MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Physics Department

Physics 8.01T Fall Term 2004

Experiment 01: Introduction to *DataStudio*

Purpose of the Experiment:
DataStudio, a computer program, and the associated SW750 interface are tools that we will use in many of the desktop experiments in 8.01T. This experiment has two goals:

1. To learn how to set up the interface so you can use the computer to make measurements. In this case you will use the motion sensor to measure position and velocity.
2. To learn how to analyze the measurements you have made, both by plotting your data on graphs so you can see how they look and by using the fitting capabilities of *DataStudio* to find a mathematical function that describes your data and determine the function parameters that give the "best" description of the data.

You will have to make an initial intellectual investment to learn how to use these tools; the reward is the ability to quickly and easily make more sophisticated measurements and to be able to interpret them more richly.

There is a web page with examples of how to use *DataStudio* to do most of the things you will need for 8.01T. You will probably find that consulting it will lead to more efficient use of your time. Here is the table of contents of that web page.

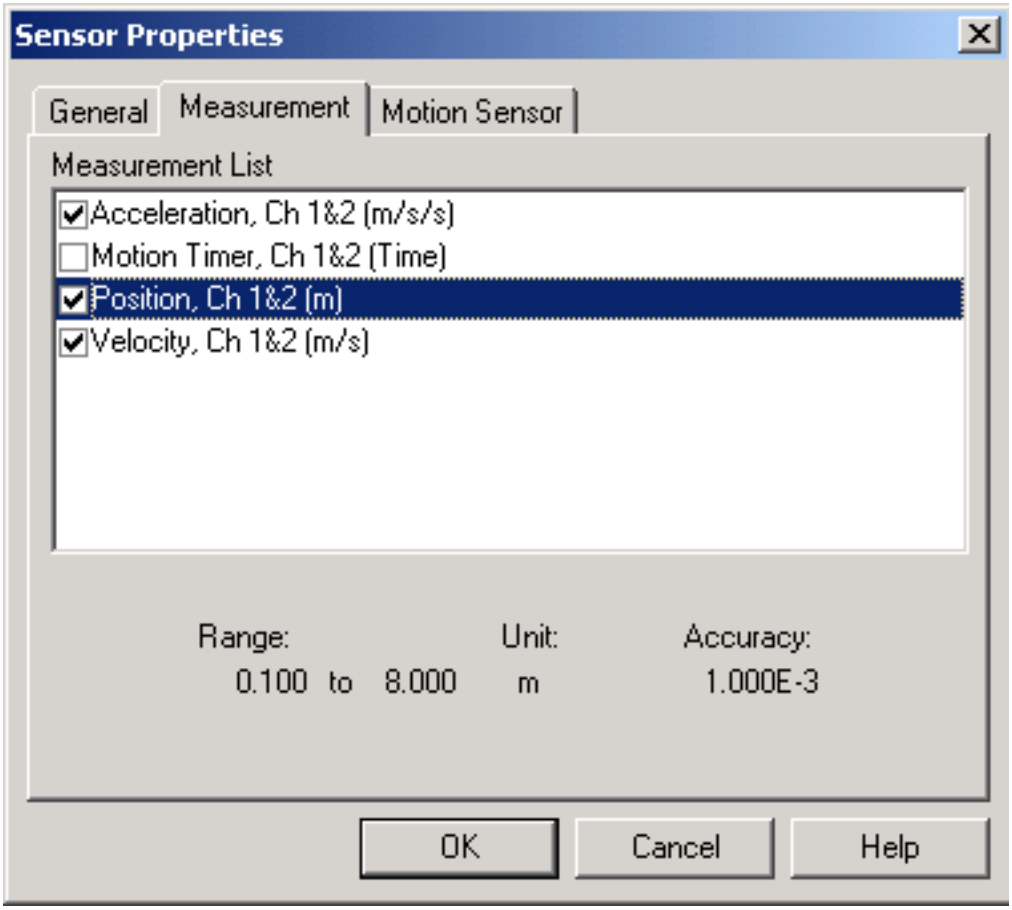
For making measurements today, the sections on the "Motion Sensor" and "Automatic Measurement" may be helpful. As this is your first experiment, these notes will give more detailed suggestions than you will find for subsequent experiments. Here are the steps to start your experiment.

If your SW750 interface box has a SCSI connection (unfortunately, most of them do), make sure the SW750 is turned on and plugged into the computer **before** you boot the computer; if it was not, you will have to reboot after you connect the SW750 and turn it on.

Start *DataStudio* and choose the "Create Experiment" option.

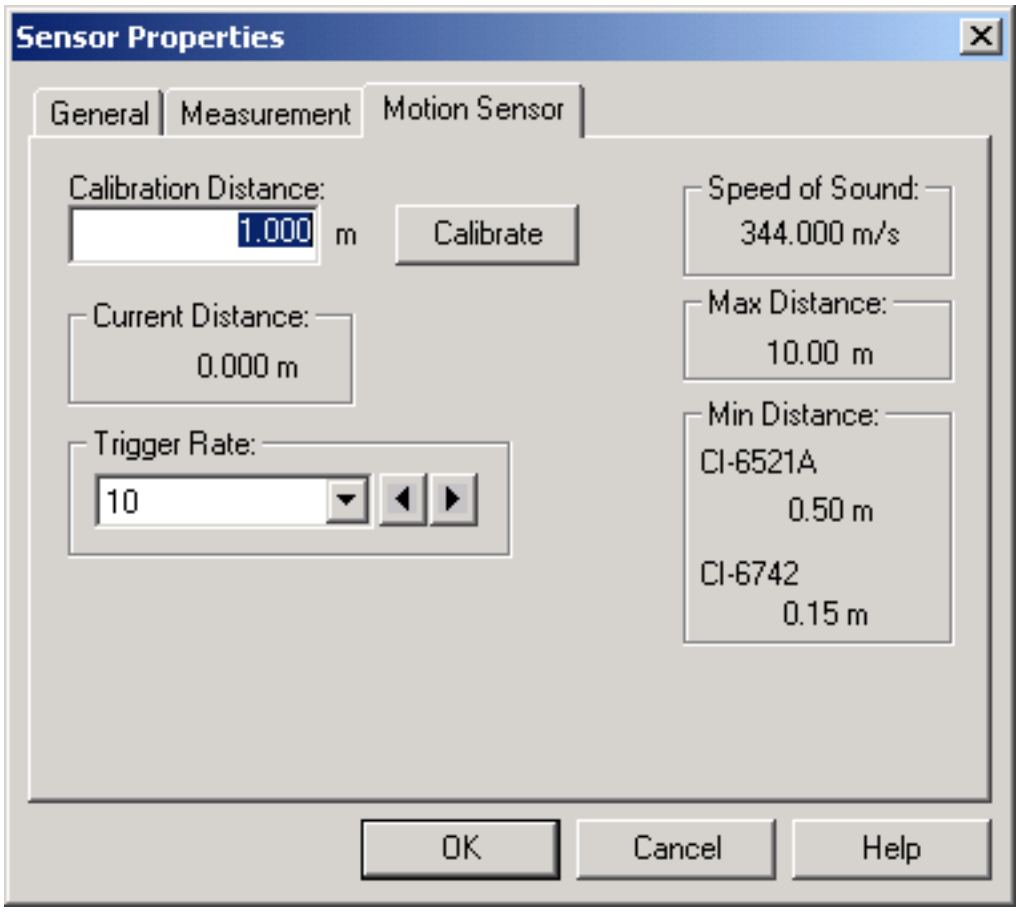
Set the motion sensor switch to the "narrow" range, connect it to the SW150 (yellow plug in #1, black plug in #2), and in the "Experiment Setup" window drag the motion sensor icon from the sensors list to terminals 1&2 of the picture of the SW750.

Double-click the motion sensor icon where it connects to the SW750. A window will open; click the "Measurement" tab to get a window that looks like this.



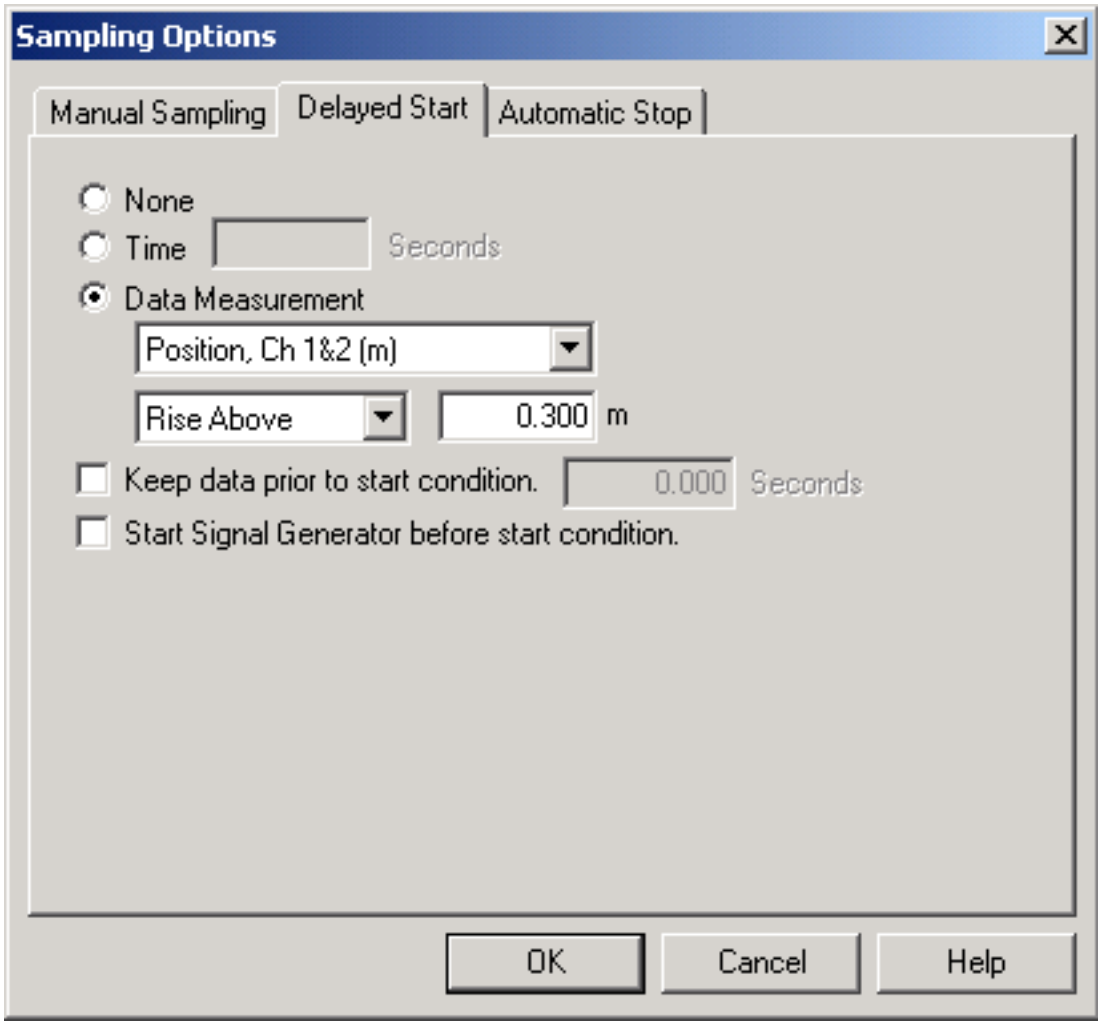
Only the Position and Velocity boxes should be checked in the Measurement List. (Click a checked box to uncheck it.) If you click on the measurement, in this case the "Position" line, to highlight it, you can see the range, units, and accuracy of the sensor.

Click the “Motion Sensor” tab to get this window.



Set the Trigger Rate to 20. You should hear clicks coming from the sensor, and if you stick your hand in front of it, the distance to your hand should appear in the Current Distance box. This indicates the sensor is working. You do not need to calibrate it for this experiment, but if you want you may move your hand around or put other objects in front of the sensor to see what it does. Place the sensor on the metal track pointing towards the stop with a black button at the end. The sensor should be at an even 10 cm mark on the ruler and between 70 and 100 cm from the stop and it should “see” the stop and report the distance to it. Then click “OK” and the window will close.

Your experiment will require moving your hand in a controlled way between even 10 cm marks that are 20 cm and 50 cm from the sensor, a 30 cm peak-to-peak motion. Click the Options button in the Experiment Setup window to open a window that allows you to choose the conditions for starting and stopping the measurement. First click the “Manual Sampling” tab and make sure that none of the boxes are checked. Then click the “Delayed Start” tab, choose the “Data Measurement” radio button, and set the start condition to ”Fall Below” 0.55 m. (See the figure on the next page.) Finally, click the “Automatic Stop” tab, the “Time” radio button, and type 10 in the Seconds box.



With this setup, after clicking the "Start" button on the main menu you can approach the sensor with your hand from greater than 0.55 m away, *DataStudio* will start to measure the distance to your hand 20 times/second when you cross the 0.55 m point, and will continue to measure it for 10 s.

When the measurement stops, you will see entries labeled "Run #1" under the Position and Velocity entries in the Data window at the left of your main *DataStudio* window. Now you should plot the results on two graphs.

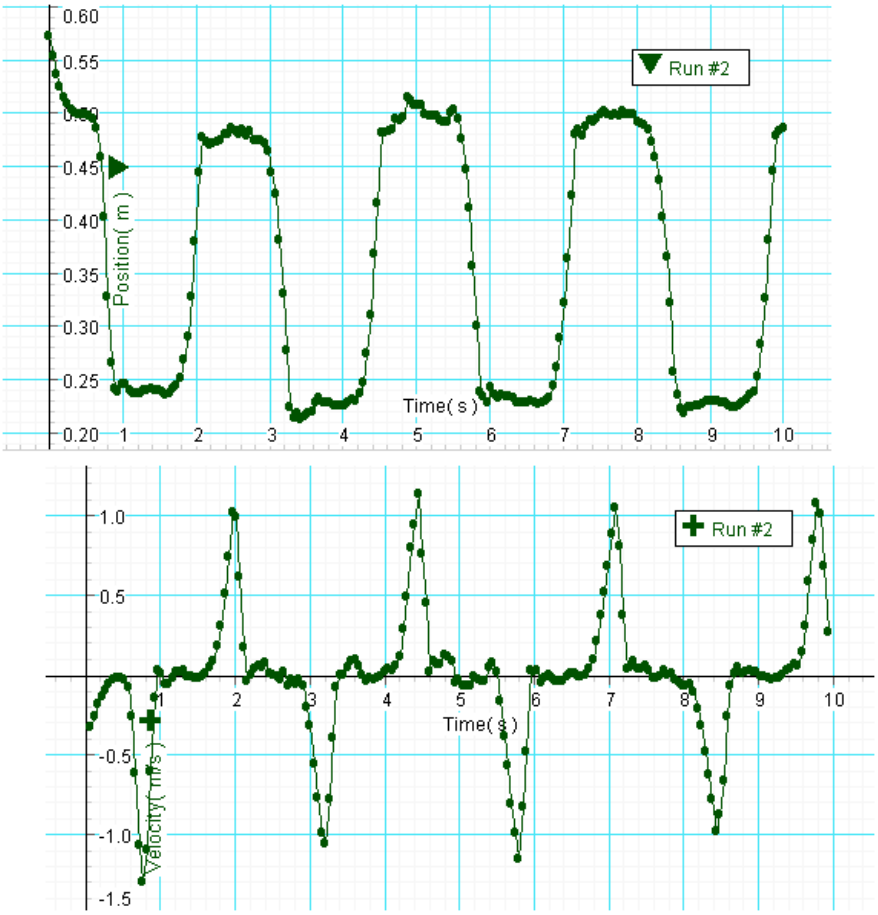
To do this, double-click the Graph entry in the Displays window, which is below the Data window. A window will open asking what data you want to plot on the graph. Click on "Position, Ch 1&2(m)" and your hand position will be plotted as a function of time on the graph. (If you want to change the color used to plot the data, you can double-click the Run #1 in the Data window; a large "Data Properties" window will open, and you can choose the color under the Appearance tab.) Double-click the Graph icon again and make another graph on which you can plot the velocity of your hand.

You can get the same results by dragging "Position, Ch 1&2(m)" from "Data" window onto the Graph entry in the "Displays" window and then doing the same with the "Velocity, Ch 1&2(m)" data.

Future measurements of position and velocity will also be plotted on these graphs. If the graphs get too crowded, you can choose which runs to plot from the pull-down menu under the Data button on the graph's toolbar. If there are data runs you do not want to keep, and there probably will be many of them, you can remove them completely (and finally) from the experiment by clicking them once in the Data window to select them, then use the Delete key.

What You Should Do for the Experiment

Here is a measurement that I made for the motion of my hand.



You can see that I was trying to make the position as a function of time be a square wave. The velocity graph gives some kind of indication how well I did.

In your experiment, try to move your hand so the position is

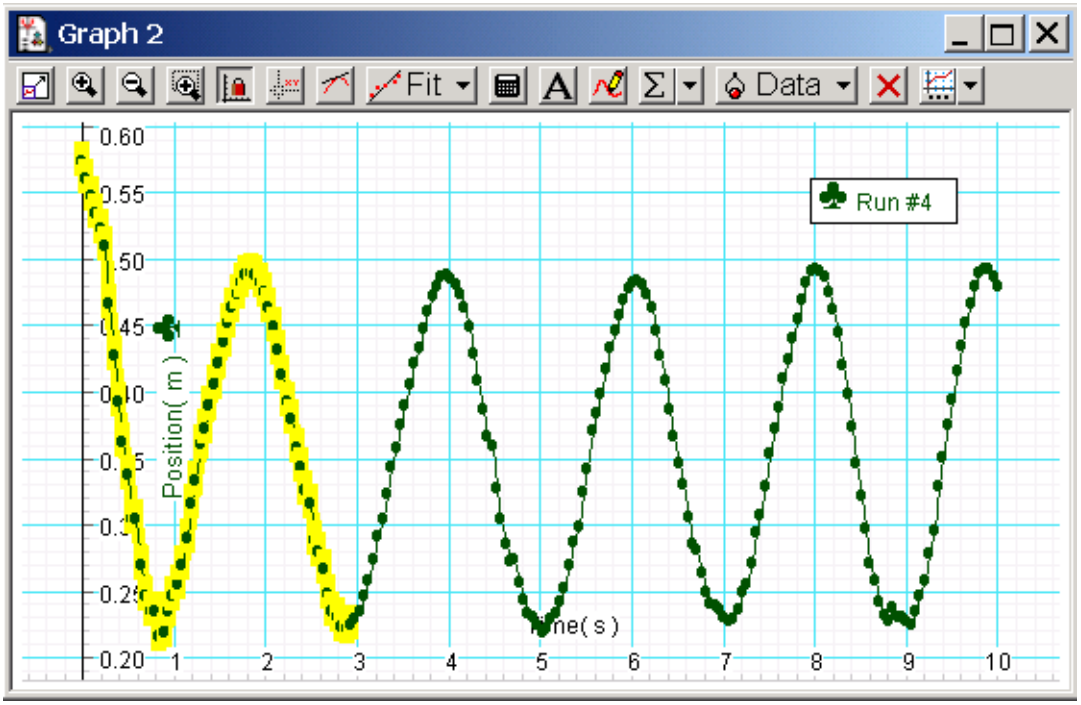
1. A square wave.
2. A triangle wave.
3. A sine (or cosine) wave.

Each of the three partners in the experiment should try these. I found the triangle wave was the most difficult and the sine wave was the easiest. Now we can have a contest to see which team can produce the best sine wave.

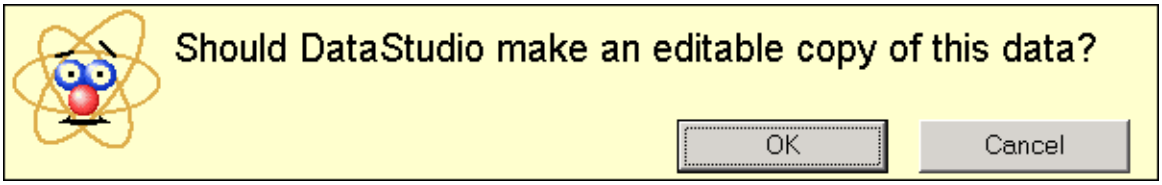
Quantitative Analysis of the Motion

Here you will use the fitting capability of *DataStudio* to obtain a quantitative measure of how close to a sine wave your hand motion was, and in the process gain some practice analysing data.

First, decide on your best sine wave measurement and plot it on a graph. On the graph, choose which five second interval of data you want to be your group's entry in the contest. (It is important for the analysis that you choose precisely a five second interval, as it will have 100 points.) Suppose you decide the data between 3.0 and 8.0 s are the ones you want. Use the computer's pointer to drag a rectangle around all of the data points for $t < 3.0$ s. The selected points will be outlined in yellow.

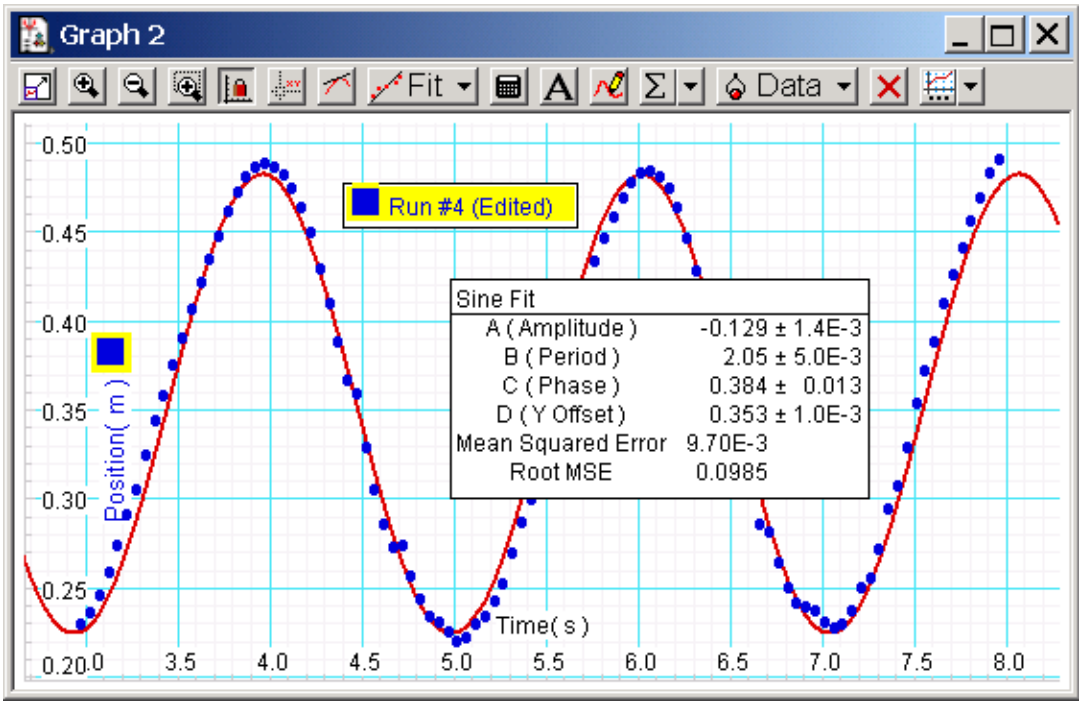


Click the button with the red X in the graph's toolbar. The following window will open.



Click OK and the graph will be replaced with one that has the selected data points removed. Next, select all of the data points with $t > 8.0$ s and click the red X. You will get a new graph that only has the data from 3.0 to 8.0 s. There will aslo be a new entry in the Data window for this reduced data set.

Then from the pull-down menu under the Fit tab on the graph's toolbar, choose "Sine Fit". You should get the result shown on the next page.



If you double-click the text box in the graph's window that displays the fit results, a "Curve Fit" window will open, giving more information about the fit. You will see that the data were fit to

$$y = A \sin\left(\frac{2\pi(x - C)}{B}\right) + D$$

where y is the position of the hand and x is the time. You can click the "Properties" button in it if you want to change the color for plotting the fit function.

The fit does not appear to be too bad. To obtain a number that expresses how good or bad it is, take the "Root MSE" number, divide by the magnitude of the amplitude "A", and divide that result by 10 (the reason for the division by 10, which is the square root of the number of data points you fit, is a bug in *DataStudio*; see Appendix II of these notes). The result for me was 0.076. This is the RMS average deviation from the "Sine Fit" curve to my measured hand position (as a fraction of the 13 cm amplitude), and shows my hand was within 8%, on average, of a perfect sine wave. How the fitting program chooses the parameters A, B, C, and D is discussed in Appendix II.

This deviation is about 10 times the accuracy that the motion sensor can measure distance, so it is almost entirely the result of my hand motion not being a precise sine wave—due to variations in the timing and amplitude of the motion.

When your group has obtained the analogous number for its best five seconds of "sine wave" hand motion, use the "Data Input Page" link on the experiments web page to report it, and the section instructor will be able to plot a distribution for the class.

Appendix I: Sine Fit Function

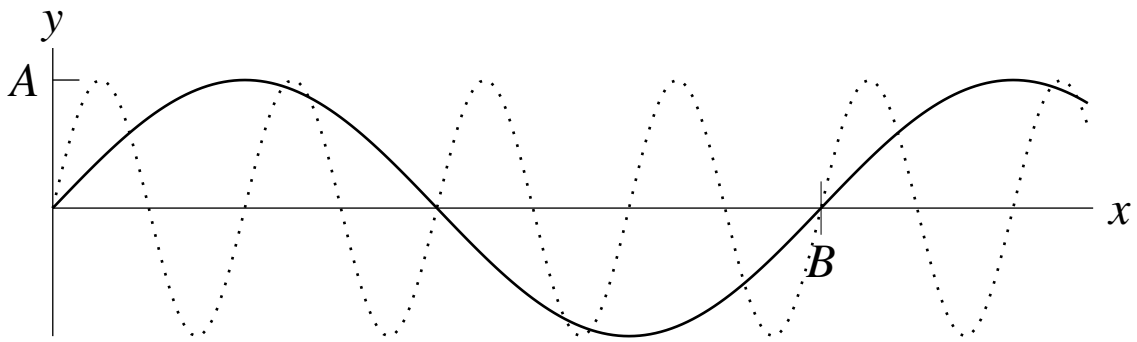
In analyzing measurements from this experiment you will fit the dependent variable y to the following function of the independent variable x:

$$y = A \sin\left(\frac{2\pi(x - C)}{B}\right) + D$$

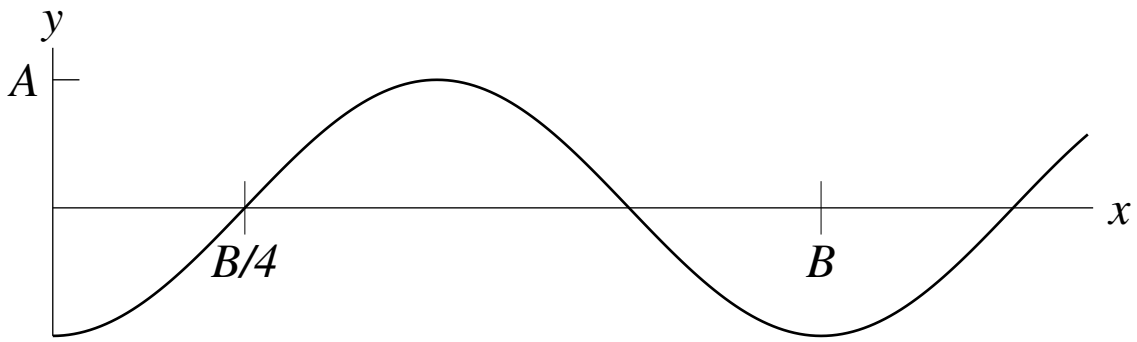
This appendix discusses this function and the four adjustable parameters A, B, C and D that it has. If you understand this function, there is no need for you to read this appendix. This function is a sine wave of amplitude A added to a constant (background) term D. The wavelength of the sine wave along the x direction is B and C is an offset that determines where the sine wave passes through zero. Commonly the angle $2\pi C/B = \phi$, which is measured in radians, is called a *phase* offset, and $2\pi/B = k$ is called the *wave vector*. Thus the function may also be written as

$$y = A \sin(kx - \phi) + D$$

The easiest way to see how the parameters change the function is to plot it on graphs for some different values of the parameters.



The above graph shows the function when $D = \phi = 0$. (The dotted curve is the same as the solid one, but with B four times smaller.) It is easy to imagine that $D \neq 0$ simply shifts the entire graph up or down and that changing A simply changes the vertical distance between the peaks and the valleys. To illustrate the effect of the phase, here is the same graph plotted, but with $C = B/4$ (or $\phi = \pi/2$).



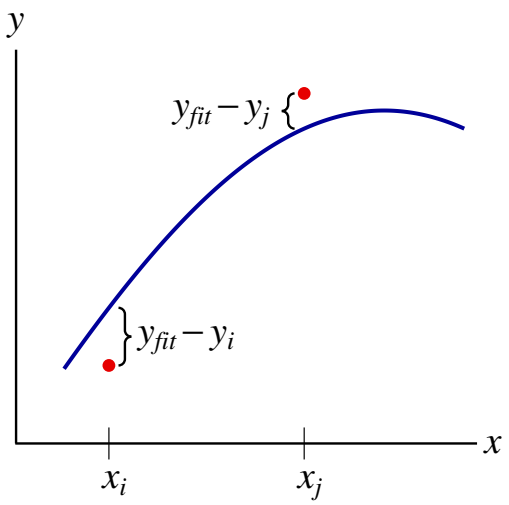
The effect is to shift the graph to the right by an amount $B/4$, since it is now necessary to have $x = B/4$ in order for the argument of the sine function to be zero (whereas it was zero at $x = 0$ when C was zero).

Appendix II: Fitting Data

In several of our experiments we will measure some dependent variable y as a function of an independent variable x. The independent variable will often be time, while the dependent variable may be position, velocity, force, or voltage. In order to interpret and analyse the results, it is useful to plot the data on a graph and to find a mathematical function that can represent them. This function will usually have several parameters and we would like to find the set of parameters that gives the "best" fit of the function to the data. There are two main reasons to carry out this fit.

The first is that we may have a theoretical model that we believe should describe the behavior measured; in that case the fit tests how well the model does explain the measured results and the fit will enable us to determine the parameters of the model. If we lack a theory for the measurements, then finding a function that represents the data may help us to develop a theory—the force between magnets in Experiment 03 will be an example. In either case, having a function that fits the measurements will allow us to interpolate between measured values of the variables and extrapolate to some degree outside the measured range.

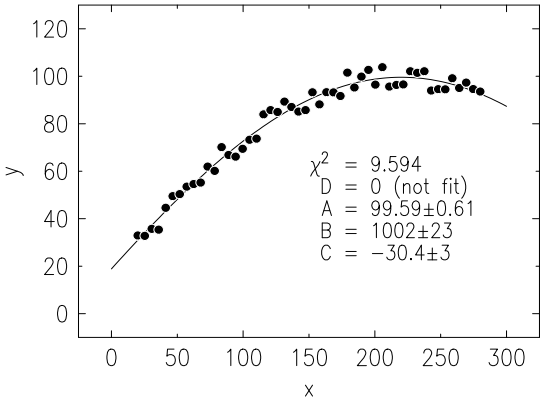
So how should we determine the "best" fit? The figure to the right plots a fitting function (the solid blue line) and two data points $y_i(x_i)$ and $y_j(x_j)$. As will normally be the case, the function does not pass through the points, but misses each of them by a deviation (also called a "residual") $y_{fit}(x_i) - y_i$. As the deviation may be positive or negative, the most common procedure is a "least squares fit" in which the parameters are chosen to minimize the "total square error" (the sum of all the residuals squared):



$$\Phi = \sum_i [y_{fit}(x_i) - y_i]^2$$

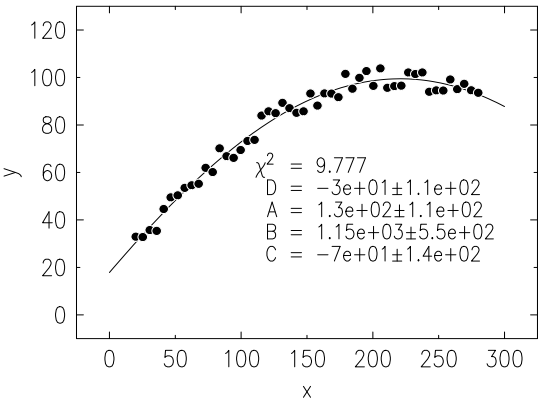
Now I would like to show a few examples; The mathematical details of how fitting programs work are of less importance here. It is more important that you understand what the fitting programs do. If properly written, they find the parameters that give the smallest value for Φ. All the examples here use the fitting function

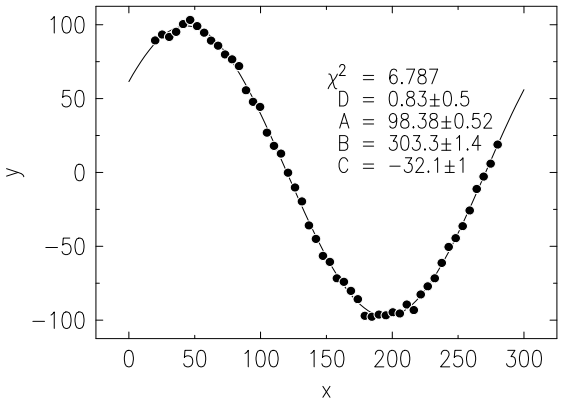
$$y_{fit}(x_i) = A \sin\left(\frac{2\pi(x_i - C)}{B}\right) + D\,.$$



The "data" were calculated from this function with $D = 0$, $A = 100$, $B = 1000$, and $C = -30$, unless otherwise stated. Random noise, of peak-to-peak amplitude 10 (mean squared value 8.3), was added to simulate measured data. On the left is a graph of 50 data points and the function that fits them best. The variable χ^2 is Φ divided by the number of points (50) less the number of parameters fitted (3), or 47. ($\chi^2 = 9.6$ compares well with the mean squared noise of 8.3.)

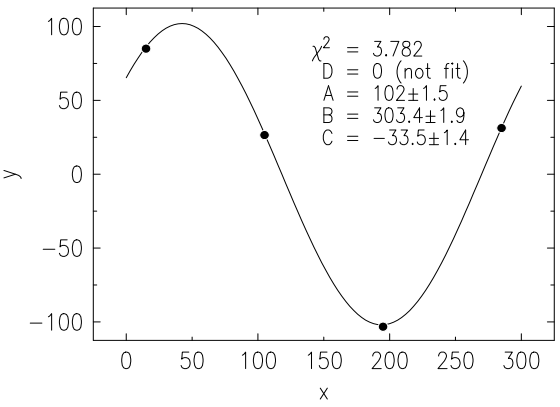
Here are two more fits.





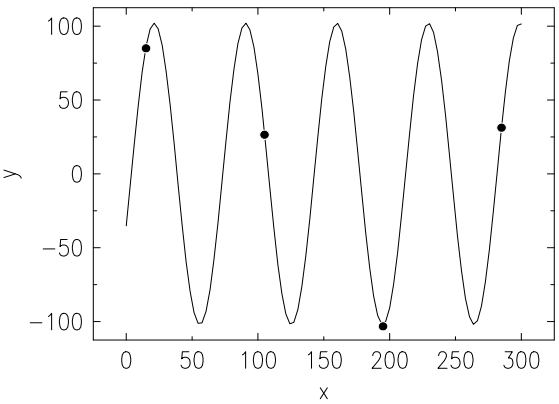
The graph on the left has the same data as the previous page, but the parameter D is allowed to vary in the fit. The uncertainties in the parameters are huge. That is because the data are somewhat noisy and they do not extend over a wide enough range in x (compared to B) for it to be apparent that $B = 1000 \pm 500$ is silly. The graph on the right has equally noisy data, but they extend over a wide enough range in x so there is enough wiggle in y to determine B. The lesson is that if you want to determine the parameters of your $y_{fit}(x)$ you must have data that extend over a wide enough range to test the form of the function. Over a small enough range, many functions look like a slightly curved line and you can't learn much from trying to fit them.

Now I want to show what may happen if you try to fit a small number of points.



The graph to the left contains four equally spaced data points from the same data set as the graph to the right above ($B = 300$). Even though there are only four points, the range in x is enough that the fit can determine A, B, and C reasonably well. However, the graph below shows a different fit to the same four points; this fit has the same χ^2 as the one to the left, but the parameters are $A = 102 \pm 1.5$, $B = 69.4 \pm 0.1$, and $C = 3.9 \pm 0.3$. Here the points are too widely spaced to rule out the extra wiggles when B is several times smaller.

This situation arises because $y_{fit}(x)$ is periodic in x and the points are equally spaced. However, it is just an extreme example of a problem that can occur more generally. The fitting program must always start with some initial values for the parameters. The algorithim will calculate Φ and vary the parameters to move "downhill" (in a direction of decreasing Φ in the multi-dimensional parameter space) searching for a minimum.



The problem is that Φ may have several minima in the parameter space, and the program will likely find the one closest to the initial values. That may not be the best fit in two ways: it may not be the lowest minimum and it may not be the one that makes the most sense. If you don't choose reasonable initial values for the parameters, the program may not be able to find a minimum at all.

You may wonder about the $\pm$ uncertainties in the fitted parameters that were reported by the program I used to do these fits. You can imagine that around the point in parameter space where Φ is minimum (the bottom of a "valley") there are contour lines of constant Φ. Any set of parameters that lie inside the contour line where Φ (or χ^2) has twice its value at the minimum are considered to be essentially equivalent results of the fit. That is because, if the parmeters are Gaussian random variables, this contour determines the standard deviation of the parameters. The $\pm$ changes given by the fit program are those that result in doubling χ^2. The *DataStudio* program we use calculates parameter standard deviations in this way.

It is virtually certain that whatever career you choose, you will sometimes make use of least squares fitting to data, also called regression analysis. If Φ is a linear function of the parameters, it is linear reqression; if not (as in the case of B and D in the "Sine Fit" function), it is non-linear regression. Many software packages are now available to implement regression analysis, and it is important for you to be aware of some of the ways they might mislead you.

Our program *DataStudio* reports the "Mean Squared Error" which supposed to be Φ divided by the number of data points and the "Root MSE" which is its square root. Beware that sometimes the program does not do the division and simply reports Φ and its square root. If you know the number of data points in the fit, you can correct the result; the Sine Fit function we use in this experiment is a culprit.

The classic book in the field is by Philip R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences*, McGraw Hill, New York, 1969. It is old enough that the code is written in FORTRAN, but it is still my favorite. A more recent book is by N. R. Draper and H. Smith, *Applied Regression Analysis*, 3rd edition, John Wiley & Sons, 1998.