-----|-----|-----|-----|
|                             | 80              | 82 | 84 | 86 | 88  | 90  | 92  | 94  |
| 50                          | 81              | 83 | 85 | 88 | 91  | 95  | 99  | 103 |
| 55                          | 81              | 84 | 86 | 89 | 93  | 97  | 101 | 106 |
| 60                          | 82              | 84 | 88 | 91 | 95  | 100 | 105 | 110 |
| 65                          | 82              | 85 | 89 | 93 | 98  | 103 | 108 | 114 |
| 70                          | 83              | 86 | 90 | 95 | 100 | 106 | 112 | 119 |
| 75                          | 84              | 88 | 92 | 97 | 103 | 109 | 116 | 124 |

30. The stress intensity factor  $K$  at a crack is given by

$K = C\sigma\sqrt{\pi a}$  where  $\sigma$  is the far-field stress,  $a$  is the crack length, and  $C$  is a parameter that depends on the geometry of the specimen and crack. For the case of the edge crack shown in the figure,  $C$  is given by:

$$C = \frac{1 - (a/b)/2 + 0.326(a/b)^2}{\sqrt{1 - a/b}}$$

Write a script file that will print out a table of values with the ratio  $a/b$  in the first column and the corresponding parameter  $C$  in the second column. Let  $a/b$  range between 0.05 and 0.80 with increments of 0.05.

