Curve Fitting in Matlab

Shimi Machluf*

November 8, 2008

Contents

*Like any other document, this one can have mistakes or not be clear enough. I would like to get comments regarding this text to my email: machlufs@bgu.ac.il.

1 Introduction

In this tutorial I assume you know the basics of working with Matlab. For the ones who don't know, there are many other tutorials, like the ones you can find in Matlab help, Highlern or just search the internet. Still, I'll explain the basic commands and will give examples.

Few more general comments:

- The pictures of the computer screen dosn't look clear. This problem is due to the screen size relative to the paper size, zooming in will solve this problem.
- In the pictures that you see a window and the command window in the background, usually you can see in the command window the commands I used.

In Fig.1 we can see the Matlab window, where the big part is the "Command Window" and on the left there are 3 tabs: "Current Directory", "Workspace" and "Command History".

- The "Command Window" is the main window and in it we insert/add data into the Matlab Workspace or writing commands.
- The "Current Directory" is where the Matlab path is set. You can change this directory or open files directly from this tab. If you want

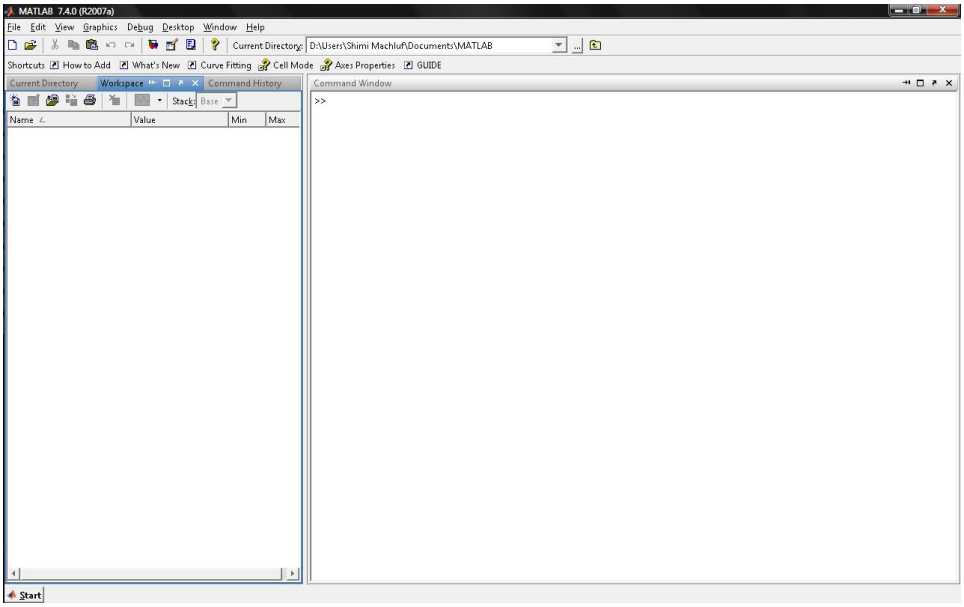

Figure 1: An example of the Matlab window.

to run scripts or functions, they need to be in this directory or you need to go to their directory.

- The "Workspace" is a tab showing all the variables you already defined in this Matlab session. When you close Matlab the Workspace is being deleted, but you can save the Workspace to file and load it in different time with the File menu.
- The "Command History" tab saves all the command you typed in the "Command Window", also in old sessions of Matlab.

Now I'll explain very shortly how to insert data into the Matlab Workspace, manipulate it and then plot it to figure. In Fig.2, I typed 3 commands. The first one:

>> x = [0:0.1:2*pi]

just defines a vector (an array) of numbers. Starting at 0, to 2π with jumps of 0.1. After typing the command Matlab plots the result - the value of the vector x. In the second command:

>> y = sin(x).^2;

I define a new variable and set its values to be $\sin(x)^2$. In this command I

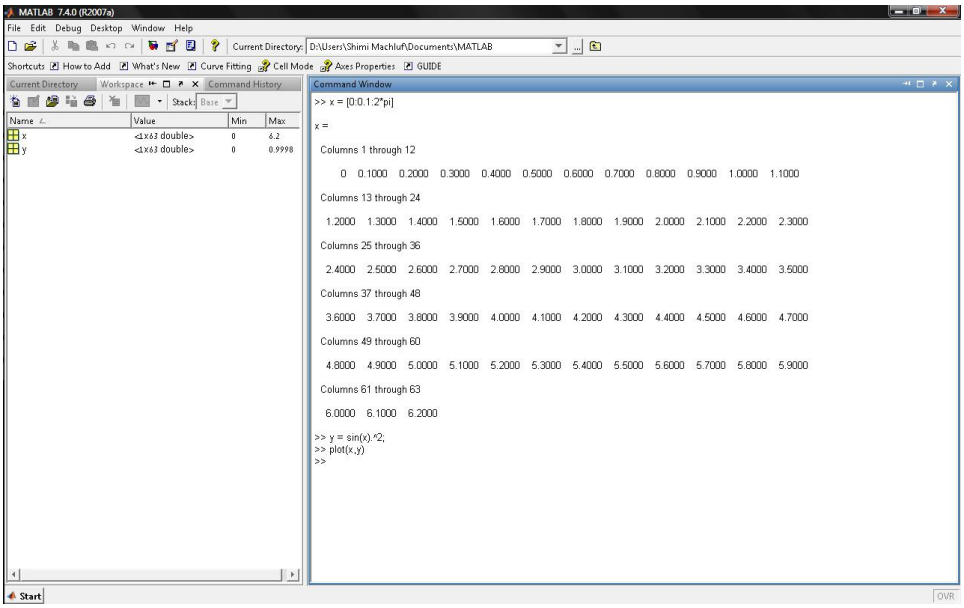

Figure 2: An example to define 2 variables and plot them.

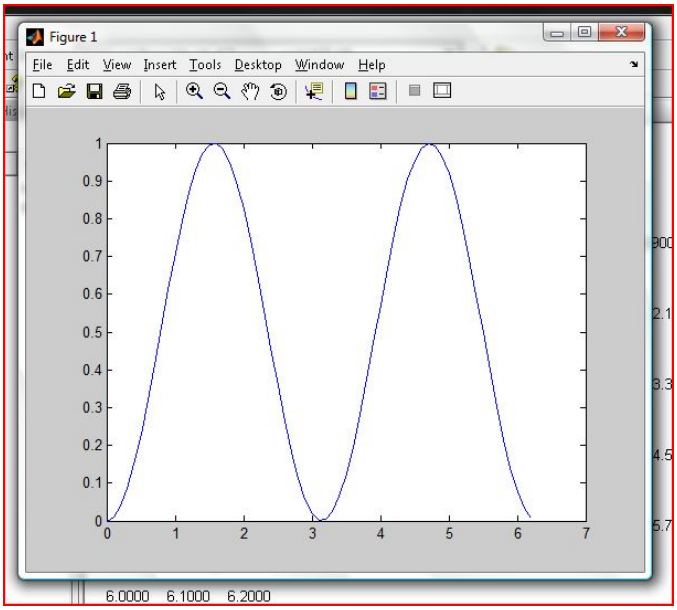

Figure 3: The result of the "plot(x,y)" command.

would like you to pay attention to 2 things:

1. First, to the fact that there was no output. This is because of the semicolon (;) at the end of the command. This semicolon tells Matlab that it shouldn't print the output. You should put the semicolon at the end of every line, unless you want to see the output, for 2 main reasons: It's easier to see what commands you already inserted and when there is a lot to print it can take some time.

2. The second thing is that when I wanted to use the "square" operator I put a dot (.) before it. The reason is that Matlab, by default, works with matrices and vectors, and in order to multiply 2 vectors at size n, their size need to be $[1 \times n] \cdot [n \times 1]$ for scalar product, or $[n \times 1] \cdot [1 \times n]$ for getting the square of each element. Since our vectors have the same size, using the dot before the square operator tell Matlab to do this operation (square in our case) on each element and not on the vector.

The last command just opens a new Figure window (Fig.3) and plots y as a function of x.

I want to mention a few more things before we continue:

- As you can see, the 2 variables were added to the Workspace, see Fig.2. We can see their size and their nim/max values.

- The variable π, is already defined in the Matlab Workspace but we don't see it in the Workspace tab. Also the imaginary number $\imath$ is

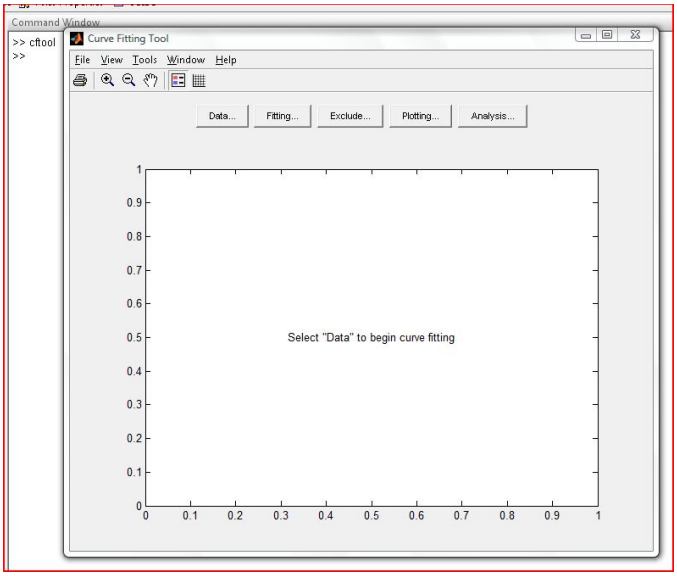

Figure 4: The Curve Fitting window.

already defined as

>> i=sqrt(-1);

- The functions "sin" and "sqrt", square root, are Matlab functions. The Matlab Help contains information about these functions and on any other Matlab function. There are A LOT OF Matlab functions. Any mathematical function like "sin" or "exp" or manipulation function (on vector or matrices) like "max" or "mean" that you can think of, is probably there.

2 The Curve Fitting Tool

to open the Curve Fitting tool just type:

>> cftool

and this tool will be open in a new window. See Fig.4. This window has 5 buttons:

Data...: Will open a new window that will allow us to import data from the Matlab workspace into the Curve Fitting tool environment (called data set).

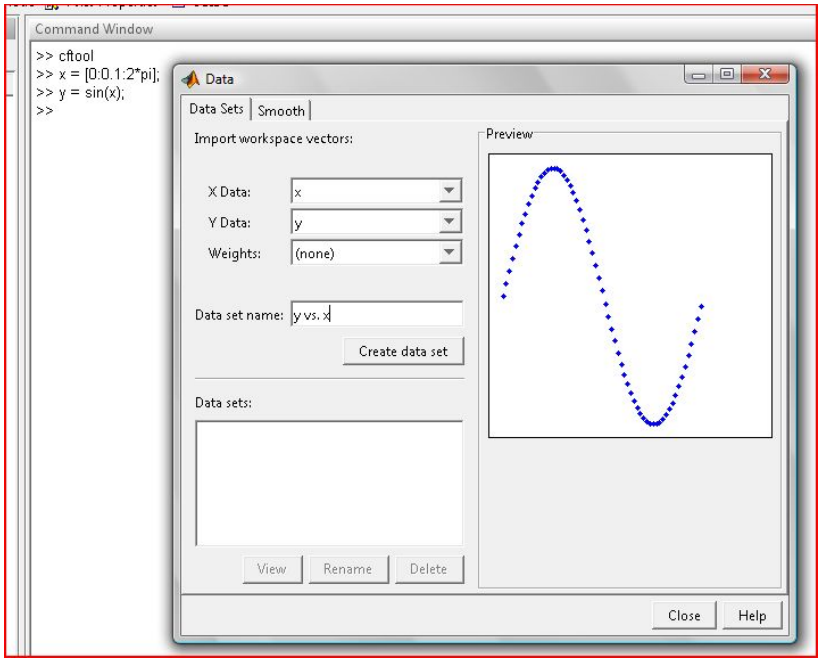

Figure 5: Creating a new data set.

Fitting...: This one will set the fitting function and parameter.

Exclude...: With this window we can exclude some data points from the data set.

Plotting...: If we have a few data set and/or a few graphs, here we can decide which one we want to see, to plot. The rest are just not shown, there are not deleted.

Analysis...: If we want to evaluate the fitted function we can use this window.

The Data... and Fitting... are the most important and I'll explain how to use them in more details using the next example ($y = sin(x)$).

2.1 Data...

After pressing the Data... button, a new window will open, Fig.5, and we could choose the x, y and weights variables[1], set the data set name and click the "Create data set" button. After that you'll see your data set in the list and you can close the window.

[1]These variables MUST exist in the Matlab Workspace, otherwise they won't appear in the list.

In this short example I didn't set any value to the weight because I created the x and y variables. When you want to analyze some data, usually the x parameter is something you choose. Like measuring something every second (or some other time), or every 10 cm. That's the meaning of free parameter. The y parameter is something you measure (the dependent parameter) and since we're in real life, the measurement has errors. Some of the data points will have small error and some will have big error[2]. The regular fit (without weights) consider all the data points as equal, but if some of the data points have larger error bars (relative to the other points) it means that they are less accurate and that the fit should consider these points less in its calculations. That's what the weights are for. The more accurate data points get higher weights relative to the less accurate data points.

The way to create the weights, according to the above logic, is that the weight is the inverse of the square of the error bar. In a mathematical way: if w_i is the weight and σ_i is the error of the i^{th} data point, then:

$$w_i = \frac{1}{\sigma_i^2}.$$

The Smooth tab is for smoothing noisy data, something we don't need for now and I won't explain.

2.2 Fitting...

To start fitting the data, we need to click the Fitting... button so the Fit Editor will open. There we need to start "New fit", give it the name we want (the default is fit and a number), choose the data set we want to fit, choose the fitting function (Type of fit) and click Apply.

There are a lot of built in fitting function which are grouped, like "Sum of Sin Function" (what I used because I want to fit the data to Sin function) or "Polynomial" functions, where in every group you need to choose the specific function you want to fit to.

After clicking the Apply button, the fit is computed and the results of the fit are printed in the "Result" part of the fit editor and also plotted as a graph in the Curve Fitting Tool. In Fig.6 we can see the input I entered the Fit Editor and the result of the fit. In this Figure I also adjusted the Fit Editor so we could see all of the output. Just by looking on the plot we can see that the fit is good and the R-square and Adjusted R-square (check the Help for details) confirm that. We can also see that the amplitude (a1) is 1, the frequency (or more precisely, $\omega = 2\pi f$, which is b1 if our x variable

[2]If we took a few measurement for each x value, we can put all of them into the fit. The error (or error bar) is for each measurement. It can happen that all the measurements for the same x value will have the same error.

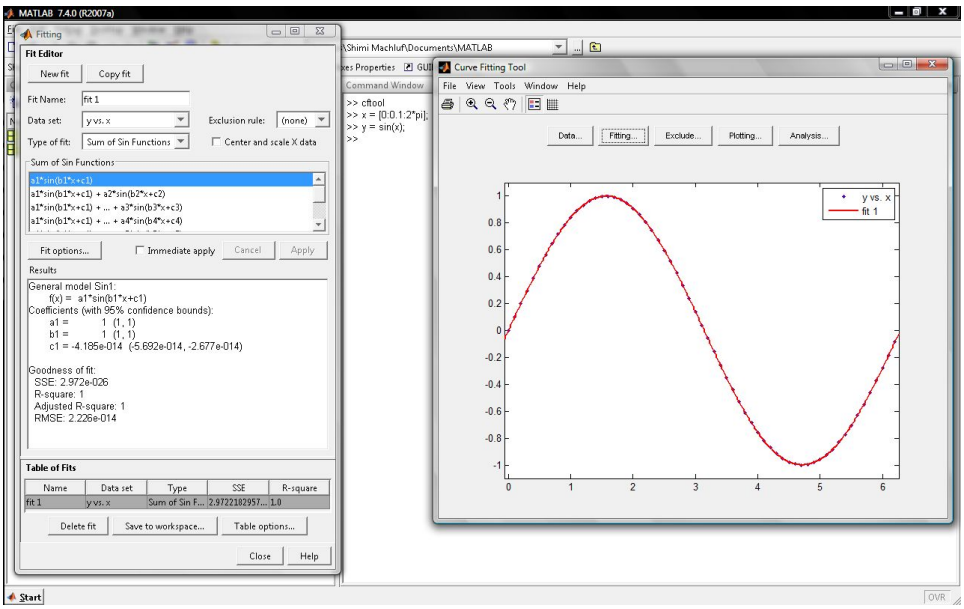

Figure 6: Fitting the data with $y = a1 \cdot sin(b1 \cdot x + c1)$ function.

is the time) is also 1 and that the phase (c1) is 0, like the parameters we choose when we created the y vector ($y = sin(x)$).

The numbers in parenthesis near the values of the parameters (Coefficients) are the confidence bounds - the error in the fit. Since our fit fits exactly the data set, these confidence bounds are zeros (the same value as the parameter itself). In other examples, like in Sec. 3.1, where the fit won't be so good, we'll see that the confidence bounds are not zero any more.

In general, the fitting algorithm start from some values of the fitting constants (a1, b1 and c1, in out example) that Matlab choose, and try to converge from that initial parameter to the final ones. Sometimes the initial parameters are not close enough to the final ones so the algorithm converge to something else or doesn't converge at all. Clicking the "Fit options..." will open a window that allow as to change this initial parameter. I will use that in one of the examples.

If the fit function that we need isn't in the list of functions, we can choose as the Type of fit "Custom Equations" and then "New equation" and to build the needed function. I will show how to do that in example 3.2.

"Save to workspace..." will save the fit data to the workspace, and there we can continue to work with this data in the Command Window. In example 3.1, I'll show another way to save data to the Workspace and to plot it with Error bars.

2.3 Exclude..., Plotting... and Analysis...

The first 2 options will open a new window that allows us to exclude some of the data points from the fitting (they are still in the data set) or to remove some data set or fits from the plot. These 2 option are very intuitive and I'll not elaborate on them.

The analysis will open a window that enable us to get the value of the fitted function in some values that we choose (not related to the data set) and the confidence bounds. We can also get the first 2 derivatives or to integrate the function. I'll explain more when I'll show an example (Sec. 3.1) with confidence bounds.

2.4 Residuals

Going to the menu: View → Residuals → Scatter Plot, will open a new graph that will show the residual - the final difference between our data points and the fit. Fig.7 is what we'll get for our example.

When looking on the residuals it's important to see that they are scattered randomly and evenly around the zero value. That they are not positive in one part of the fit and negative in other part, or that their value isn't small or large in different parts.

In our graph both of these criterions don't exist, but if we look on the value of the residuals, up to $5 \cdot 10^{-14}$, we can understand that these is just rounding (or calculation of the *sin* function) error because the numbers in the computer are not infinitely accurate.

3 Examples

3.1 Simple linear fit with error bars

In this example I'll fit 5 points to linear fit. The input is:[3]

```
>> x=[1 2 3 4 5];
>> y=[1 2 3 4 4.5];
```

and the error bars are:

```
>> err=[0.1 0.2 0.3 0.4 0.5];
```

This is clearly a strait line with slope of 1, just the last point looks to be wrong. One option is to exclude this point from the graph but this option is not the preferred one. We'll see that adding weights has similar result.

[3] This is a way to create a vector with specific values.

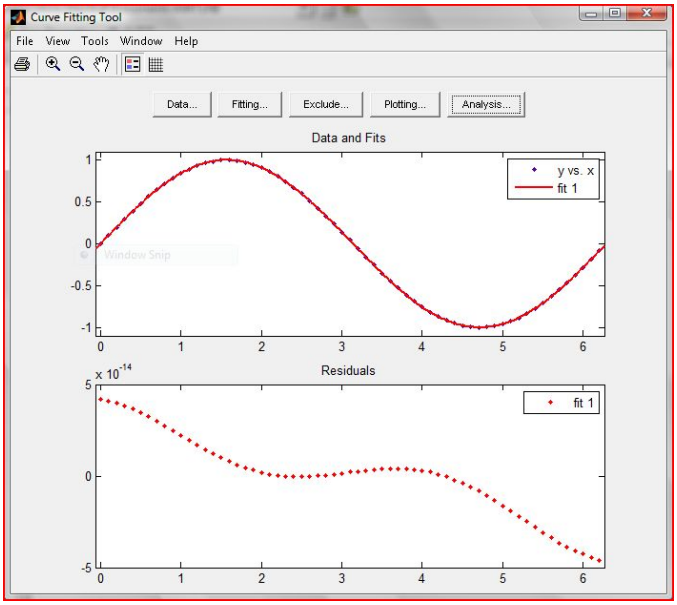

Figure 7: The plot with the residuals.

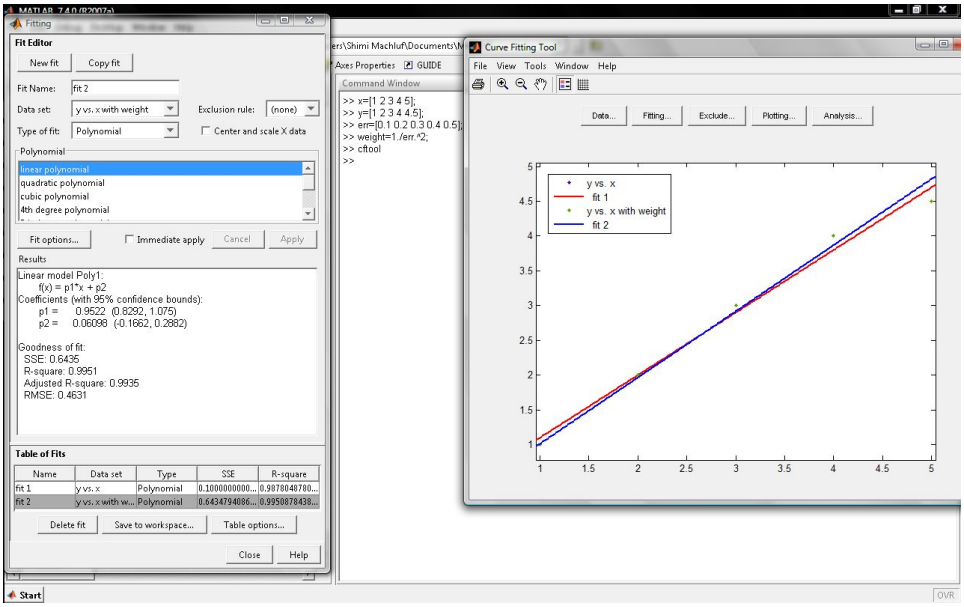

Figure 8: Fitting the data to linear line. The red (blue) fit is without (with) weights

In Fig.8 we can see 2 fits, without (red) and with (blue) weights. In the Fit Editor we see only the result of the fit with the weights, and we see that

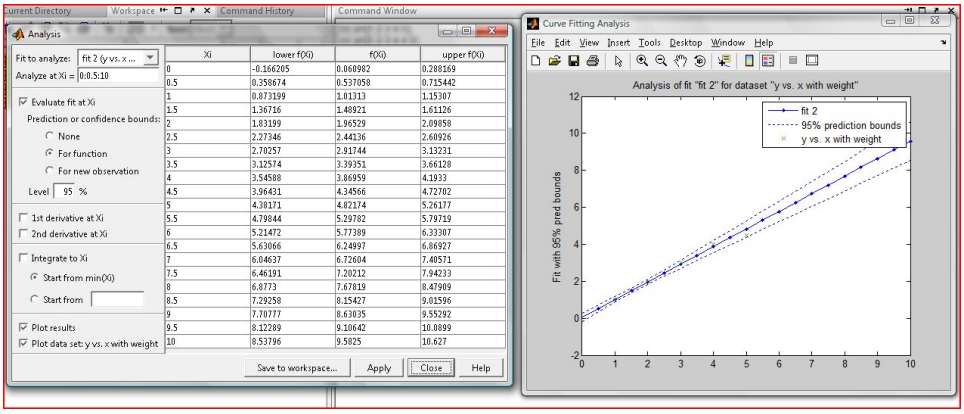

Figure 9: The Analysis window and the output.

the fitted parameters are almost 1 and 0, as we would expect. Now, when the fit isn't perfect, the confidence bound has deferent values. Clearly the fit with the weights gives better results, it still not perfect but our data isn't perfect too. Between the 2 windows, the Matlab commands are also visible (see the way I defined the weights). The error is something that we need to calculate separately - in this case it's 10% of the x value.

Now we'll go back to the Curve Fitting Tool and open the Analysis window. In it we'll choose "fit 2 (y vs. x with weights)", and set the analyze points to be: 0:0.5:10 (between 0 to 10 in jumps of 0.5). After that we'll choose "Evaluate fit at X_i" and "For function". We'll also add "Plot Results" and "Plot data set", and click Apply. The output is displayed in Fig.9.

We see that the fit is relatively accurate where we have data points, but in the range without data points it start to be less accurate. If we'll do the same for the data set without the weights we'll see that the prediction is less accurate.

Plotting the data we have in an Error Bar plot with the fitting function and the confidence bounds is a bit tricky and I'll explain how to do that.

Let's analyze again the data set with weights and save it to Workspace (using the button). A new variable will be created. We can change its name but I used the default one: "analysisresults1". Double click on this variable in the Workspace tab will open the "Array Editor" where we can see what this variable contains.

In the Matlab Command Window type the following commands (everything after % is comments and aren't necessary):

>> figure % open a new figure window
>> errorbar(x, y, err, 'ro') % plots the data, as circles (small letter o - not zero), with errorbars in a red color

>> hold on % Plotting the next plots without deleting the old ones
>> plot(analysisresults1.xi, analysisresults1.yfit, 'b-', analysisresults1.xi, analysisresults1.lower, 'b:', analysisresults1.xi, analysisresults1.upper, 'b:') % plots the fit in solid blue line and the confidence bounds in dotted blue line
>> hold off % release the figure

(If you use copy/paste delete any new line in the Command Window if you have one - all the commands are one line).

Fig.10 shows the result. If you want to adjust the figure, like adding titles, clicking the small icon below the help menu will open the needed editors (or with the menu: View → Plot Browser and View → Property Editor). Choosing the Axes in the Plot Browser will allow us to add titles (in the Property Editor) to the plot and axes, and choosing one of the plots will allow us to change the color, shape and more properties of the plot lines themselves. We can also insert "Legend" or "Textbox" with the "Insert" menu. In the text box we can copy the fit data, for example, see Fig.11.

3.2 More advanced fit

In this example I'll fit a data to sin^2 function. This function isn't a built in function so we'll need to create it. This time the fit won't work with the initial parameters so we'll also need to fix them. In this example I'll use a noisy data but without errorbars - just because I'll already explain about them and in this example I want to show something else, not because they're not important.

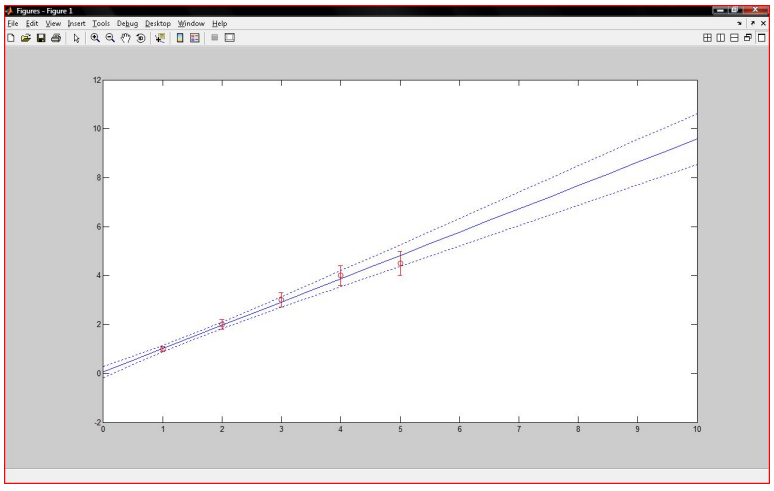

Figure 10: A plot with error bars, the fitted function and the confidence bounds.

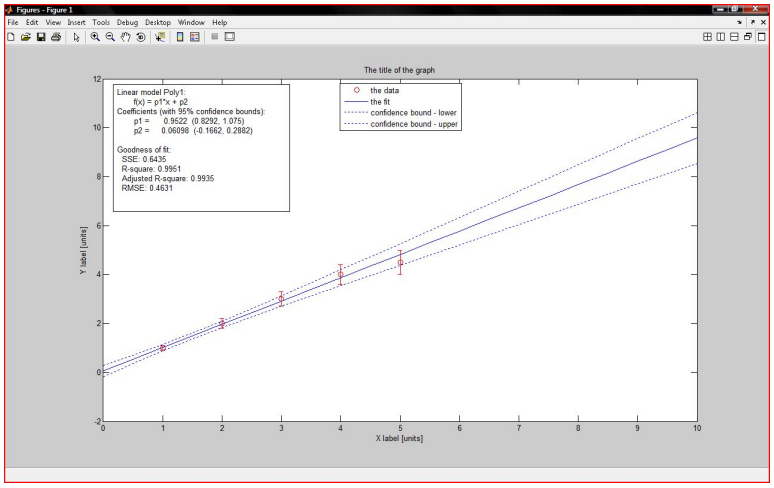

Figure 11: The same plot as in Fig.10 but with titles and some data on the fit.

In Fig.12 we can see the commands that create the data and add some random value. The data is sin^2 with amplitude of 8, frequency of 0.2 and phase of 1, and we would like to get those numbers from the fit.

In order to get to the state of Fig.12, we need to create the variables

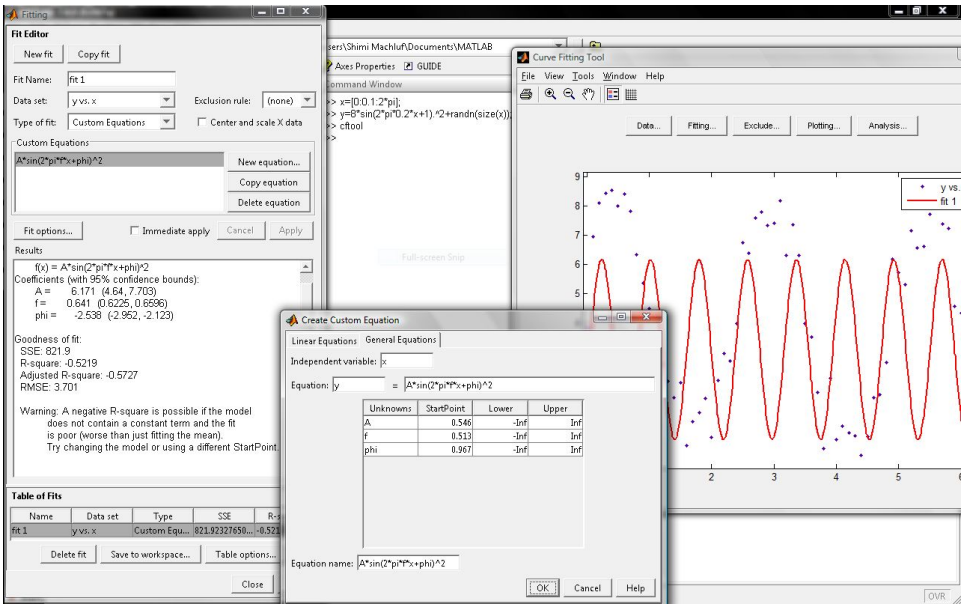

Figure 12: sin^2 example.

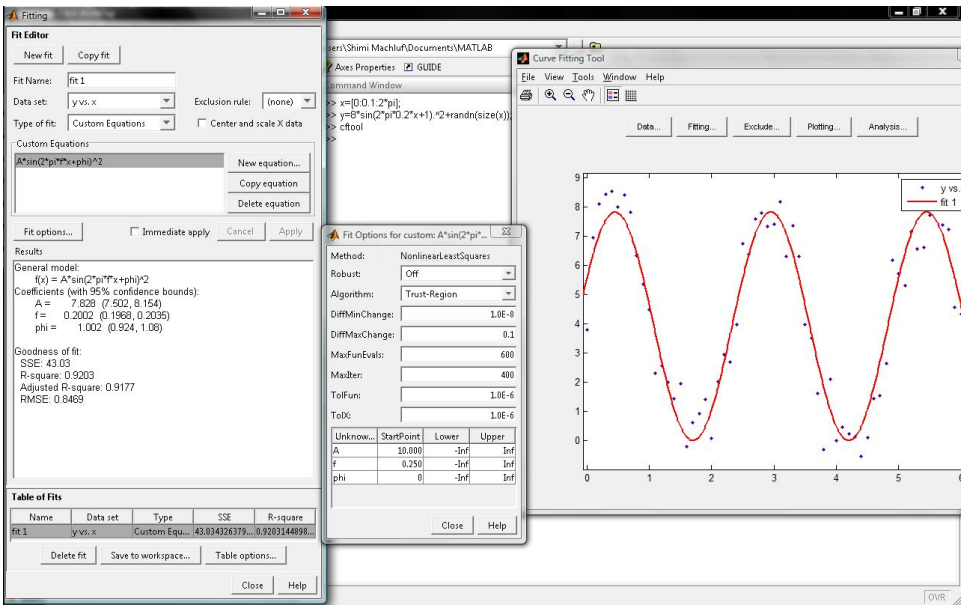

Figure 13: sin^2 example - after changing the initial parameters.

in the Matlab Workspace, open the Curve Fitting Tool and create the data set, and then to open the Fit Editor, create New Fit, Custom Equation

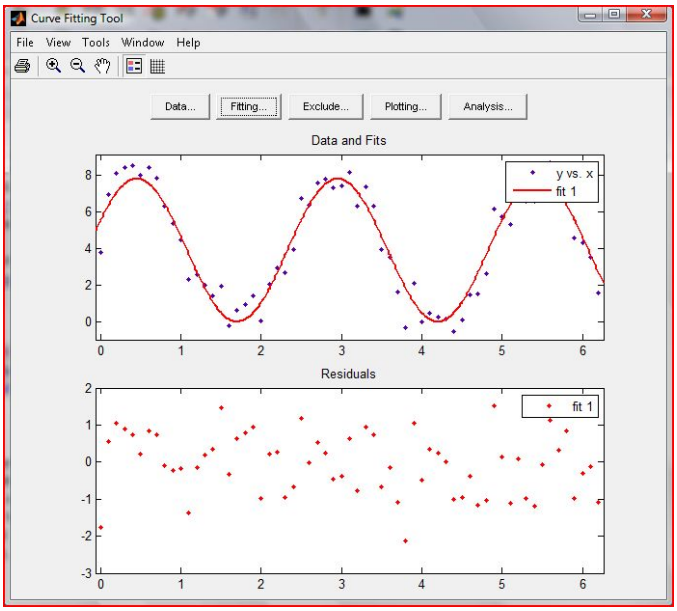

Figure 14: sin^2 example - the residuals of the final fit.

and New Equation. In the new window, "Create Custom Equation", go to "General Equations" tab and type the equation (here we don't need the dot before the operators, like we need it in the Matlab Command Window). Here we should also insert the initial parameters for the fit. I will leave the one Matlab choose only for pedagogic reason, this way I could show how to change them afterward. After clicking OK this equation will be added to the "Custom Equations" in the Fit Editor. Clicking Apply will give the result that is plotted in the Curve Fitting Tool in Fig.12, which is clearly not the right fit. Also in the Result part of the Fit Editor there is a warning that says the same.

Clicking the "Fit Options..." button will open a new window where we can change the fit parameters, see Fig.13.

Achieving the right result can be a bit tricky and annoying, especially for periodic functions, and it requires that after every change you apply the changes and look on the fit to see if it looks right. After you find the right one you need also to plot the residuals and see that they are randomly distributed, see Fig.14. My logic here was that the amplitude is about 10, we see that from the graph. The phase I set to 0, and since in the wrong fit the frequency looks to high, I put less then what I got in the wrong fit.

List of Figures