Data Processing with Matlab for the Experimental Physics Laboratory

Amrozia Shaheen, Muhammad Wasif,
Waqas Mahmood and Muhammad Sabieh Anwar
LUMS School of Science and Engineering

September 27, 2011

Data analysis and representation are vital steps in any experimental exercise. They lend meaning to the experiment and provide insight leading to a more fundamental understanding of the underlying concept. Intelligent data processing and representation also help the experimenter in re-designing the experiment for increased accuracy and precision. Clever thinking may even encourage her to adapt and tailor the procedural steps to elicit some otherwise hidden facet.

In the experimental physics lab, we will use Matlab for,

- analyzing experimental data and computing errors,
- curve fitting, and
- graphically representing experimental data.

The present write-up serves as a first introduction to Matlab. Students who are not familiar with Matlab, or even with the computer, need not to worry. We will proceed slowly, allowing everyone to familiarize and acclimatize with the culture of computing. Luckily, Matlab is a highly user-friendly and interactive package that is very easy to learn. Furthermore, subsequent laboratory sessions will give all of us ample opportunity to practice Matlab.

It is important that every student independently works through all the examples given in this hand-out and attempts all challenge questions. These challenge questions are labelled with the box Q.

APPROXIMATE PERFORMANCE TIME 6-8 hours of independent work.

1 Introduction to Matlab

Matlab is a technical computing language and interactive environment for data visualization, data analysis, algorithm development and numerical computation. It is an important and versatile tool used to perform basic arithmetic operations, graph functions, solve equations, accomplish statistical tests and much more.

1.1 Starting Matlab

You can start Matlab by clicking the Matlab icon located on the Desktop, a special window called the Matlab desktop. The desktop is a window that contains other windows and a **Start** button similar to that used in Window 2000 or XP. The default configuration of the Matlab desktop is shown in Figure (1).

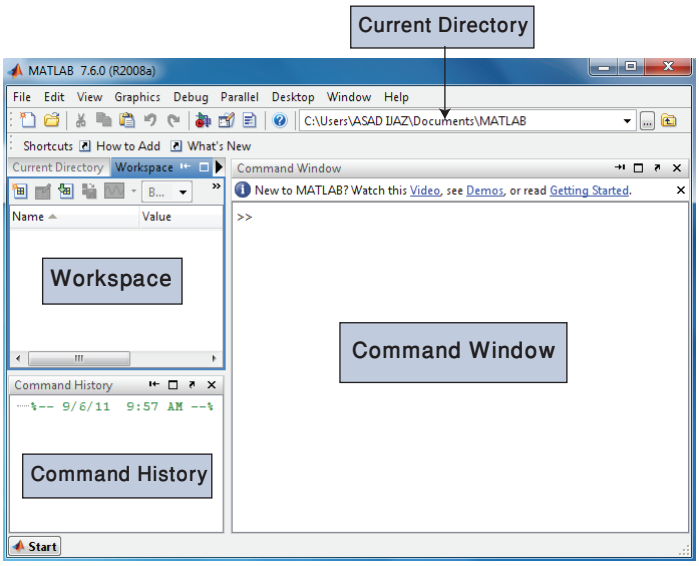


Figure 1: The default Matlab desktop.

The accessible major tools are,

1. The Command Window
2. The Current Directory
3. The workspace
4. The Edit/Debug Window

The function of these tools will be discussed here.

1.2 The Command Window

The *Command Window* is on the right of default Matlab desktop, whereby a user can enter commands at the command prompt (≫), and they are executed on the spot.

Suppose we want to calculate the area of a circle with a radius of 2.5 m. This can be done by typing,

```
≫ area = pi * 2.5 ^ 2

area =

    19.6350
```

Matlab calculates the answer as the command is entered, and stores the answer in a variable. As in the above example the variable is *area*. The contents of the variable are displayed in the Command Window as shown in Figure (2).

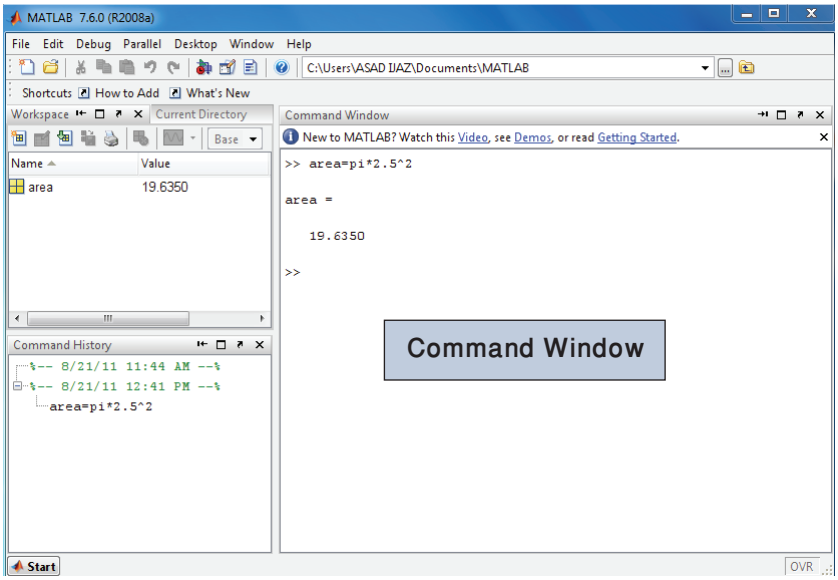


Figure 2: The Command Window.

1.3 The Current Directory

The current directory shows the contents of the working directory. Whenever you have to execute any function file, it is necessary that the executed file must be in the current directory of Matlab. For example if you are going to type a **lsqcurvefit(@straight,[20 1],n,f)** command in the Command Window, the function file **straight** is in the current directory to execute it without error.

1.4 The Edit/Debug Window

An *Edit Window* is created when you create a new M-file. You can create a new M-file with the **File→New→M-file** selection from the desktop menu or with the selection of **Start→Desktop Tools→Editor**.

The *Edit Window* is a programming text editor with the Matlab languages features displayed in different colours. In an M-file, comments appear in green, variables and numbers in black, complete character strings in magenta, incomplete character string in red and language keywords in blue.

An *Edit Window* showing an M-file named *calc_ area.m* is depicted in Figure (3). This file calculates the area of a circle and displays the result. After saving an M-file, it can be executed by typing its name in the Command Window. For the M-file of Figure (3), the output is,

≫ **calc_ area**

The area of the circle is 19.635

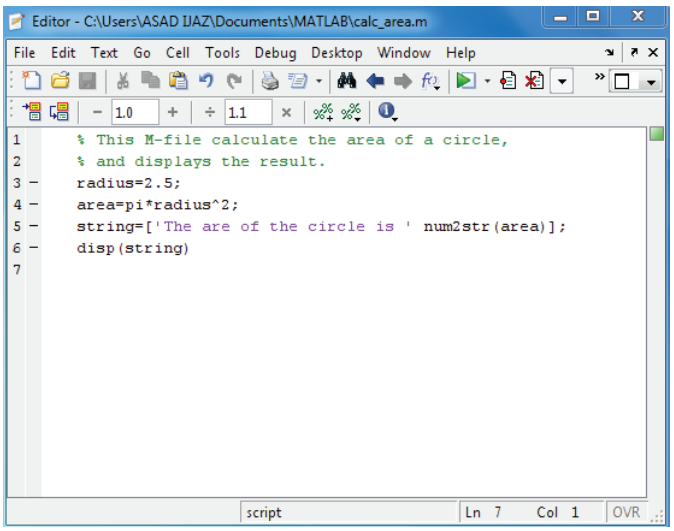


Figure 3: The Matlab Editor as an independent window.

1.5 The Workspace

A workspace contains all the variables and arrays that can be used by the Matlab when a command, M-file or function is executing. All commands executed in the Command Window have a common workspace, a way by which they can all share variables.

A statement such as,
≫ **x=10;**
creates a variable named *x*, stores the value 10 in it and saves it in the Workspace characterized as a part of the computer memory.

The contents of any variable or array may be accessed by typing the appropriate name in the Command Window, e.g., the contents of *string* can be found as,

≫ **string**

string =

The area of the circle is 19.635

Figure (4) shows the workspace displaying the string content and the Array Editor. The Array Editor can be accessed by clicking a variable in the workspace Browser, allows to change the values of a variable.

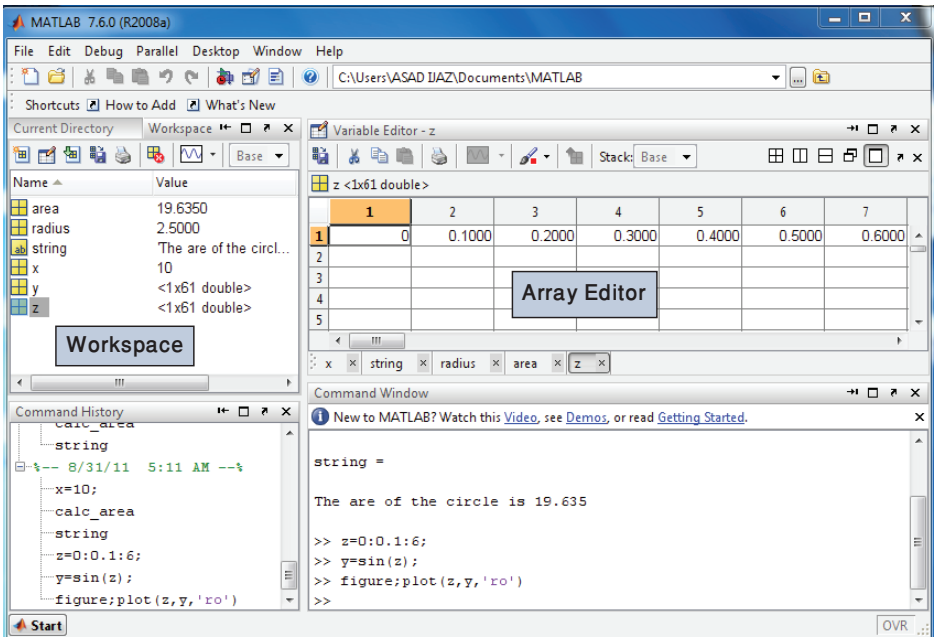


Figure 4: The Workspace and the Array Editor.

A variable can be deleted from the Workspace using the *clear* command, and it takes the form as,

≫ **clear x**

In order to remove a number of variables in one go, you need to follow the command,

≫ **clear var1 var2 var3 .....**

where, **var1**, **var2** and **var3** are the names of variables to be deleted. The command **clear all** or **clear** deletes all the variables from the current workspace.

2 Basic mathematics

2.1 Arithmetics

It is important to learn how some basic elementary operations are performed in Matlab and what would be the proper way. In this section we will practice some basic arithmetic operations being executed in Matlab.
Numbers can be entered directly in the *Command Window* using the keyboard. For example, if we type 5 and press **Enter**, the screen shows,

$\gg$ **5**
ans =
5

The Matlab displays the variable name **ans** to the resulting quantity if you do not assign a specific name. Suppose we desire to assign the name **x** to the value 5, so we type,

$\gg$ **x = 5**
x =
5

If we terminate the command with the semi-colon,

$\gg$ **x = 5;**
the output 5 will not be displayed.

Suppose we wish to add two values to form $z1$ as the sum, we write,

$\gg$ **z1 = 5 + 3**
What do you see? Indeed, 8.
Multiplication of scalars is done by placing $(*)$ between the numbers, hence we can write,

$\gg$ **z2 = 5 * 3;**

Similarly division with scalars can be achieved using the forward slash (/),

$\gg$ **z3 = 5 / 3;**

However, making use of variables, a better approach is,

$\gg$ **z4 = x / y;**

Now take the square of a number, for example, by typing,

$\gg$ **z5 = 5 ^ 2**

and verify if you get the correct answer.
The numbers entered so far have all been very simple. What if we have very small or very large values, for that we make use of the exponential form. For example, a microwave frequency $f = 15$ GHz (1 GHz= 10^9 Hz) can be entered as,

$\gg$ **f = 15e9**

$f =$
$1.5000e + 010$

The square root of a scalar quantity can be written as,

$\gg$ **z6 = sqrt(x);**

2.2 Creating vectors and matrices

Matlab is centred around the manipulation of matrices. In fact, the word Matlab is acronym for MATrix LABoratory. Let us generate a simple list of numbers, called a *vector*. This vector comprises all even numbers greater than or equal to 2 and less than 20. We call this vector evenlist.

$\gg$ **evenlist = [2 4 6 8 10 12 14 16 18]**

The vector will be displayed, all entries ranging from 2 to 18 lined up in a row. We have just created a *row vector*. A compact way of creating evenlist would be to write,

$\gg$ **evenlist2 = 2:2:18**

with the first 2 representing the first element, the second 2 representing the step size and the 18 showing the last element of the vector. Indeed evenlist and evenlist2 are equal. At some later stage, if we want to recall what the vector evenlist2 is, we just retype the label.

$\gg$ **evenlist2**

How do we make a *column vector*, where all the entries are arranged vertically instead of horizontally? We can use the semicolon as a delimiter among rows.

$\gg$ **evenlist3 = [2; 4; 6; 8; 10; 12; 14; 16; 18]**

Alternatively, we can avoid keying in the numerical entries by taking the transpose of the row vector evenlist2.

$\gg$ **evenlist4 = evenlist3';**

2.3 Matrix arithmetic

Another simple example illustrates matrix multiplication in MATLAB.

$\gg$ **a = [2 4 6; 1 3 5; 7 9 11];**

The above operation generates a matrix of order 3×3.

$$\begin{pmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \\ 7 & 9 & 11 \end{pmatrix}. \tag{1}$$

If we desire to know the size of matrix **a**, we type,

$\gg$ **size(a)**

Type in the command,

$\gg$ **a ^ 2;**

This performs the product of the matrices as **a a** and the resulting matrix is,

$$\begin{pmatrix} 50 & 74 & 98 \\ 40 & 58 & 76 \\ 100 & 154 & 208 \end{pmatrix}. \tag{2}$$

Now perform the following operation on the same matrix,

$\gg$ **a. ^ 2**

This operation just takes the square of *each* entry as shown,

$$\begin{pmatrix} 4 & 19 & 36 \\ 1 & 9 & 25 \\ 49 & 81 & 121 \end{pmatrix}. \tag{3}$$

By typing **a'** in the command window, we get the transpose of the generated matrix **a** as,

$$\begin{pmatrix} 2 & 1 & 7 \\ 4 & 3 & 9 \\ 6 & 5 & 11 \end{pmatrix}. \tag{4}$$

The inverse of a matrix **a** is determined by the simple command,

$\gg$ **c = inv(a);**

To understand how MATLAB interprets the forward slash **/** and the backward slash \, we try some simple commands.

By typing,

$\gg$ **a=4/2**

We obtain the answer 2, the result of a division operation. That is, the number on the left hand side of the forward slash is being divided by the number on the right hand side. On the other hand, if we type,

$\gg$ **b=4\2**

The answer is 0.5, which clearly indicates that the number on the right hand side is being divided by the number on the left hand side.

2.4 Extracting Elements from Matrices

Now suppose, we wish to select some entries from a generated row or column vector or from matrices. Define the row vector,

$\gg$ **a = [2 4 6 8 10 12 14 16 18 20]**

We want to extract the entries from column 3 to 7. We write,

$\gg$ **b = a(3:7);**

The colon operator will extract the entries from column 3 to 7, thus giving us the output,

$\gg$ **b = [6 8 10 12 14]**

Similar procedure can be repeated with a column vector.

We define a matrix by,

$\gg$ **a = [5 8 9; 2 4 6; 1 3 5]**

$$\begin{pmatrix} 5 & 8 & 9 \\ 2 & 4 & 6 \\ 1 & 3 & 5 \end{pmatrix}. \tag{5}$$

The order of the matrix a is 3×3. Starting from the easiest concept of selecting one single entry from a matrix, we will move on to select the whole row or column of that matrix. Suppose we want to select the entry 4 in the above matrix. We look at the position of that specified entry inside the matrix. The element is located in the second row and second column of the matrix.

$\gg$ **b =a(2,2)**

This command takes the value from second row and second column of **a** and saves it in **b**. The displayed output is 4.

To select a complete row or column of any matrix we have to use the colon operator, ":" which means that *all* entries of that specified row or column will be selected. For example,

$\gg$ **a(2,:)**

displays all the entries of the second row of the matrix, and

$\gg$ **a(:,2)**

displays all the entries in the second column of the matrix **a**.

If we write **d=a(:,:)** in the command window, we get the complete matrix again, i.e., we have selected all the rows and all the columns.

3 Data manipulation

3.1 Introduction to 'for' Loops

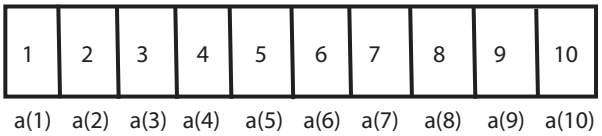
For loops are very powerful when we want to continuously update elements of any vector. The typical structure of a **for** loop is

for *(condition)*
statements
end

Now define a row vector,

$\gg$ **a = [1 2 3 4 5 6 7 8 9 10]**

A row vector stores information in the following way,



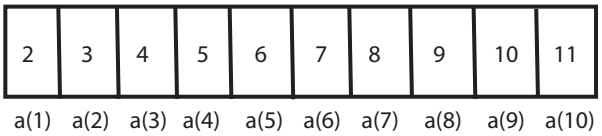
If we now want to add $+1$ to all the elements of **a**, we can write a for loop,

$\gg$ **for k = 1:10**

$\gg$ **a(k) = a(k) + 1;**

$\gg$ **end**

Matlab now updates every element of **a** by $+1$. The new array will look like,



Note that a **for** statement needs an accompanying **end** statement marking the end of the statements that are being executed.

4 Exercises

Q 1. Suppose that $u = 1$ and $v = 3$. Evaluate the following expressions using Matlab.

(a) $\frac{4u}{3v}$

(b) $\frac{2v^{-2}}{(u+v)^2}$

(c) $\frac{v^3}{v^3-u^3}$

(d) $\frac{4}{3}\pi v^2$

(e) $\sqrt{13-v^2}$

(f) $-1.6\times 10^{-19}\times \pi^3$

Q 2. The distance traveled by a ball falling in the air is given by the equation

$$x = x_o + v_o t + \frac{1}{2}at^2.$$

Use Matlab to evaluate the position of the at time $t = 5\,\text{s}$ if $x_o = 10\,\text{m}$, $v_o = 15\,\text{m/s}$ and $a = 9.81\,\text{m/s}^2$.

Q 3. Assume that a, b, c and d are defined as follows,

$$a = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} -1 & 2 \\ 0 & 1 \end{pmatrix}, \quad c = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \quad d = 5.$$

What is the result of each of the following mathematical operations?

(a) $a + b$

(b) $a * b$

(c) $a * c$

(d) $a. * d$

(e) $a. * b$

(f) $a^{-1}b^{-1}$

Q 4. Answer the following questions for the array shown below. Show command typed into Matlab.

$$c = \begin{pmatrix} 1.1 & -3.2 & 3.4 & 0.6 \\ 0.6 & 1.1 & -0.6 & 3.1 \\ 1.3 & 0.6 & 5.5 & 0.0 \end{pmatrix}.$$

(a) What is the size of c?

(b) What is the value of $c(2,3)$?

(c) List the subscripts of all elements containing the value 0.6.

(d) How do you access all the elements of second row?

(e) Determine $c(1:2, 2:\text{end})$, $c(4:\text{end})$, $c(1:2, 2:4)$, and $c([1\ 3], 2)$. Briefly describe your results.

Q 5. A certain physical system has three unknown variables, x_1, x_2 and x_3, that can be described by the following linear equations:

$$\begin{aligned} x_1 + 2x_2 - x_3 &= -8 \\ -x_1 + x_2 + 3x_3 &= 7 \\ 3x_1 + 2x_2 + x_3 &= 4 \end{aligned}$$

Solve for the three unknowns using matrix algebra. (Hint: Use $Ax = b$).

Q 6. Use **for** loop to calculate squares, square roots and cube roots of all integers between 1 and 10.

5 Graphs and plotting

Graphs are extremely important in experimental physics. There are three important uses of graphs [1].

- First, with the help of graphs we can easily determine slopes and intercepts.
- Second, they act as visual aids indicating how one quantity varies when the other is changed, often revealing subtle relationships. These visual patterns also tell us if there exist conditions under which simple (linear) relationships break down or sudden transitions take place.
- Third, graphs help compare theoretical predictions with experimentally observed data.

It is customary to plot the independent variable (the "cause") on the horizontal axis and the dependent variable (the "effect") on the vertical axis. In Matlab, the data for the independent and dependent variables are typed in as vectors.

5.1 Introduction to plotting

Matlab has the capability to generate plots of many types, including linear plots, line plots, logarithmic plots on both scales, logarithmic plots on one scale, bar graphs and three-dimensional plots. In order to achieve a two-dimensional plot, we need to create two vectors containing the x and y values and use the **plot** function.

Let's consider an example of a stretched string fixed at one end to a rigid support, is strung over a pulley and a weight of 1.2 kg is attached at the other end. The sting can be set under vibrations using a mechanical oscillator (woofer) connected to the signal generator.

The relation of angular velocity (ω) with the wave vector (k) is called the dispersion relation and given by,

$$\omega(k) = \sqrt{\frac{T}{\mu}} \times k = n\sqrt{\frac{T}{\mu}} \times \frac{\pi}{L}, \tag{6}$$

where L (1.5 m) is the effective length between the wedge and woofer and n is the number of mode. The experiment is to sweep the signal generator frequency, and note down frequencies at which the resonance occurs. The data obtained is given in Table (1),

Resonance mode	1	2	3	4	5
Frequency (Hz)	20.82	41.82	61.32	82.32	104.1

Table 1: Relationship between frequency and resonance modes.

Now let's plot frequency as a function of number of mode, keeping frequency on the vertical and number of mode on the horizontal axis.

The best way to perform the task is to create a row vector containing all values of the independent variable at which points are to be determined. This can be done by typing the number of modes, then enclosing the brackets to form a vector.

≫ **n=[1 2 3 4 5];**

We can also write it as,

≫ **n=[1:5];**

Next, we define a row vector for the dependent variable.

≫ **f=[20.82 41.82 61.32 82.32 104.1];**

The general command for a linear plot is,

≫ **plot(n,f)**

where n is the horizontal variable and f is the vertical variable. Note that no semicolon at the end of the command.
If we will enter another **plot** command, the previous plot is omitted and the new one is displayed. So in order to avoid this inconvenience, we will prefer to open a new plot window each time we plot a function.

A new plot window can be opened using the **figure** command,

≫ **figure;plot(n,f)**

Usually the desired commands for most of figures are **xlabel**, **ylabel** and **title**, their names are self explanatory. First we enter the **xlabel** command,

≫ **xlabel('Resonance mode (n)')**

Note that the desired quantity is enclosed by **('')**. Next, we type the **ylabel** command,

≫ **ylabel('Frequency (Hz))')**

Same as the title command is entered as,

≫ **title('Dispersion relation for a bare string')**

The resulting graph is a solid line joining the individual points as depicted in Figure (5a). The data points can be highlighted using the symbol 'o', ,'+', '•' etc. The colour can also be adjusted. Suppose we want to plot a solid red line with data points circles, for that we need to type the command,

≫ **figure;plot(n,f,'r-o')**

Furthermore, if it is required to display the data points only, suppressing the line that connects between these points, we type,

≫ **figure;plot(n,f,'ro')**

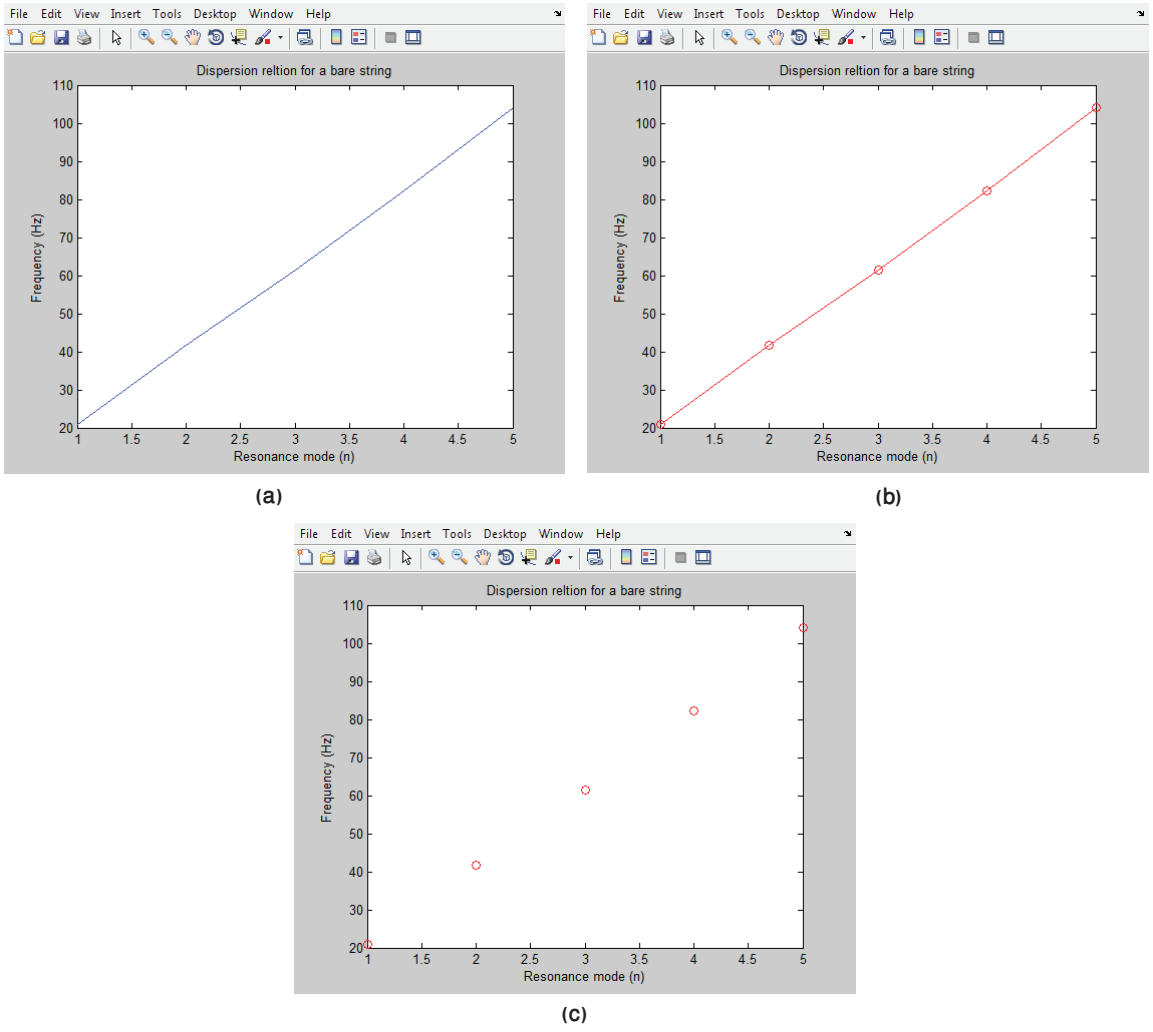


Figure 5: (a) Output from **figure; plot(n,f);** a solid jagged line connects the data points; (b) output from **figure; plot(n,f,'r-o');** a solid red line connects the data points that are now highlighted; (c) output from **figure; plot(n,f,'ro');** showing just the data points.

This latter plot, shown in Fig (1c) in fact, represents a more justifiable picture of the exper-

imental data. This is because the lines drawn in (a) and (b) represent more than what the data warrants: the lines show that the frequency and the resonance mode have some kind of jagged relationship, something that is highly likely. A more reasonable prediction is that the relationship is a straight line. In the next section, we will discuss how to draw one such line, using the procedure of *least squares curve fitting*.

5.2 Multiple Plots

It is also possible to plot multiple curves on the same figure. This is a highly useful feature as you will soon realize. Suppose we wish to plot graphs of the following functions,

$$y = 2\cos(x), \quad \text{and} \tag{7}$$
$$y = 2\sin(x). \tag{8}$$

for the range of $x \in [0, 4\pi]$.

Let's define the input vectors,

$\gg$ **x=0:0.1:4*pi;**

$\gg$ **y=2*cos(x);**

$\gg$ **y1=2*sin(x);**

The cosine function can be plotted by using the command,

$\gg$ **figure; plot(x,y,'r-d')**

where we have used "**r-d**" for red line with data points in diamond symbol. The output is shown in the Figure (6 a).

The sine function is plotted as,

$\gg$ **figure; plot(x,y1,'b-*')**

Here we have used "**b-***" to plot blue line with data set in star shape as depicted in Figure (6 b)

To plot both the graphs simultaneously, one on top of another, we will use the **hold on** command,

$\gg$ **figure; plot(x,y,'r-d')**

$\gg$ **hold on**

$\gg$ **plot(x,y1,'b-*')**

The result is shown is Figure (6 c).

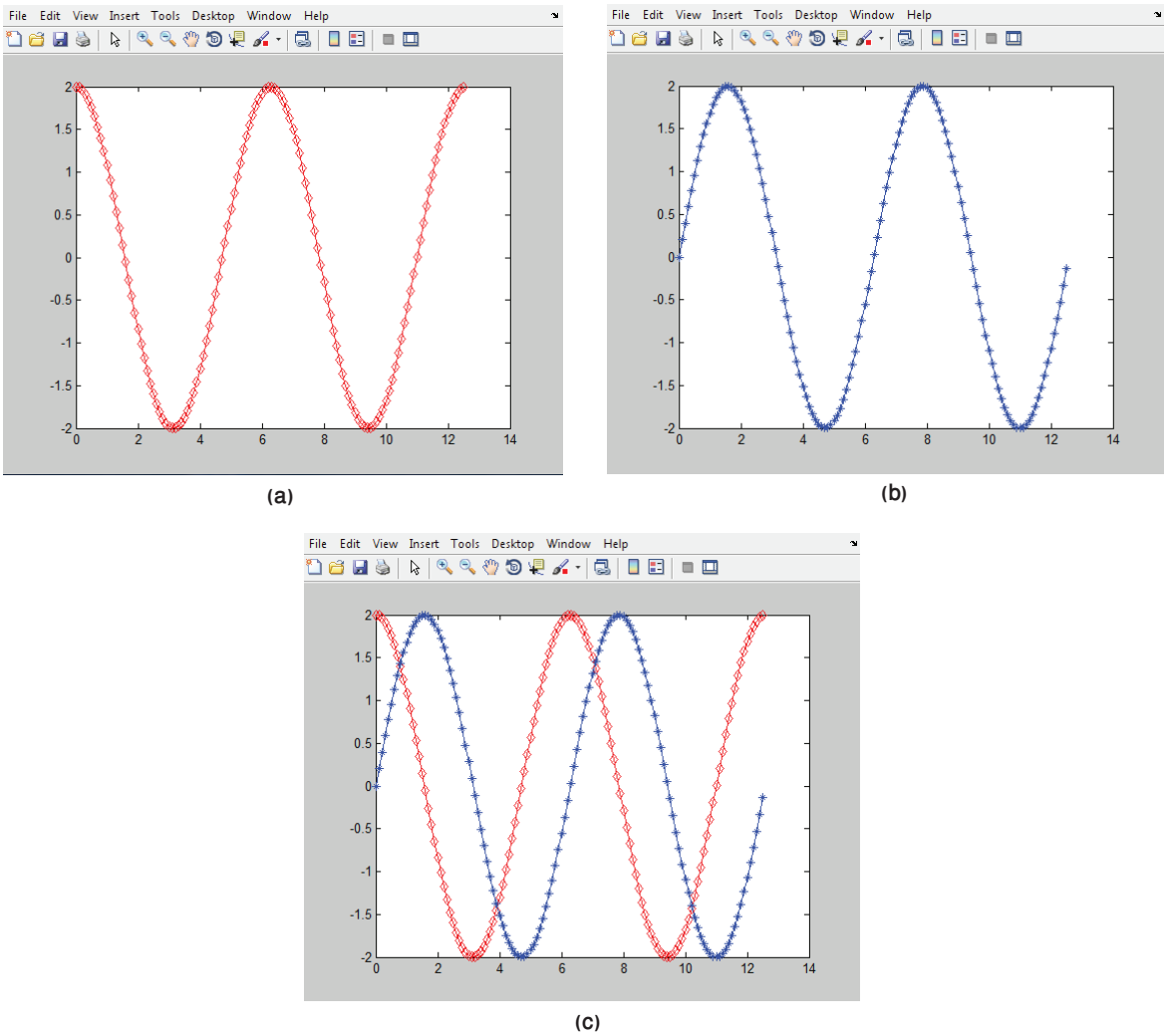


Figure 6: (a) Graph of $y = 2cos(x)$, (b) $y1 = 2sin(x)$, and (c) Overlaid both plots.

5.3 Resolution of the Graph

Suppose we have a sine curve,

$$\sin(t), \tag{9}$$

sampled at interval of 1 s for a duration of 10 s, it means there are eleven data points contained within the sampled duration. We know from experience that a plot of the sine function should be smooth, unlike the irregular curve shown. Why is this discrepancy? The reason is that we have not sampled enough points. Decreasing the sampling interval to 0.1 s and hence, increasing the number of samples to 101, we recover a smooth sine curve, shown in Figure (7 b). These plots have been made using the following commands.

$\gg$ **t1=0:1:10;**

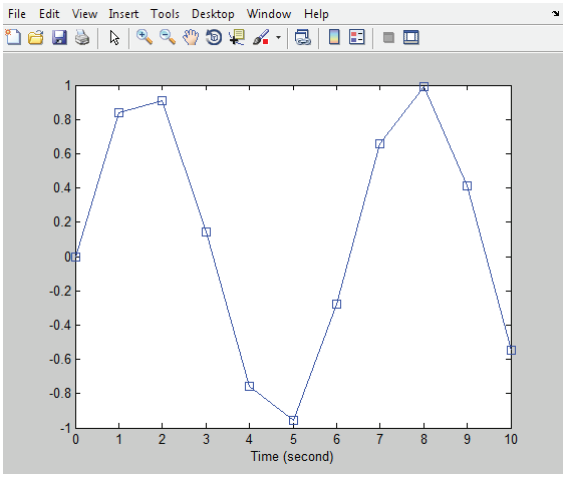
$\gg$ **x1=sin(t1)**;

$\gg$ **figure; plot(t1,x1,'b-s');** (*for the subfigure (a)*)

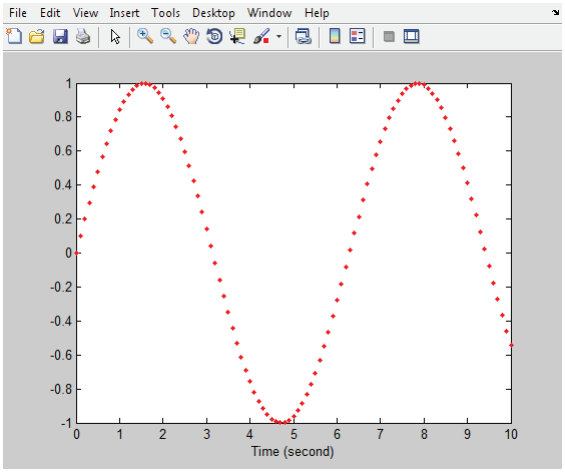
$\gg$ **t2=0:.1:10;**

≫ **x2=sin(t2)**;

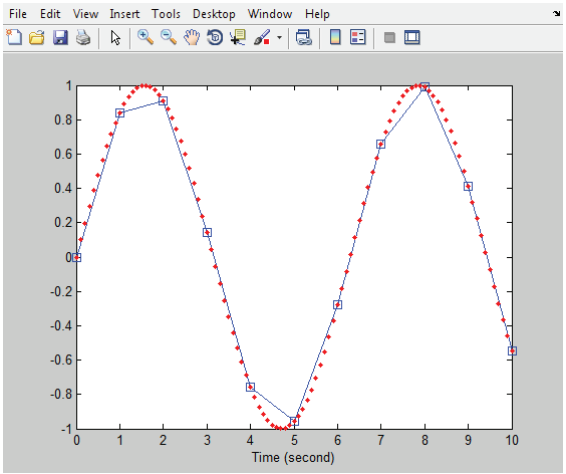
≫ **figure; plot(t2,x2,'r.');** (*for the subfigure (b)*)



(a)



(b)



(c)

Figure 7: (a) Lower resolution graph, (b)higher resolution graph, (b) the two graphs have been drawn on top of each other.

These plots can be overlaid one on top of each other using the command,

≫ **figure; plot(t1,x1,'b-s'); hold on; plot(t2,x2,'r.');**

The result is shown in Figure (7 c).

6 Exercises

Q 7. The deflection of a cantilever beam is the distance its end moves in responce to a force applied at the end. The following table gives the deflection **x** that was produced in a particular beam by the given applied force **f**. Find a functional relationship between **x** and **f** and plot the graph.

Force f (Pounds)	0	100	200	300	400	500	600	700	800
Deflection x (inches)	0	0.09	0.18	0.28	0.37	0.46	0.55	0.65	0.74

Table 2: An experiment to measure force and deflection in cantilever beam.

Q 8. The following Matlab statements plot the function $y = 2e^{-0.2x}$ for the range $0 \leq x \leq 10$.

Q 9. Draw a graph of the function,

$$y = \frac{\sin t}{t} \tag{10}$$

for $0 \leq t \leq 10$.

Q 10. For the values of x, $0 \leq 2\pi$, show by drawing a graph that,

$$\sin^2 x + \cos^2 x = 1. \tag{11}$$

Q 11. Draw a graph of the function,

$$z = \exp(-0.5t)\cos(20t - 6) \tag{12}$$

for $0 \leq t \leq 8$.

Q 12. Draw a graph of the function,

$$y = -x\exp(-x) \tag{13}$$

for $0 \leq x \leq 10$.

Q 13. Suppose the relationship between the dependent variable y and the independent variable x is given by,

$$y = ae^{-x} + b \tag{14}$$

where a and b are constants. Sketch a curve of y versus x using arbitrary values of a and b. Is it possible to obtain a straight line that represents this functional relationship?

7 Curve Fitting

Linear relationships occur naturally in numerous natural instances and that is why they have become the scientist's favourite. Linear relationships [1] are direct manifestations of direct proportionality. If the variables x and y are directly proportional ($x \propto y$), an equal increase in x always results in an equal increase in y . Be it the extension of a spring when loaded with masses, the acceleration of an object as it experiences a force or the magnetic field that winds around a current carrying conductor, linear relationships are ubiquitous. When these linear functions are drawn on paper (or on the computer screen), they become straight lines.

[1] $F = ma$
(Newton's law)
$F = -kx$
(Hooke's law)
$B = \mu_0 NI$
(Ampere's law)

7.1 Least Squares Curve Fitting of Linear Data

Suppose we have a data set of Table (1). The graph between frequency and the resonance mode is shown in Figure (8). Can we draw a straight line through these points, not necessarily touching them? What could be the significance of such a line?

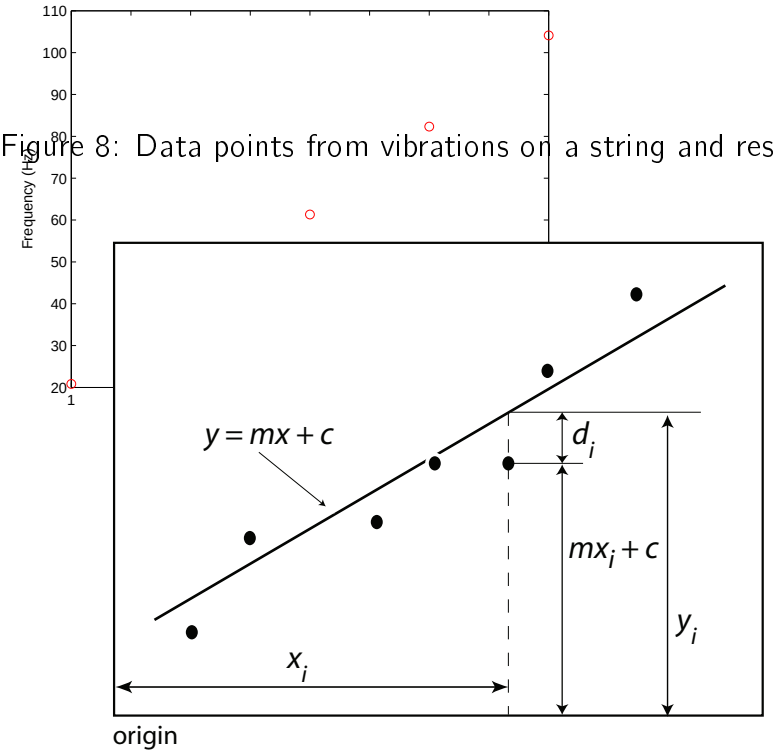


Figure 8: Data points from vibrations on a string and resonance.

Figure 9: Setting for the least squares best fit.

Consider Figure 9 where a straight line has been drawn around a set of experimentally measured data points (x_i, y_i). In this example we have $N = 7$ pairs of measurements. The line is represented by the equation,

$$y = mx + c \tag{15}$$

where m is the *slope* and c is the *intercept*. Of the many lines that can be drawn, this particular line has a special property that we now investigate. If the reading along the abscissa (x axis) is x_i, the corresponding measurement along the ordinate (y axis) is y_i, but the line we have just drawn takes up the value, $mx_i + c$ instead, which in general, is different from y_i. This difference

$$d_i = y_i - mx_i - c \tag{16}$$

is called the *residual* or *deviation*. The special line we have drawn has the property that it minimizes the sum of the squares of the deviations,

$$S = \Sigma_{i=1}^{N} d_i^2 = \Sigma_{i=1}^{N} (y_i - mx_i - c)^2, \tag{17}$$

and hence the name *least squares curve fit*. If the d_i's are considered to be the errors, the least squares curve fit is the best fit in the sense that it minimizes the squares of the errors.

Now we will fit the data given in Table 1 to a least squares curve using Matlab. The input vectors are,

$\gg$ **n=[1 2 3 4 5];**

$\gg$ **f=[20.82 41.82 61.32 82.32 104.1];**

Our *best fit* will be of the form,

$$\mathbf{y=mx+c}, \tag{18}$$

where m is the slope and c is the intercept. The curve fitting procedure determines approximations to these parameters, m and c.

Instead we use the inbuilt Matlab command **lsqcurvefit**. We first make a new function file named **straight.m** that contains the fitting function. Follow the below mentioned steps to make a new function file, also called an "M-file".

1. From the **File** menu item, click **New** and **M-file**. A blank text editor opens.
2. Type in the following text in the editor window.

   function fout=straight(p,fin)
   fout=p(1)*fin+p(2);
   end

   and save the file in the current working directory as **straight.m**.

Let's parse this file, line by line. The first line starts with the label **function** indicating that this m-file is a function file, or in other words, this file contains the declaration of a function named **straight** that can be called from inside the command window.

The function **straight** takes in two vector arguments, **p** and **fin**. The former is a vector containing the unknown parameters. In our case **p** has two elements **p(1)** and **p(2)** which are respectively m and c. The latter **fin** is the input vector, in our case this is the vector containing the frequency values. The second line defines the fitting function; this is the Matlab way of writing Equation (18). Finally, the m-file ends with the statement **end**.

Once the fitting function has been defined, we can find the least squares curve [2] using the command,

$\gg$ **lsqcurvefit(@straight,[21 1],n,f)**

[2] lsqcurvefit requires the optimization toolbox

The first argument in small bracket references the function *straight* we have just created. The second argument is a vector containing initial guesses of the unknown parameters. It will be easier for Matlab if we could make intelligent guesses of these parameters. The last two arguments, **n** and **f** are the x-axis and y-axis variable names.

After you enter the above mentioned command in the Command Window, Matlab returns the values of the parameters, $m = 20.7060$, $c = -0.0420$. The initial data points and the higher resolution curve fit are then plotted using the set of commands given below.

≫ **n1=1:0.1:5;** (*high sampling rate for plotting the fitted curve*)

≫ **cfit=20.7060*n1-0.0420;**

≫ **figure; plot(n,f,'ro'); hold on;**

≫ **plot(n1,cfit);**

The results are shown in Figure **??**. The command,

≫ **[x,resnorm]=lsqcurvefit(@sinusoid,[21 1],n,f)**

also returns the sum of the squares of the residuals,

$$\sum_{i}^{N} d_i^2 \tag{19}$$

which is a measure of the goodness of the fit.

7.2 Exercise

Q 14. Suppose a rocket is fired into the space from rest. The distance covered (in miles) by the rocket and the height gained (in miles) is given in the table below,

Distance (miles)	0	1	2	3	4	5	6	7	8	9	10	11	12
Height (miles)	0	0.53	0.75	0.92	1.07	1.20	1.31	1.41	1.51	1.60	1.69	1.77	1.85

Table 3: Height of a rocket versus Distance.

Plot the graph of distance against height and perform curve fitting using the equation,

$$y = a\sqrt{bx}. \tag{20}$$

Q 15. An object covers a distance d in time t. A measurement of d with respect to t produces the set of values given in Table 4 [5].

Plot the distance with respect to t. Then plot with respect to t^2. If the object was initially at rest, calculate the acceleration. Use curve fitting.

t (s)	1	2	3	4	5	6	7	8
d (m)	0.20	0.43	0.81	1.57	2.43	3.81	4.80	6.39

Table 4: Measurements of distance as a function of time.

Q 16. Biomedical instruments are used to measure many quantities such as body temperature, blood oxygen level, heart rate and so on. Engineers developing these devices often need a response curve that describes how fast the instrument can make measurements. The response voltage v can be described by one of these equations,

$$\begin{aligned} v(t) &= a_1 + a_2 e^{-3t/T} \\ v(t) &= a_1 + a_2 e^{-3t/T} + a_3 t e^{-3t/T} \end{aligned} \tag{21}$$

where t is the time and T is an unknown constant. The data given in Table 5 gives the voltage v of a certain device as a function of time. Which of the above functions is a better description of the data [7]?

t (s)	0	0.3	0.8	1.1	1.6	2.3	3
v (V)	0	0.6	1.28	1.5	1.7	1.75	1.8

Table 5: Response of a biomedical instrument switched on at time $t = 0$.

Q 17. In an RC series circuit, a parallel plate capacitor having capacitance C charges through a resistor R. During the charging of capacitor, charge Q starts to accumulate on the plates of the capacitor. The expression for growth of charge V is given by,

$$V = V_o(1 - \exp(-t/\tau)) \tag{22}$$

where the time constant $\tau = RC$. Fit the given data in Table 7 to the equation for the voltage increase and find the value of τ.

t (s)	0	3	6	9	12	15	18	21	24	27	30
V (V)	0	6.55	10	13	14.5	15	16	16.2	16.3	16.5	16.55

Table 6: Charging pattern for a capacitor in an RC circuit.

Q 18. When a constant voltage was applied to a certain motor initially at rest, its rotational speed $S(t)$ versus time was measured. The table given below shows the values of speed against time.

Time (s)	1	2	3	4	5	6	7	8	10
Speed (rpm)	1210	1866	2301	2564	2724	2881	2879	2915	3010

Table 7: Motor speed when it is given a push.

Try to fit the given data with the function given below. Calculate the constants b and c.

$$S(t) = b(1 - e^{ct}) \tag{23}$$

Q 19. The yield stress of many metals, σ_y, varies with the size of the grains. Often, the relationship between the grain size, d, and the yield stress is modelled with the Hall-Petch equation,

$$\sigma_y = \sigma_0 + kd^{-1/2} \tag{24}$$

d (mm)	0.006	0.011	0.017	0.025	0.039	0.060	0.081	0.105
σ_y (MPa)	334	276	249	235	216	197	194	182

Table 8: Measurement of flow velocity.

Determine the constants and best fit the data points.

8 Appendix

A Linear Relationships

Figure 10(a) is a reproduction of the data points as given in Table (9), taken from the Millikan's experiment [4]. However, in this graph we have also drawn two straight lines. Why straight lines? The straight lines we have drawn in Figure 10(a) represent a kind of

Frequency (Hz)	0.5488	0.6914	0.7413	0.8219	0.9597	1.1834
Stopping voltage (Volts)	- 2.1	- 1.524	- 1.367	- .9478	- .3718	.3720

Table 9: Millikan's readings for the stopping voltage as a function of frequency of the incident length; results extracted from [4].

interpolation. In the real experiment, we measure the variables, (x_i, y_i). In our case these are frequency and stopping voltage. In a set of measurements, we have six pairs of data points $(x_1, y_1), (x_2, y_2), \ldots, (x_6, y_6)$. What if we want to determine the stopping voltage for a frequency that was not used by Millikan? We could either repeat his experiment with a light source with the desired frequency or estimate using *available* data. In the latter case, we draw a straight line around the available measurements (x_i, y_i). This line negotiates data points not available to the experimenter.

But what straight line do we actually draw? This is a matter of choice. For example, we have drawn two lines in the Figure. The light colored line takes the first and the last data points as reference and connects these points; whereas the dark colored line connects the mean (or the centre of gravity of the data) to the end point. Both lines are different and at the outset, are equally suitable for defining the linear relationship between the variables of interest.

Let's briefly digress to see how we plotted, say, the red line. To plot a line, we need an equation for the line. Given two points (x_1, y_1) and (x_6, y_6), a straight line through these will be given by,

$$\frac{y - y_1}{y_6 - y_1} = \frac{x - x_1}{x_6 - x_1}, \tag{25}$$

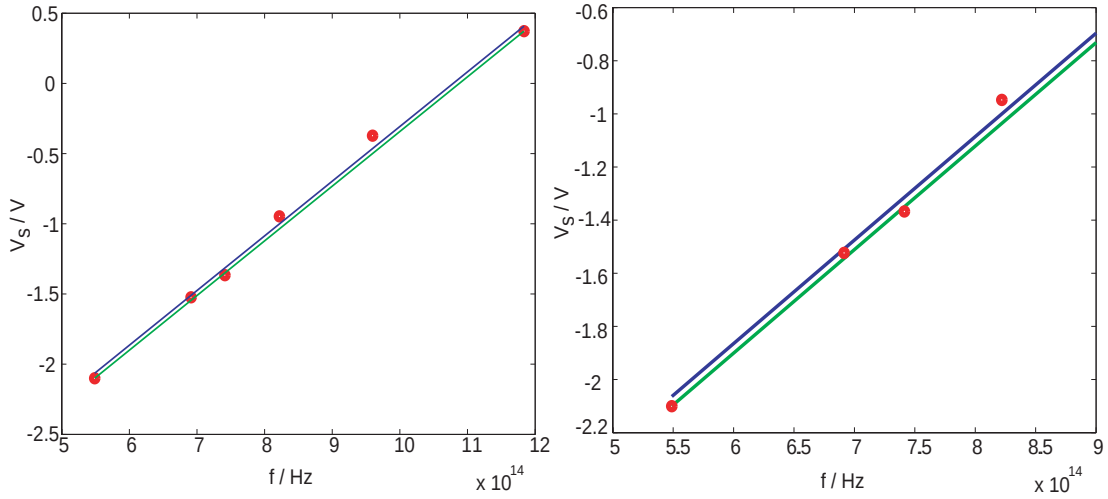


Figure 10: (a) Data points from Millikan's experiment [4] with two possible lines defining the functional relationship between f and V_s; (b) magnified region from the graph (a), closely showing the data points and the straight lines.

and in our case $(x_1, y_1) = (0.5488 \times 10^{15}, -2.1)$ and $(x_6, y_6) = (1.1834 \times 10^{15}, 0.3720)$. (These numbers have been taken from the row vectors **f** and **vs**.) After some basic arithmetic (also done in Matlab) we arrive at the following equation for the red line,

$$y = 3.895 \times 10^{-15} x - 4.2375, \tag{26}$$

where in our particular case y is the stopping voltage v_s and x is the frequency f. Similarly, the equation for the blue line was computed by first calculating the means of the x and y values. The resulting equation is,

$$y = 3.895 \times 10^{-15} x - 4.2018, \tag{27}$$

yielding a line parallel to the first, but displaced upwards. Figure 10(b) shows a close-up of (a), revealing that these lines do not actually touch a majority of the data points, they just graze within that region.

The graph has been plotted by using the following set of commands.

≫ **line1=3.895e-15*f-4.2375;**

≫ **line2=3.895e-15*f-4.2018;**

≫ **figure; plot(f,vs,'ro',f,line1,'g-',f,line2,'b-')**;

Q 20. Why do we minimize the sum squares of the residuals $\sum_{i=1}^{N} d_i^2$ instead of the sum of the residuals $\sum_{i=1}^{N} d_i$?

There is an algorithmic procedure for deriving the equation for the least squares fit. The goal is to find the parameters m and c that minimize the quantity S. The minimum of S can be determined from elementary calculus. Take the derivative of S, first with respect to m and then with respect to c and put the derivatives equal to zero,

$$\frac{\partial S}{\partial m} = -2\sum_{i=1}^{N} x_i(y_i - mx_i - c) = 0 \tag{28}$$

$$\frac{\partial S}{\partial c} = -2\sum_{i=1}^{N}(y_i - mx_i - c) = 0. \tag{29}$$

Rearranging Equation 29, we obtain,

$$\begin{aligned}
\sum_{i=1}^{N}(y_i - mx_i - c) &= 0 \\
\sum_{i=1}^{N} y_i - m\sum_{i=1}^{N} x_i - cN &= 0 \\
\implies c &= \frac{\sum_i^N y_i - m\sum_i^N x_i}{N}, \qquad (30)\\
\text{where } \sum_i^N &\equiv \sum_{i=1}^{N}. \qquad (31)
\end{aligned}$$

The expression for c is inserted into Equation 28 and after some algebraic manipulation,

$$\begin{aligned}
\sum_{i}^{N} x_i(y_i - mx_i - c) &= 0 \\
\sum_{i}^{N}(x_i y_i) - m\sum_{i}^{N} x_i^2 - c\sum_{i}^{N} x_i &= 0 \\
\sum_{i}^{N}(x_i y_i) - m\sum_{i}^{N} x_i^2 - \left[\frac{\sum_i^N y_i - m\sum_i^N x_i}{N}\right]\sum_{i}^{N} x_i &= 0 \\
\sum_{i}^{N}(x_i y_i) - m\sum_{i}^{N} x_i^2 - \frac{1}{N}\left(\sum_{i}^{N} x_i\right)\left(\sum_{i}^{N} y_i\right) + \frac{m}{N}\left(\sum_{i}^{N} x_i\right)^2 &= 0, \qquad (32)
\end{aligned}$$

the following expression for m pops out,

$$m = \frac{\sum_i^N (x_i y_i) - \frac{1}{N}\left(\sum_i^N x_i\right)\left(\sum_i^N y_i\right)}{\left(\sum_i^N x_i^2\right) - \frac{\left(\sum_i^N x_i\right)^2}{N}}. \tag{33}$$

This cumbersome looking expression can be simplified by noticing that,

$$\frac{\sum_i^N x_i}{N} = \overline{x} \tag{34}$$

is the mean of x_i and

$$\frac{\sum_i^N y_i}{N} = \overline{y} \tag{35}$$

is the mean of y_i, yielding,

$$m = \frac{\sum_i^N (x_i y_i) - N\overline{x}\,\overline{y}}{\sum_i^N x_i^2 - N\overline{x}^2}. \tag{36}$$

Furthermore, we can also make use of the following simplifications for the numerator and denominator of the above expression,

$$\begin{aligned}
\sum_i^N (x_i y_i) - N\overline{x}\,\overline{y} &= \sum_i^N (x_i y_i) - (\sum_i^N y_i)\overline{x} \\
&= \sum_i^N y_i (x_i - \overline{x}), \quad \text{and} \qquad (37) \\
\sum_i^N x_i^2 - N\overline{x}^2 &= \sum_i^N x_i^2 + N\overline{x}^2 - 2N\overline{x}^2 \\
&= \sum_i^N x_i^2 + N\overline{x}^2 - 2\overline{x}\sum_i^N x_i \\
&= \sum_i^N x_i^2 + \sum_i^N \overline{x}^2 - 2\overline{x}\sum_i^N x_i \\
&= \sum_i^N (x_i^2 + \overline{x}^2 - 2\overline{x}x_i) \\
&= \sum_i^N (x_i - \overline{x})^2. \qquad (38)
\end{aligned}$$

This tedious but fruitful exercise yields the following compact expression for the slope of the least squares curve fit,

$$m = \frac{\sum_i^N y_i (x_i - \overline{x})}{\sum_i^N (x_i - \overline{x})^2}. \tag{39}$$

Substituting the expression for m back into (31) we can determine the intercept,

$$c = \overline{y} - m\overline{x}. \tag{40}$$

Now we use Matlab to find the least squares curve for Millikan's experimental data. The commands that generate the best fit line are given below.

$\gg$ **numerator=sum(vs.*(f-mean(f)));**

$\gg$ **denominator=sum((f-mean(f)).^ 2);**

$\gg$ **m=numerator/denominator;**

$\gg$ **c=mean(vs)-m*mean(f);**

The values are $m = 3.9588 \times 10^{-15}$ V/Hz and $c = -4.2535$ V. We can now easily plot the least squares fit, shown in Figure 11.

$\gg$ **line3=m*f+c;**

$\gg$ **figure; plot(f,vs,'ro',f,line3,'g-')**

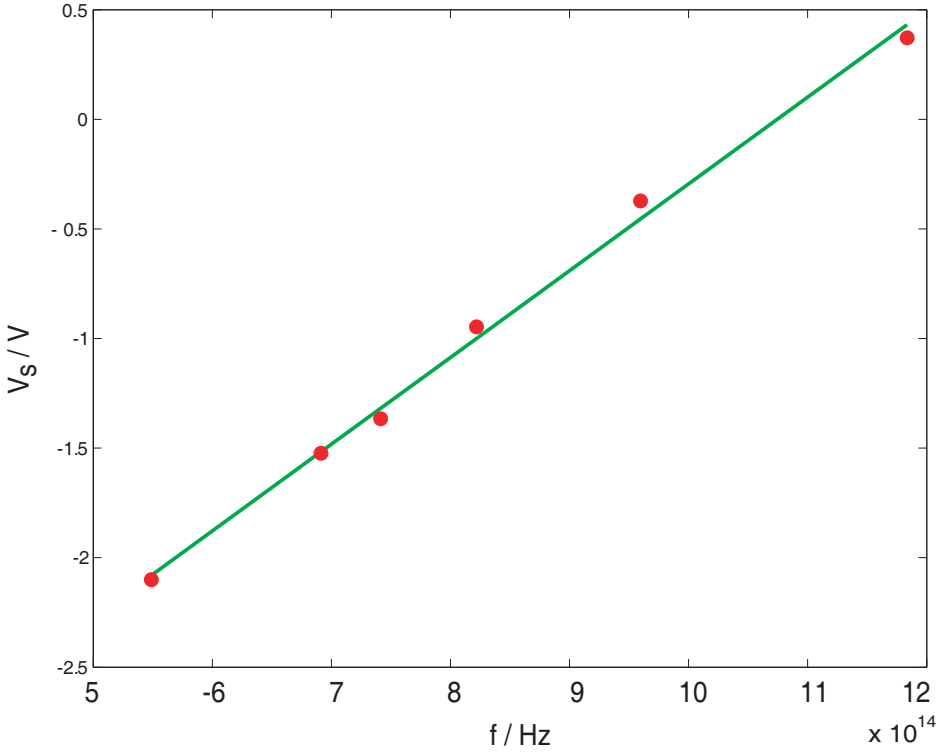


Figure 11: Data points for Millikan's experiment and the least squares curve fit.

The straight line, in fact, has real physical value as well. For example, according to Einstein's interpretation of the photoelectric effect, light is carried in the form of small packets called photons. Corresponding to the frequency f, the photon carries an energy hf, where h is Planck's constant. As light is shone on the metal surface, a fraction of the energy called the work function $W < hf$ is absorbed by the metal surface. The ejected electron carries the energy difference $hf - W$ appearing as its kinetic energy. As the voltage V_s is made more and more negative, the number of electrons reaching electrode Q diminishes with only the more energetic electrons being able to overcome the opposing voltage. At the stopping voltage, the maximum kinetic energy equals the voltage barrier. Given a potential of V_s, the corresponding potential energy is eV_s, e being the charge of the electron. This description allows us to write the following equation,

$$\begin{aligned} eV_s &= hf - W \\ V_s &= (\frac{h}{e})f - (\frac{W}{e}). \end{aligned} \tag{41}$$

Comparing this with the least squares fitted equation 15, we immediately recognize that the slope m is in fact an estimate of h/e and the intercept c is an estimate of W/e. Using the slope and intercept from the best-fit and a value of $e = 1.6022 \times 10^{-19}$ C, the Planck constant calculates to $h = 6.342 \times 10^{-34}$ J s and the work function to $W = 6.814 \times 10^{-19}$ J or 4.2535 eV.

References

[1] G.L. Squires, Practical Physics, (Cambridge University Press, 1999).

[2] http://nsbri.tamu.edu/HumanPhysSpace/focus6/student2.html.

[3] http://zirka.dp.ua/Instructions.htm.

[4] R. Millikan, "A direct photoelectric determination of Planck's "h"", Phys. Rev. **7** 355 (1917).

[5] D. W. Preston, "The Art of Experimental Physics", (Cambridge University Press, 1991).

[6] www.maths.dundee.ac.uk/ ftp/na-reports/MatlabNotes.pdf.

[7] W. J. Palm, "Introduction to Matlab 6 for Engineers", (McGraw-Hill Companies, 2000).

[8] The Math Works, "The Language of Technical Computing", (The Math Works, 2000).

[9] Amos Gilat and Vish Subramaniam, "Numerical Methods for Engineers and Scientists", (Wiley Companies, 2007).