Using Matlab for Curve Fitting in Junior Lab

MIT Department of Physics
Advanced Experimental Physics I & II
(Dated: June 13, 2008)

1. INTRODUCTION

Curve fitting is one of the most common analytical tasks you will perform during Junior Lab. Students are welcome to utilize any set of routines for curve fitting as long as the standards for reporting results, identified in this and other Junior Lab documents, are met.

This short guide is designed to get you started using Matlab, a commercial product available on the Win-Athena computer cluster in Junior Lab. **Matlab is free for MIT students wishing to run it on their own computers; see `web.mit.edu/matlab/www` and is also available on any Athena workstation at the Institute.** [1]

2. STARTING MATLAB

1. From the Athena prompt, attach the Matlab locker (you can also add this to you .cshrc.mine file to have it automatically attached at login):

   ```
   % add matlab
   ```

2. Open Matlab into a new window:

   ```
   % matlab &
   ```

Once it has finished loading, it will present you with the Matlab prompt: `>>`. From this prompt you can execute any of the Matlab commands or run a Matlab script. To run a script, first make sure it ends in `.m` and resides in your `matlab` directory and then simply type the name at the prompt (without the `.m`):

```
>> myscript
```

3. USING MATLAB SCRIPTS

One very powerful yet simple way to utilize Matlab is to use scripts. Scripts are simply text files that contain a series of Matlab commands. The entire process of curve fitting will require at least a handful of commands so it is useful to have them all in a single script. Once you have this script you can return to it later, repeat your fit, make modifications, etc. without having to retype all of the commands.

If you plan to use Matlab scripts, it's a good idea to create a 'matlab' directory in your home directory. You can do this by typing:

```
% cd ~; mkdir matlab
```

You should save any Matlab scripts that you write in this directory. You can use emacs (or any other text editor) to create and edit Matlab scripts. For example, to create a new script called `myscript.m` type:

```
% >> edit myscript.m
```

Notice the "`.m`" extension on the script. All Matlab scripts must end in "`.m`" in order to execute correctly. The edit command opens up a text editor within Matlab.

4. JLAB FITTING TEMPLATE

To accommodate quick and easy fitting for Matlab beginners, we have created a script that you can use as a template for fitting. This script is available from the Matlab section of the Junior Lab website. You should start by downloading all of the Matlab '.m' files at `http://web.mit.edu/8.13/www/jlmatlab.shtml` to a newly created directory on your Athena lockers.

Before examining the script in detail, try simply running it by typing (within Matlab):

```
>> fittemplate08
```

This script performs the nonlinear fit and produces a publication quality graphic, already for use within a Junior Lab written summary or in an oral exam presentation!!! As you can see, a Matlab scripts are is a very powerful tool with which you will want to become very familiar! By encapsulating all the pertinent information into a 'script' it is very easy to return at a later date to recreate the fit or to apply the same script to a new data set. It is a good idea to get into the habit of adding comments to each new script and to use 'intuitive' naming schemes for your filenames.

The basic procedure for using this fitting script is outlined as follows:

---

[1] While Matlab is the default Junior Lab solution, some students prefer to use alternative mathematics packages (e.g. Gnuplot, Maple, Mathematica or LabVIEW). The only requirement is a solid understanding of the underlying algorithm and mathematics, detailed in many places, most notably Bevington and Robinson (2003), Reference [1]. It is a good idea to have this reference at your side while doing data analysis as you will be continually referring to it throughout your work!

1. **Open** the script in Matlab using the 'edit' command.
2. **Modify** the script such that it:
   (a) Loads data from an arbitrary space delimited file into the vectors `x` and `y`
   (b) Assigns the appropriate errors (this is the hard part!)
   (c) Contains the functional form that you want to fit (consider adding baseline terms for real data containing noise and insrumental offsets!)
   (d) Plots figures with labels appropriate for your data
3. **Save** the file with a new name, keeping the .m extension
4. Run the script!
5. Matlab will **fit** your data, output the information relevant to the fit, and **plot** (1) the fitted curve on top of the original data and (2) the residuals.

We will now go through this process in detail.

4.1. Non-Linear Least Squares Example

In developing this guide, we have used the Statistics Reference Dataset "Gauss3" available from the National Institute of Standards in Technology at `http://www.itl.nist.gov/div898/strd/nls/data/gauss3.shtml`. Since NIST has kindly provided certified values for this dataset, it is an excellent test case for checking the acceptability of alternative data fitting products. You should always test your fitting algorithms on a certified data set to test it's accuracy before applying it to your own Junior Lab data!

The sample dataset, entitled "gauss3.dat" (downloaded previously from the Matlab section of the Junior Lab web page), consists of two poorly resolved Gaussian peaks on a decaying exponential background and must be fit using using a general (nonlinear) custom model. It has normally distributed zero-mean noise with a variance of 6.25.

Here we step through the process of modifying the template script to fit this 'known' dataset. Note also that the any and all parts of the "template script" may be entered directly from the matlab command line. This is useful when diagnosing script generated errors.

1. **Open the script in the Matlab editor window**

```
% >> edit fittemplate08.m
```

2. **Modify the script such that it:**

(a) Loads data from your file into the vectors `x` and `y`.

Currently the script is set up to load the simulated data file called 'gauss3.dat' described earlier. It can be changed to load whatever dataset name you require:

```
load gauss3;
```

The result of a `load` command on a tab-delimited text file is a single "matrix" variable with the name gauss3 (250 rows x 2 col). The next lines in the script assign values to the `x` and `y` vectors from this parent matrix.

```
x=gauss3(:,1);
y=gauss3(:,2);
```

(Note that this syntax utilizes the entire data set in the vector definitions. By placing indices on either side of the colon, a subset of the entire data file may be selected for fitting.)

(b) Assigns the appropriate errors

You will want to create a vector of error values. While the error values may or may not be uniform, the error vector must be the same size (length) as the vectors `x` and `y`.

In the case of gauss3, we have a constant variance of 6.25 so we can assign the weights with the commands:

```
sig=ones(size(x))*sqrt(6.25);
```

which creates a weighting vector of the same size as the vector `x` with a constant value of one over 2.5. See Reference [1] for more information on weighting vectors.

Note: Since there are several different ways you could assign values to the error vector `sig`, the script includes several different assignment statements that cover a few cases. Once you have chosen one and modified it for the fit you are doing, you will need to *comment out* the other lines that assign values to sigma. To comment something out, simply put a '`%`' in front of it. LaTeX regards any thing following a '`%`' as a comment and does not interpret it. You'll notice that the template is heavily commented to explain what the commands are doing.

(c) Contains the functional form that you want to fit

The two general classes of functions that you will fit in Junior Lab are simple straight lines that can be solved in closed form and arbitrary functions which must be solved iteratively. The template script invokes the functions `fitlin.m` and `levmar.m` for these two cases respectively. Each of these cases is handled slightly differently within the fitting template. We will discuss each separately.

Arbitrary Functions

When fitting an arbitrary non-linear function (e.g. a gaussian or exponential function), you will need to specify the following:

- The `x` and `y` vectors of your data,
- A vector `sig` containing the uncertainties of each data point,
- The functional form you want to fit to the data,
- An vector with an initial guess of the parameters.

The function `levmar.m` performs the fit by iteratively searching for the minimum value of the Chi-Square (χ^2) using the Marquardt algorithm detailed in Bevington and Robinson (2003). Each iteration varies the parameter by a certain stepsize to determine the minimum to within a certain tolerance (the variable 'chicut' in the matlab script). Thus, your fit is more likely to be fast and accurate with a carefully chosen set of initial parameters. Additionally, being able to estimate parameters quickly from graphs of data is an important skill that will serve you well.

It is also possible for such a fitting process to converge on a local minimum in χ^2 space that does not represent the best fit. If this is happening to you, you can modify both the 'stepsize' and the tolerance 'chicut' used by `levmar.m`. Both of these parameters are labeled within the function.

Straight Lines

When fitting a straight line, simply comment out the line that specifies the starting parameters `a0` and the line that calls `levmar` within the script. Then, simply uncomment the line that calls `fitlin`. Notice that for a straight-line fit, you only need to specify the `x` and `y` vectors of your data and the `sig` vector containing the error values. You do not need to specify starting parameters.

(d) Plots figures with labels appropriate for your data

Running the script will automatically generate plots for you. You'll need to make sure these graphs display the data you want with the correct labels.

Currently the script will generate one figure that is split into two sections. The top section will contain the fitted curve plotted on top of your data points with errorbars. The bottom section will contain the residuals of your fit.

The commands `'title'`, `'xlabel'`, and `'ylabel'` label the title, x axis and y-axis of the graph respectively. Right now the script just gives them generic titles. Make sure to modify these lines to give your graph meaningful labels with units.

3. **Save the file with a new name, keeping the .m extension.**
   Be sure to save it with a *different* filename so that the template script will be left intact for your next fitting adventure.

4. **Open Matlab and run the script.**
   If the script is saved in your matlab directory, you should be able to run it simply by typing in the name at the Matlab prompt (without the the `.m`). You may need to change directories if the file is stored elsewhere. Matlab has the same directory navigation commands as Athena such as `'ls'`, `'cd'`, `'pwd'`, etc.

5. **Matlab will fit your data, output the information relevant to the fit, and plot (1) the fitted curve on top of the original data and (2) the residuals.**

The script is currently setup to output the values determined for the coefficients and their errors as well as the reduced chi-square[2] ($\chi^2_{\nu-1}$) of the fit. The script causes these particular values to be output because they are declared with no semi-colon. In general, a semi-colon at the end of a Matlab command suppresses the output of the command.

5. GENERATING PRESENTATION GRAPHICS

Figure 1 demonstrates a typical graphic you might create for an oral presentation or written summary. The

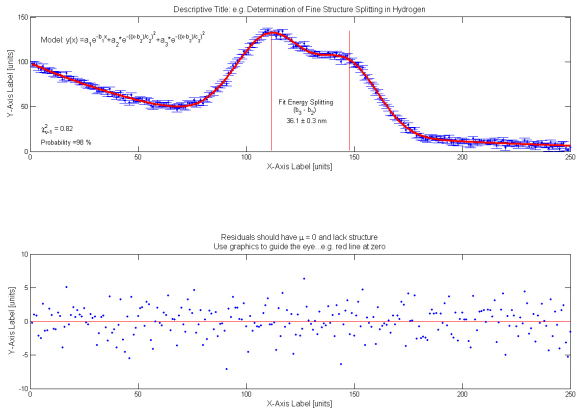

FIG. 1: This is an example of a basic figure for Junior Lab notebooks, presentations and written summaries. **You should check with your individual section instructor for explicit instructions on how to prepare figures for presentation within your section.**

fitting template script is already configured to generate graphs of your data, the fitted curve and the residuals. However, as previously mentioned you will need to customize the titles and axis labels so they are relevant and meaningful to your graph. In addition, you may also wish to annotate your graph with additional information. **The JLAB Fitting Template has several examples of how to automatically place properly formatted fit results within a plot.** You can do this graphically with the tools provided in the graph window or by using the `'text'` command within the script. To use the `'text'` command you simply specify x and y coordinates and the string you wish to appear. The coordinates should be in the same units of the graph to which they refer. For example:

```
text(18,5,'y(x) = ae^{-bx}+a_1e^{-((x-b_1)/
```

[2] This is simply the χ^2 (a statistic used frequently by physicists and in Bevington) divided by the DFE (Error Degrees of Freedom). The DFE is equal to the number of data points minus the number of fitted coefficients.

```
c_1)^2}+a_2e^{-((x-b_2)/c_2)^2})
```

would provide an appropriate label for the gauss3 graph in the lower left-hand corner. Notice also that Matlab can interpret latex formatting to display Greek characters, superscripts, and subscripts.

When you have your graphs just the way you (and your section instructor!) want them (a completed graph might look like Figure 1) , you can output them for use in your lab notebooks, written summaries, and oral presentations. If you simply want printouts of your graphs for your notebooks, select `'Print...'` from the `'File'` menu and make sure the correct printer is specified.

You should save your graphic in Portable Document Format (PDF) for easy inclusion into LaTeX generated reports and oral presentations. Be sure to scale your text appropriately for the desired medium (fonts should be much larger for oral presentation slides).

6. GETTING STARTED WITH MATLAB AT MIT

Exhaustive details about running Matlab can be found at: `web.mit.edu/matlab/www` including a several introductory guides and a free (certificates based) on-line tutorial. Beyond these, perhaps your best resource is simply to talk to friends and classmates; many of them have a great deal of Matlab experience!

[1] Bevington, P.R., and D.K. Robinson, *Data Reduction and Error Analysis for the Physical Sciences, 3rd Ed.*,WCB/McGraw-Hill, Boston, 2003

[2] Levenberg, K., "A Method for the Solution of Certain Problems in Least Squares", *Quart. Appl. Math*, Vol. 2, pp. 164-168, 1944

[3] Marquardt, D., "An algorithm for Least Squares Estimation of Nonlinear Parameters", *SIAM J. Appl. Math*, Vol. 11, pp. 431-441, 1963

[4] Branch, M.A., T.F. Coleman, and Y. Li, "A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems", *SIAM Journal on Scientific Computing*, Vol. 21, Number 1, pp. 1-23, 1999

[5] Hecht, E., *Optics: 4th Edition*,Addison-Wesley, 2002

[6] Kittel, C., and Kroemer, H., *Thermal Physics, 2nd Ed.*, W.H. Freeman, 1980