Chapter 2: Introduction to MATLAB Programming

Exercises

1) Write a simple script that will calculate the volume of a hollow sphere,

$$ V = \frac{4\pi}{3} (r_o^3 - r_i^3) $$

where \( r_i \) is the inner radius and \( r_o \) is the outer radius. Assign a value to a variable for the inner radius, and also assign a value to another variable for the outer radius. Then, using these variables, assign the volume to a third variable. Include comments in the script.

Ch2Ex1.m

% This script calculates the volume of a hollow sphere
%
% Assign values for the inner and outer radii
ri = 5.1
ro = 6.8
%
% Calculate the volume
vol = (4*pi)/3*(ro^3-ri^3)

2) The atomic weight is the weight of a mole of atoms of a chemical element. For example, the atomic weight of oxygen is 15.9994 and the atomic weight of hydrogen is 1.0079. Write a script that will calculate the molecular weight of hydrogen peroxide, which consists of two atoms of hydrogen and two atoms of oxygen. Include comments in the script. Use help to view the comment in your script.

Ch2Ex2.m

% Calculates the molecular weight of hydrogen peroxide
%
% Initialize the atomic weights for oxygen and hydrogen
atWtOxygen = 15.9994;
atWtHydrogen = 1.0079;
%
% Hydrogen peroxide is 2 atoms of hydrogen and 2 of oxygen
molWtHydrogenPeroxide = 2*atWtHydrogen + 2 * atWtOxygen

>> help ch2ex2
Calculates the molecular weight of hydrogen peroxide

3) Write an input statement that will prompt the user for the name of a chemical element as a string. Then, find the length of the string.
>> elemname = input('Enter a chemical element: ', 's');  
Enter a chemical element: hydrogen  
>> length(elemname)  
an = 8

4) Write an **input** statement that will prompt the user for a real number, and store it in a variable. Then, use the **fprintf** function to print the value of this variable using 2 decimal places.

    >> realnum = input('Enter a real number: ');  
Enter a real number: 45.789  
    >> fprintf('The number is %.2f\n', realnum)  
The number is 45.79

5) The **input** function can be used to enter a vector, such as:

    >> vec = input('Enter a vector: ')  
Enter a vector: 4:7  
vec =  
   4  5  6  7

Experiment with this, and find out how the user can enter a matrix.

    >> mat = input('Enter a matrix: ')  
Enter a matrix: [4:6; 9:11]  
mat =  
   4  5  6  
   9 10 11

    >> mat = input('Enter a matrix: ')  
Enter a matrix: zeros(2)  
mat =  
   0  0  
   0  0

6) Experiment, in the Command Window, with using the **fprintf** function for real numbers. Make a note of what happens for each. Use **fprintf** to print the real number 12345.6789.

    realnum = 12345.6789;

    * without specifying any field width

    >> fprintf('The number is %f\n', realnum)
The number is 12345.678900

• in a field width of 10 with 4 decimal places

```matlab
>> fprintf('The number is %10.4f\n', realnum)
The number is 12345.6789
```

• in a field width of 10 with 2 decimal places

```matlab
>> fprintf('The number is %10.2f\n', realnum)
The number is 12345.68
```

• in a field width of 6 with 4 decimal places

```matlab
>> fprintf('The number is %6.4f\n', realnum)
The number is 12345.6789
```

• in a field width of 2 with 4 decimal places

```matlab
>> fprintf('The number is %2.4f\n', realnum)
The number is 12345.6789
```

7) Experiment, in the Command Window, with using the `fprintf` function for integers. Make a note of what happens for each. Use `fprintf` to print the integer 12345.

```matlab
intnum = 12345;

• without specifying any field width

```matlab
>> fprintf('The number is %d\n', intnum)
The number is 12345
```

• in a field width of 5

```matlab
>> fprintf('The number is %5d\n', intnum)
The number is 12345
```

• in a field width of 8

```matlab
>> fprintf('The number is %8d\n', intnum)
The number is 12345
```

• in a field width of 3

```matlab
>> fprintf('The number is %3d\n', intnum)
```
The number is 12345

8) Create the following variables
    \[
    x = 12.34;
    y = 4.56;
    \]
    Then, fill in the \texttt{fprintf} statements using these variables that will accomplish the following:
    \[
    \begin{align*}
    &\text{>> fprintf('x is %8.3f
\n', x)} \\
    &\quad \text{x is 12.340} \\
    &\text{>> fprintf('x is %.f
\n', x)} \\
    &\quad \text{x is 12} \\
    &\text{>> fprintf('y is %.1f
\n', y)} \\
    &\quad \text{y is 4.6} \\
    &\text{>> fprintf('y is %-8.1f!
\n', y)} \\
    &\quad \text{y is 4.6 !}
    \end{align*}
    \]

9) Write a script to prompt the user for the length and width of a rectangle, and print its area with 2 decimal places. Put comments in the script.

\texttt{Ch2Ex9.m}

\begin{verbatim}
% Calculate the area of a rectangle
% Prompt the user for the length and width
length = input('Enter the length of the rectangle: '); 
width = input('Enter the width of the rectangle: '); 

% Calculate and print the area 
rect_area = length * width; 
fprintf('The area of the rectangle is %.2f
', rect_area) 
\end{verbatim}

10) Write a script called \texttt{echoname} that will prompt the user for his or her name, and then echo print the name in a sentence in the following format (use \%s to print it):
    \[
    \begin{align*}
    &\text{>> echoname} \\
    &\quad \text{What is your name? Susan} \\
    &\quad \text{Wow, your name is Susan!} \\
    \end{align*}
    \]

\texttt{echoname.m}

\begin{verbatim}
% Prompt the user and echo print name
iname = input('What is your name? ','s'); 
fprintf('Wow, your name is %s!
',iname) 
\end{verbatim}

11) Write a script called \texttt{echostring} that will prompt the user for a string, and will echo print the string in quotes:
    \[
    \begin{align*}
    &\text{>> echostring} \\
    \end{align*}
    \]
Enter your string: hi there
Your string was: 'hi there'

echostring.m

% Prompt the user and print a string in quotes
str = input('Enter your string: ', 's');
fprintf('Your string was: ''%s''
',str)

12) In the metric system, fluid flow is measured in cubic meters per second (m³/s). A cubic foot per second (ft³/s) is equivalent to 0.028 m³/s. Write a script titled flowrate that will prompt the user for flow in cubic meters per second and will print the equivalent flow rate in cubic feet per second. Here is an example of running the script. Your script must produce output in exactly the same format as this:

>> flowrate
Enter the flow in m^3/s: 15.2
A flow rate of 15.200 meters per sec
is equivalent to 542.857 feet per sec

flowrate.m

% Converts a flow rate from cubic meters per second
% to cubic feet per second

cubMperSec = input('Enter the flow in m^3/s :');
cubFperSec = cubMperSec / .028;
fprintf('A flow rate of %.3f meters per sec\n', ...
cubMperSec)
fprintf('is equivalent to %.3f feet per sec\n', ...
cubFperSec)

13) On average, people in a region spend 8% to 10% of their income on food. Write a script that will prompt the user for an annual income. It will then print the range that would typically be spent on food annually. Also, print a monthly range.

Ch2Ex13.m

% Calculates and prints the likely $ amount spent on food
% based on annual income

income = input('Enter your annual income: '); 
fprintf('You are likely to spend between $%.2f\n',.08*income)
fprintf('and $%.2f annually on food.\n\n', .1*income)

% print the monthly range
fprintf('You are likely to spend between $%.2f\n',.08*income/12)
fprintf('and $%.2f monthly on food.\n', .1*income/12)
14) Wing loading, which is the airplane weight divided by the wing area, is an important design factor in aeronautical engineering. Write a script that will prompt the user for the weight of the aircraft in kilograms, and the wing area in meters squared, and will calculate and print the wing loading of the aircraft in kilograms per square meter.

Ch2Ex14.m

```matlab
% Calculates the wing loading for an airplane
% Prompt the user for the weight and wing area of the plane
plane_weight = input('Enter the weight of the airplane: ');
wing_area = input('Enter the wing area: ');

% Calculate and print the wing loading
fprintf('The wing loading is %.2f
', plane_weight/wing_area)
```

15) Write a script that assigns values for the x coordinate and then y coordinate of a point, and then plots this using a green +.

Ch2Ex15.m

```matlab
% Prompt the user for the coordinates of a point and plot
% the point using a green +
x = input('Enter the x coordinate: ');
y = input('Enter the y coordinate: ');
plot(x,y, 'g+')
```

16) Plot $\exp(x)$ for values of $x$ ranging from -2 to 2 in steps of 0.1. Put an appropriate title on the plot, and label the axes.

Ch2Ex16.m

```matlab
% Plots exp(x)
x = -2:0.1:2;
y = exp(x);
plot(x,y, '*')
title('Exp(x)')
xlabel('x')
ylabel('exp(x)')
```
17) Create a vector $x$ with values ranging from 1 to 100 in steps of 5. Create a vector $y$ which is the square root of each value in $x$. Plot these points. Now, use the `bar` function instead of `plot` to get a bar chart instead.

```matlab
% Plots same x and y points using bar and plot
clf
x = 1:5:100;
y = sqrt(x);
plot(x,y,'*')
title('sqrt(x)')
figure(2)
bar(x,y)
title('sqrt(x)')
```

18) Create a $y$ vector which stores random integers in the 1 to 100 range. Create an $x$ vector which iterates from 1 to the length of the $y$ vector. Experiment with the `plot` function using different colors, line types, and plot symbols.

```matlab
% Experiment with plot symbols and line types
y = randi([1, 100],1,15)
x = 1:length(y);
clf
plot(x,y,'rv')
figure(2)
plot(x,y,'g*:')
```

19) Plot $\sin(x)$ for $x$ values ranging from 0 to $\pi$ (in separate Figure Windows):
- using 10 points in this range
- using 100 points in this range

```matlab
% Plots sin(x) with 10 points and 100 points in range 0 to pi
x = linspace(0,pi,10);
y = sin(x);
clf
figure(1)
plot(x,y,'k*')
title('sin(x) with 10 points')
figure(2)
x = linspace(0,pi);
y = sin(x);
```
20) Atmospheric properties such as temperature, air density, and air pressure are important in aviation. Create a file that stores temperatures in degrees Kelvin at various altitudes. The altitudes are in the first column and the temperatures in the second. For example, it may look like this:

<table>
<thead>
<tr>
<th>Altitude</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>288</td>
</tr>
<tr>
<td>2000</td>
<td>281</td>
</tr>
<tr>
<td>3000</td>
<td>269</td>
</tr>
<tr>
<td>5000</td>
<td>256</td>
</tr>
<tr>
<td>10000</td>
<td>223</td>
</tr>
</tbody>
</table>

Write a script that will load this data into a matrix, separate it into vectors, and then plot the data with appropriate axis labels and a title.

Ch2Ex20.m

```matlab
% Read altitudes and temperatures from a file and plot
load alttemps.dat
altitudes = alttemps(:,1);
temps = alttemps(:,2);
plot(altitudes,temps,'k*')
xlabel('Altitudes')
ylabel('Temperatures')
title('Atmospheric Data')
```

21) Create a $3 \times 6$ matrix of random integers, each in the range from 50 to 100. Write this to a file called `randfile.dat`. Then, create a new matrix of random integers, but this time make it a $2 \times 6$ matrix of random integers, each in the range from 50 to 100. Append this matrix to the original file. Then, read the file in (which will be to a variable called `randfile`) just to make sure that worked!

```matlab
>> mat = randi([50,100], 3,6)
mat =
   91   96   64   99   98   57
   96   82   77   58   74   71
   56   54   98   99   90   96
>> save randfile.dat mat -ascii
>> newmat = randi([50,100], 2,6)
newmat =
   90   83   93   84   87   83
   98   51   97   88   70   58
>> save randfile.dat newmat -ascii -append
>> load randfile.dat
>> randfile
randfile =
```
22) Create a file called “testtan.dat” comprised of two lines with three real numbers on each line (some negative, some positive, in the -1 to 3 range). The file can be created from the Editor, or saved from a matrix. Then, load the file into a matrix and calculate the tangent of every element in the resulting matrix.

```matlab
>> mat = rand(2,3)*4-1
mat =
   1.8242    0.1077   -0.6115
  -0.8727   -0.8153    2.2938
>> save testtan.dat mat -ascii
>> load testtan.dat
>> tan(testtan)
an =
  -3.8617    0.1081   -0.7011
  -1.1918   -1.0617   -1.1332
```

23) Write a function `calcrectarea` that will calculate and return the area of a rectangle. Pass the length and width to the function as input arguments.

```
calcrectarea.m
function area = calcrectarea(length, width)
    % This function calculates the area of a rectangle
    % Format of call: calcrectarea(length, width)
    % Returns the area

    area = length * width;
end
```

24) Write a function called `fn` that will calculate y as a function of x, as follows:

\[ y = x^3 - 4x^2 + \sin(x) \]

Here are two examples of using the function:

```matlab
>> help fn
    Calculates y as a function of x

>> y = fn(7)
y =
   147.6570
```
fn.m

function out = fn(x)
% Calculates y as a function of x

% Format of call: fn(x)
% Returns x^2 - 4*x^2 + sin(x)

out = x^3 - 4*x^2 + sin(x);
end

Renewable energy sources such as biomass are gaining increasing attention. Biomass energy units include megawatt hours (MWh) and gigajoules (GJ). One MWh is equivalent to 3.6 GJ. For example, one cubic meter of wood chips produces 1 MWh.

25) Write a function mwh_to_gj that will convert from MWh to GJ. Here are some examples of using the function.

    >> mwh = 3.3;
    >> gj = mwh_to_gj(mwh)
    gj =
         11.8800

    >> disp(mwh_to_gj(1.1))
    3.9600

    >> help mwh_to_gj
    Converts from MWh to GJ

mwh_to_gj.m

function out = mwh_to_gj(mwh)
% Converts from MWH to GJ

% Format of call: mwh_to_gj(mwh)
% Returns gigajoules

out = mwh * 3.6;
end

26) The velocity of an aircraft is typically given in either miles/hour or meters/second. Write a function that will receive one input argument, the velocity of an airplane in miles per hour and will return the velocity in meters per second. The relevant conversion factors are: one hour = 3600 seconds, one mile = 5280 feet, and one foot = .3048 meters.

convertVel.m
function velmps = convertVel(velmph)
% Convert the velocity of an aircraft from % miles per hour to meters per second % Format of call: convertVel(mph) % Returns velocity in mps
velmps = velmph/3600*5280*.3048;
end

27) If a certain amount of money (called the principal P) is invested in a bank account, earning an interest rate i compounded annually, the total amount of money \( T_n \) that will be in the account after \( n \) years is given by:
\[
T_n = P \left(1 + i\right)^n
\]
Write a function that will receive input arguments for \( P, i, \) and \( n \), and will return the total amount of money \( T_n \). Also, give an example of calling the function.

function totMoney = account(p,i,n)
% Calculates the amount of money in an account % after n years at interest rate i with a % principal p invested
% Format of call: account(p,i,n) % Returns total of money after n years
totMoney = p * (1+i)^n;
end

>> format bank
>> total = account(50000,0.05,10)
total =
81444.73

28) List some differences between a script and a function.

- A function has a header whereas a script does not.
- A function typically has end at the `end` of the file.
- A function is called whereas a script is executed.
- Arguments are passed to functions but not to scripts.
- Functions can return arguments whereas scripts cannot.
- The block comment is typically in the beginning of a script but under the function header.

29) The velocity of a moving fluid can be found from the difference between the total and static pressures \( P_t \) and \( P_s \). For water, this is given by

\[
V = 1.016 \sqrt{P_t - P_s}
\]
Write a function that will receive as input arguments the total and static pressures and will return the velocity of the water.

`fluidvel.m`

```matlab
function waterVel = fluidvel(totp, statp)
% Calculates the velocity of water given the
% total and static pressures
% Format: fluidvel(total pressure, static pressure)
% Returns the velocity of the water

waterVel = 1.016 * sqrt(totp-statp);
end
```

30) For a project, some biomedical engineering students are designing a device that will monitor a person’s heart rate while on a treadmill. The device will let the subject know when the target heart rate has been reached. A simple calculation of the target heart rate (THR) for a moderately active person is

\[ \text{THR} = (220 - A) \times 0.6 \]

where \( A \) is the person’s age. Write a function that will calculate and return the THR.

`findthr.m`

```matlab
function thr = findthr(age)
% Calculate the target heart rate for
% a person of a given age
% Format of call: findthr(age)
% Returns target heart rate

thr = (220 - age) * 0.6;
end
```

31) An approximation for a factorial can be found using Stirling’s formula:

\[ n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \]

Write a function to implement this, passing the value of \( n \) as an argument.

`stirling.m`

```matlab
function approxNFact = stirling(n)
% Approximate n! using Stirling's formula
% Format of call: stirling(n)
% Returns approximate factorial of n

approxNFact = sqrt(2*pi*n) * (n/exp(1))^n;
end
```
32) The cost of manufacturing \( n \) units (where \( n \) is an integer) of a particular product at a factory is given by the equation:

\[
C(n) = 5n^2 - 44n + 11
\]

Write a script \texttt{mfgcost} that will

- prompt the user for the number of units \( n \)
- call a function \texttt{costn} that will calculate and return the cost of manufacturing \( n \) units
- print the result (the format must be exactly as shown below)

Next, write the function \texttt{costn}, which simply receives the value of \( n \) as an input argument, and calculates and returns the cost of manufacturing \( n \) units.

Here is an example of executing the script:

```matlab
>> mfgcost
Enter the number of units: 100
The cost for 100 units will be $45611.00
```

\texttt{mfgcost.m}

```matlab
n = input('Enter the number of units: ');
costPerN = costn(n);
fprintf('The cost for %d units will be ',n)
fprintf('$%.2f\n', costPerN)
```

\texttt{costn.m}

```matlab
function out = costn(n)
% Calculates the cost of manufacturing \( n \) units of a product
% Format of call: costn(n)
% Returns cost as 5*\( n^2 \)-44*\( n \)+11

out = 5*n^2 - 44*n + 11;
end
```

33) The conversion depends on the temperature and other factors, but an approximation is that 1 inch of rain is equivalent to 6.5 inches of snow. Write a script that prompts the user for the number of inches of rain, calls a function to return the equivalent amount of snow, and prints this result. Write the function, as well!

\texttt{Ch2Ex33.m}

```matlab
% Prompt the user for a number of inches of rain
```
% and call a function to calculate the equivalent amount of snow

rain = input('How much rain in inches: ');  
snow = rainToSnow(rain);  
fprintf('%.1f inches of rain would be ', rain)  
fprintf('%.1f inches of snow\n', snow)

rainToSnow.m

function outsnow = rainToSnow(rain)  
% Calculate equivalent amount of snow  
% given rainfall in inches  
% Format of call: rainToSnow(rain)  
% Returns equivalent snowfall  

outsnow = rain * 6.5;  
end

34) The volume $V$ of a regular tetrahedron is given by 
$V = \frac{1}{12} \sqrt{2} s^3$ where $s$ is the length of the sides of the equilateral triangles that form the faces of the tetrahedron. Write a program to calculate such a volume. The program will consist of one script and one function. The function will receive one input argument, which is the length of the sides, and will return the volume of the tetrahedron. The script will prompt the user for the length of the sides, call the function to calculate the volume, and print the result in a nice sentence format. For simplicity, we will ignore units.

Ch2Ex34.m

% Calculates and prints the volume of a regular tetrahedron  
% Prompts user for length of sides, calls a function to calculate the volume, and prints the result  

len = input('Enter the length of the sides: ');  
volume = tetVol(len);  
fprintf('The volume of a regular tetrahedron with ')  
fprintf(' sides of length %.1f is %.2f\n', len, volume)

tetVol.m

function vol = tetVol(len)  
% Calculates the volume of a regular tetrahedron  
% Format of call: tetVol(side length)  
% Returns volume  

vol = 1/12 * sqrt(2) * len^3;
35) Write a function that is called `pickone`, which will receive one input argument `x`, which is a vector, and will return one random element from the vector. For example,

```matlab
>> pickone(4:7)
an =
  5

>> disp(pickone(-2:0))
-1

>> help pickone
pickone(x) returns a random element from vector x
```

```matlab
pickone.m
function elem = pickone(invec)
% pickone(x) returns a random element from vector x
% Format of call: pickone(vector)
% Returns random element from the vector

len = length(invec);
r = randi([1, len]);
elem = invec(r);
end
```

36) A function can return a vector as a result. Write a function `vecout` that will receive one integer argument and will return a vector that increments from the value of the input argument to its value plus 5, using the colon operator. For example,

```matlab
>> vecout(4)
an =
   4   5   6   7   8   9
```

```matlab
vecout.m
function outvec = vecout(innum)
% Create a vector from innum to innum + 5
% Format of call: vecout(input number)
% Returns a vector input num : input num+5

outvec = innum:innum+5;
end
```
37) If the lengths of two sides of a triangle and the angle between them are known, the length of the third side can be calculated. Given the lengths of two sides (b and c) of a triangle, and the angle between them α in degrees, the third side a is calculated as follows:

\[ a^2 = b^2 + c^2 - 2bc \cos(\alpha) \]

Write a script `thirdside` that will prompt the user and read in values for b, c, and α (in degrees), and then calculate and print the value of a with 3 decimal places. (Note: To convert an angle from degrees to radians, multiply the angle by \( \pi/180 \)).

The format of the output from the script should look exactly like this:

```
>> thirdside
Enter the first side: 2.2
Enter the second side: 4.4
Enter the angle between them: 50

The third side is 3.429
```

For more practice, write a function to calculate the third side, so the script will call this function.

```
thirdside.m

% Calculates the third side of a triangle, given
% the lengths of two sides and the angle between them

b = input('Enter the first side: ');
c = input('Enter the second side: ');
alpha = input('Enter the angle between them: ');
alpha = alpha * pi / 180;

a = sqrt(b^2 + c^2 - 2*b*c*cos(alpha));
fprintf('
The third side is %.3f
', a)
```

38) A part is being turned on a lathe. The diameter of the part is supposed to be 20,000 mm. The diameter is measured every 10 minutes and the results are stored in a file called `partdiam.dat`. Create a data file to simulate this. The file will store the time in minutes and the diameter at each time. Plot the data.

```
partdiam.dat
0  25233
10 23432
20 21085
30 20374
40 20002
```

```
Ch2Ex38.m

% Read from a data file the diameter of a part every 10 minutes
```
% as it is turned on a lathe and plot this data

load partdiam.dat
mins = partdiam(:,1);
diams = partdiam(:,2);
plot(mins,diams,'k*')
xlabel('minutes')
ylabel('part diameter')

39) A file “floatnums.dat” has been created for use in an experiment. However, it contains float (real) numbers and what is desired instead is integers. Also, the file is not exactly in the correct format; the values are stored columnwise rather than rowwise. For example, if the file contains the following:

<table>
<thead>
<tr>
<th>90.5792</th>
<th>27.8498</th>
<th>97.0593</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.6987</td>
<td>54.6882</td>
<td>95.7167</td>
</tr>
<tr>
<td>91.3376</td>
<td>95.7507</td>
<td>48.5376</td>
</tr>
<tr>
<td>63.2359</td>
<td>96.4889</td>
<td>80.0280</td>
</tr>
<tr>
<td>9.7540</td>
<td>15.7613</td>
<td>14.1886</td>
</tr>
</tbody>
</table>

what is really desired is:

<table>
<thead>
<tr>
<th>91</th>
<th>13</th>
<th>91</th>
<th>63</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>55</td>
<td>96</td>
<td>96</td>
<td>16</td>
</tr>
<tr>
<td>97</td>
<td>96</td>
<td>49</td>
<td>80</td>
<td>14</td>
</tr>
</tbody>
</table>

Create the data file in the specified format. Write a script that would read from the file $floatnums.dat$ into a matrix, round the numbers, and write the matrix in the desired format to a new file called $intnums.dat$.

Ch2Ex39.m

% Reads in float numbers stored column-wise from a file, 
% rounds them to integers and writes row-wise to a new file

load floatnums.dat
inums = round(floatnums)';
save intnums.dat inums -ascii

40) A file called $costssales.dat$ stores for a company some cost and sales figures for the last n quarters (n is not defined ahead of time). The costs are in the first column, and the sales are in the second column. For example, if five quarters were represented, there would be five lines in the file, and it might look like this:

| 1100 | 800 |
| 1233 | 650 |
| 1111 | 1001 |
| 1222 | 1300 |
Write a script called `salescosts` that will read the data from this file into a matrix. When the script is executed, it will do three things. First, it will print how many quarters were represented in the file, such as:

```matlab
>> salescosts
There were 5 quarters in the file
```

Next, it will plot the costs using black circles and sales using black stars (*) in a Figure Window with a legend (using default axes) as seen in Figure 2.8.

![Figure 2.8 Plot of Cost and Sales data](image)

Finally, the script will write the data to a new file called `newfile.dat` in a different order. The sales will be the first row, and the costs will be the second row. For example, if the file is as shown above, the resulting file will store the following:

```
800 650 1001 1300 1221
1100 1233 1111 1222 999
```

It should not be assumed that the number of lines in the file is known.

```matlab
salescosts.m
load costssales.dat
costs = costssales(:,1);
sales = costssales(:,2);
```
len = length(costs); % or sales

x = 1:len;  % Note: not necessary
plot(x,costs,'ko')
hold on
plot(x,sales,'k*')
legend('Costs', 'Sales')
fprintf('There were %d quarters in the file
', len)
neworder = rot90(costssales)
% neworder = flipud(costssales');
save newfile.dat neworder -ascii