Data and Error Analysis
&
Advanced Physics Lab Fitter Guide

David Wong
PHY327 Advanced Physics Laboratory
University of Toronto

January 31, 2006

Contents

1 Motivation for Error Analysis

Experimentalists have a great responsibility in the physical sciences. They are responsible for proving or disproving a physical theory, measuring parameters of importance, and discovering phenomena that can be characterized. For all of these responsibilities, the experimentalist also must ensure that what he says is justified. In a nutshell, he must always know how well he knows what he knows.

In addition, experimentalists want to be efficient and make the most of their lab time. The statistical uncertainty in any experiment can always be characterized by repeating the experiment. Of course, if you had to repeat the same experiment 1000 times just to figure out the uncertainty in your result, you'd spend years trying to measure the density of water or acceleration due to gravity. By employing select tools from the field of statistics, experimentalists can make good estimates of the uncertainties in their experiments without having to perform them thousands of times. Furthermore, the statistical tools allow the experimentalist to justify the quality of his data and the confidence in his results.

2 Statistics and Measurement

One of the tenets of experimental science is **repeatability**. This means that if you run two iterations of an experiment, and you follow the same procedure, you should get the same result. A question you may ask is: What is meant by the "same" result? Are some pairs of numbers more equal than others? Clearly, we don't expect a result to be exactly equal. No quantity can be measured exactly, so there is a problem in definition here. We can define quantities like c or μ_0 exactly, but by doing so, they cannot be measured. So, since a quantity cannot be measured exactly, how do we describe our measurements?

2.1 Random Variables

A random variable is a function, X, that generates a numerical measurement, x, from an outcome state, ζ, of a random experiment[1] [1, p.84]. This can be equivalently

[1]By random experiment, I mean an experiment with some random component, which means **all** experiments, really. The outcome state, ζ is a formal method of assigning results to a number. For example, in coin-flipping you can assign $\zeta(\text{heads}) = \pi$ and $\zeta(\text{tails}) = 55.32$, but any two unique numbers are just as valid. If the measurement is number of heads, the function $X(\zeta)$ simply has to map $\pi \to 1$ and $55.32 \to 0$.

written as

$$x = X(\zeta). \tag{2.1}$$

If we want to describe the behavior of the measurement, x, we can consider the probability that the measurement will lie within some specified range of measurement values. If the probability is significant for a small range of values, we can say that the measurement is precise. How can we characterize these probabilities? We can use the probability distribution function[2] (PDF), $f_X(x)$, for the measurement x of random variable X.

To provide an analogy to quantum mechanics, the PDF is similar to the magnitude of the wavefunction. You can't determine the probability of an electron being at a particular position, but you can evaluate the probability of the electron being in one half of a potential well. Similarly, since you can't measure physical quantities exactly, you can't determine the probability of the measurement being some x_0. However, you can evaluate the probability of the measurement being in some range $[x_0, x_1]$ by integrating the PDF over the bounds.

$$P(x \in [x_0, x_1]) = \int_{x_0}^{x_1} f_X(x)\, dx. \tag{2.2}$$

As an example, suppose you are measuring current with an analog picoammeter. The problem is that the reading is fluctuating, so even though the scale allows you to read to very high precision, such a reading wouldn't be appropriate. Ruling out systematic error, one can define a random variable to describe the reading of the picoammeter. The PDF of the random variable should completely describe the statistics of the measurement. So, the probability of measuring between 1.0000 and 1.0001 pA may be very small, but the probability of measuring between 0.99 and 1.01 pA may be significant ($P \sim 70\%$).

Random variables can be difficult to characterize because a measurement is described by a curve instead of a single number. Fortunately, the curves can usually be reduced to 2 parameters that describe the measurement equally well. For measurements, the PDF is usually peaked about a mean, μ, with breadth characterized by a variance, σ^2.

$$f_X(x) \iff [\mu, \sigma^2] \iff \mu \pm \sigma \tag{2.3}$$

This is where the notion of the uncertainty of a measurement comes from. Our measurement can be written as $x = \mu \pm \sigma$, but to be able to use such parameters, the shape of the PDF must be known at least approximately. Of course, the magnitude of the uncertainty is somewhat arbitrarily determined. By convention, the uncertainty is generally quoted so that the probability you will measure a value of $x \in [\mu - \sigma, \mu + \sigma]$ is 68%.

[2] Also known as probability density function

So in our picoammeter example, suppose we find that the needle lies within 0.99 and 1.01 pA about 70% of the time. Thus, a reasonable measurement of the current would be $I = 1.00 \pm 0.01$ pA.

2.2 Probability Distribution Functions

2.2.1 Gaussian Distribution

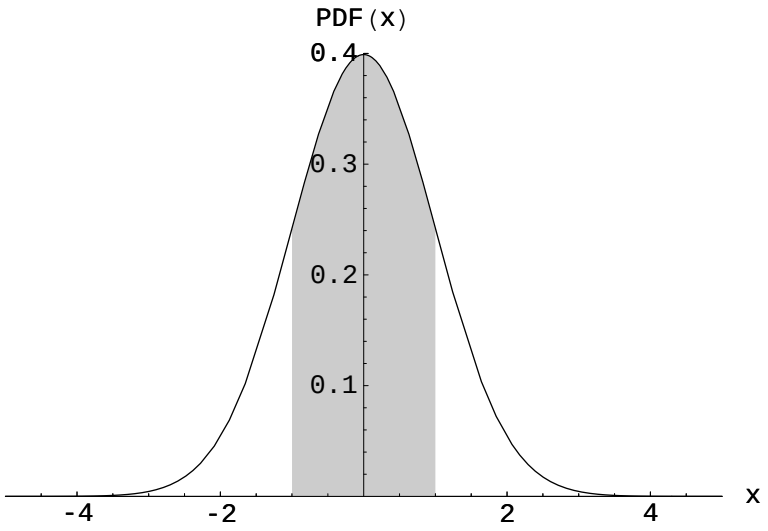

Figure 2.1: Gaussian distribution with $\mu = 0$, $\sigma = 1$ and 68% confidence region shaded.

The Gaussian or normal distribution is probably the most common kind of error statistic that you will find in the lab. Any process which is being affected by a multitude of random effects which are essentially averaged into your result has a Gaussian probability distribution [2, p.27]. It usually the "default" PDF that you assume for your measurement. There is good reason for this choice. It is easy to work with mathematically, many other distributions approach the Gaussian distribution in common limiting cases, and it is qualitatively reasonable. The distribution is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(\frac{-(x-\mu)^2}{2\sigma^2}\Big), \tag{2.4}$$

where x is the measured value, μ is the mean, and σ is the standard deviation of the PDF. The Gaussian distribution is so common that it was the choice for the uncertainty convention of 68%. An uncertainty of one standard deviation is consistent

with 68% probability as shown below

$$\int_{\mu-\sigma}^{\mu+\sigma} \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(\frac{-(x-\mu)^2}{2\sigma^2}\Big)\, dx = \mathrm{erf}\Big(\frac{1}{\sqrt{2}}\Big) \approx 0.6827. \tag{2.5}$$

2.2.2 Poisson Distribution

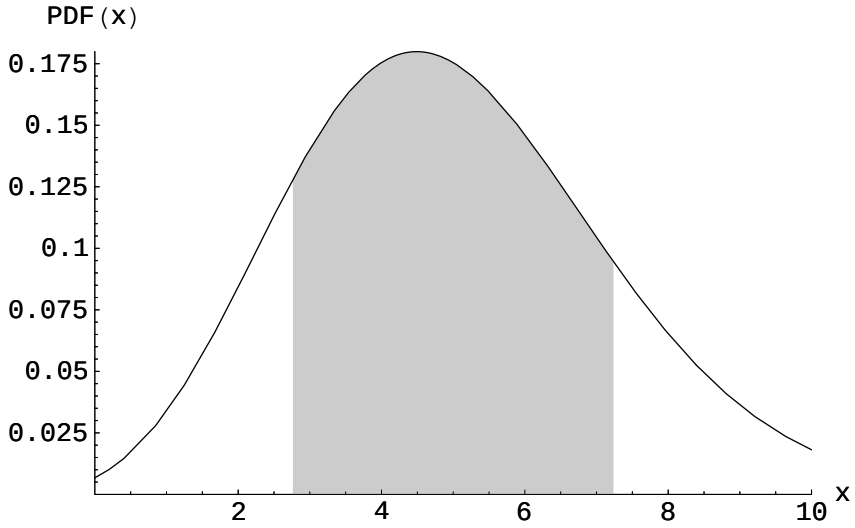

Figure 2.2: Poisson distribution with $\mu = 5$ and $\sim 70\%$ confidence region shaded.

The Poisson distribution is another common probability distribution that you may encounter in the lab. The Poisson distribution is most easily understood as a limiting case of the binomial distribution where the number of possible events is large, but the probability of any one of them happening is very small. If you were to count the number of successes (events) in a period of time, the distribution of counts adheres to the Poisson distribution. The number of events must be non-negative, and the number of events must be an integer, so the Poisson distribution is only defined for non-negative integers. However, a continuous distribution can be defined using special functions. The Poisson distribution is given by

$$f_n = \frac{e^{-\mu}\mu^n}{n!} \qquad f(x) = \frac{e^{-\mu}\mu^x}{\Gamma(x+1)}, \tag{2.6}$$

where n is the number of counts, x is the interpolated number of counts, μ is the mean value of counts and Γ is the Euler gamma function. An interesting feature of the Poisson distribution is that there is no freedom in the standard deviation or variance of the distribution. If we calculate the variance we find that

$$\sigma^2 = \langle (x-\mu)^2 \rangle = \sum_{n=0}^{\infty} \frac{e^{-\mu}\mu^n}{n!}(x-\mu)^2 = \mu. \tag{2.7}$$

In other words, the variance is equal to the mean, and so the standard deviation of the Poisson distribution is $\sqrt{\mu}$. The probability of obtaining $x \in [\mu - \sqrt{\mu}, \mu + \sqrt{\mu}]$ is also quite close to $\sim 70\%$, so it is reasonable to state the uncertainty of a counting measurement to be $\mu \pm \sqrt{\mu}$. This also means that you need to increase 2 orders of magnitude to reduce relative error by 1 order of magnitude, which makes precision measurements very time consuming.

As an example, suppose you have a chunk of radioactive material with a substantial half-life and you wish to count the decays from the sample to calculate the half-life, or the source intensity. What we have is a large number of possible events (i.e. every single atom decaying simultaneously!) but a very small probability for an individual event occuring. In 10 minutes, suppose you count 294 "beeps" on your detector. Assuming 100% detector efficiency, you can estimate the decay rate as

$$\lambda = \frac{294 \pm \sqrt{294}}{600s} = 0.490 \pm 0.029 \text{ s}^{-1}. \tag{2.8}$$

An important note is that as μ gets large, the Poisson distribution is well approximated by a Gaussian distribution with $\sigma = \sqrt{\mu}$. In fact, the two are nearly indistinguishable for $\mu > 10$. So, for the purposes of analysis, using Gaussian statistics is desirable for large μ since it is simpler than using Poisson statistics.

2.2.3 Lorentzian Distribution

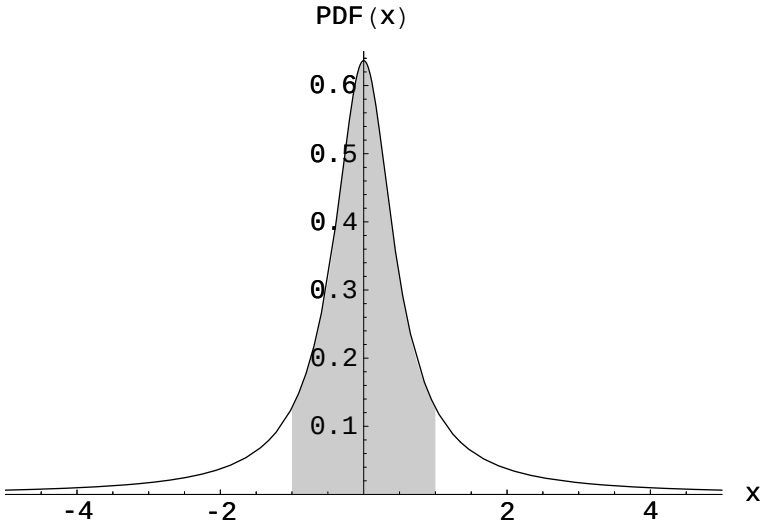

Figure 2.3: Lorentzian distribution with $\mu = 0$, $\Gamma = 1$, and $\sim 70\%$ confidence region shaded.

The Lorentzian distribution is slightly less common, but has some usage in the undergraduate lab. It is suitable for describing the probability distribution of resonant absorption (Mössbauer Effect) versus energy or the distribution of reaction cross section versus energy for a nuclear reaction [2, p.31].

The Lorentzian probability distribution function is given by

$$f(x) = \frac{1}{\pi} \frac{\Gamma/2}{(x-\mu)^2 + (\Gamma/2)^2}, \tag{2.9}$$

where x is the measured value, μ is the mean value, and Γ is the full width half maximum[3]. The Lorentzian distribution is different than the Gaussian and Poisson distributions because it falls into the class of infinite-variance distributions. If you calculate the variance of the distribution, you will find that it diverges. However, $\sim 70\%$ of the distributions area is contained within Γ of the peak as shown below

$$\int_{\mu-\Gamma}^{\mu+\Gamma} \frac{1}{\pi} \frac{\Gamma/2}{(x-\mu)^2 + (\Gamma/2)^2}\, dx = \frac{2\tan^{-1}(2)}{\pi} \approx 0.7048. \tag{2.10}$$

As an example, suppose you perform a nuclear resonance measurement and find the FWHM of the absorption peak is 1 eV, then the absorption peak energy can be quoted to ± 1 eV with 70% confidence.

3 Quadrature and Error Analysis

The ideas of error propagation were introduced in first year as a set of rules to follow when performing operations on data with errors. Unfortunately, there wasn't much justification for using these rules and all we knew was that they worked. Here, I will outline a general method of finding out how to determine the propagation of errors [2, p.39].

We can define variance of a random variable by the ensemble average of the square of the deviation from the mean,

$$\sigma_u^2 = \langle (u_i - \bar{u})^2 \rangle. \tag{3.1}$$

The square root of variance is the standard deviation. The uncertainty in a quantity is assumed to be its standard deviation, σ_u, which is at least true for Gaussian error statistics. We will use these ideas of variance and standard deviation to derive the rules of quadrature.

[3]The full width half maximum (FWHM) is the width of the distribution at half its peak value.

3.1 Functions of One Variable

Suppose that you have a measured quantity u and you wish to find the uncertainty in a calculated quantity $x = f(u)$.

$$\sigma_x^2 = \langle (x_i - \bar{x})^2 \rangle \tag{3.2}$$

Uncertainties in u are directly related for the uncertainties in x, so we Taylor expand the function, $f(u_i)$ about the mean $\bar{u}$ to find

$$f(u_i) = f(\bar{u}) + (u_i - \bar{u}) f_u(\bar{u}) + \frac{1}{2}(u_i - \bar{u})^2 f_{uu}(\bar{u}) + \cdots . \tag{3.3}$$

Rearranging and assuming the errors are small, we drop all terms of quadratic or higher order to find

$$\begin{aligned} x_i - \bar{x} &= (u_i - \bar{u}) f_u(\bar{u}) + \frac{1}{2}(u_i - \bar{u})^2 f_{uu}(\bar{u}) + \cdots \\ x_i - \bar{x} &\simeq (u_i - \bar{u}) f_u(\bar{u}). \end{aligned} \tag{3.4}$$

If we use this linear approximation to calculate the variance of x, we find

$$\begin{aligned} \langle (x_i - \bar{x})^2 \rangle &\simeq \langle f_u(\bar{u})^2 (u_i - \bar{u})^2 \rangle \\ &\simeq f_u(\bar{u})^2 \langle (u_i - \bar{u})^2 \rangle \\ \sigma_x^2 &\simeq f_u(\bar{u})^2 \sigma_u^2 \\ \Rightarrow \sigma_x &\simeq |f_u(\bar{u})| \sigma_u. \end{aligned} \tag{3.5}$$

Thus, the uncertainty in the calculated quantity is simply scaled by the slope of the calculating function at the mean. This makes sense intuitively as it's the same principle of operation for electronic amplifiers, and the SQUID.

3.2 Functions of Two Variables

The concepts used for one variable dependence can also be used for multivariable dependence. Suppose we have two measured quantities, u, and v, and a calculated quantity $x = f(u, v)$.

We can perform the same expansion as before to obtain

$$f(u_i, v_i) = f(\bar{u}, \bar{v}) + (u_i - \bar{u}) f_u(\bar{u}, \bar{v}) + (v_i - \bar{v}) f_v(\bar{u}, \bar{v}) + \cdots \tag{3.6}$$

$$x_i - \bar{x} \simeq (u_i - \bar{u}) f_u(\bar{u}, \bar{v}) + (v_i - \bar{v}) f_v(\bar{u}, \bar{v}). \tag{3.7}$$

Using this result to calculate the variance of x, we find

$$\begin{aligned}
\langle (x_i - \bar{x})^2 \rangle &\simeq \langle ((u_i - \bar{u}) f_u(\bar{u}, \bar{v}) + (v_i - \bar{v}) f_v(\bar{u}, \bar{v}))^2 \rangle \\
&\simeq \langle (u_i - \bar{u})^2 f_u(\bar{u}, \bar{v})^2 \rangle + \langle (v_i - \bar{v})^2 f_v(\bar{u}, \bar{v})^2 \rangle \\
&\quad + \langle 2(u_i - \bar{u})(v_i - \bar{v}) f_u(\bar{u}, \bar{v}) f_v(\bar{u}, \bar{v}) \rangle \\
\sigma_x^2 &\simeq \sigma_u^2 f_u(\bar{u}, \bar{v})^2 + \sigma_v^2 f_v(\bar{u}, \bar{v})^2 + 2\sigma_{uv}^2 f_u(\bar{u}, \bar{v}) f_v(\bar{u}, \bar{v}), \qquad (3.8)
\end{aligned}$$

where the term σ_{uv}^2 is called the covariance of u and v and is defined as

$$\sigma_{uv}^2 = \langle (u_i - \bar{u})(v_i - \bar{v}) \rangle. \qquad (3.9)$$

If the two measured quantities are perfectly uncorrelated, or completely independent, then the covariance of the two quantities is zero. On the other hand, if the two measured quantities are perfectly correlated, then the covariance is the geometric mean of the variances of u and v.

We will assume that all quantities are independent, so all covariances are zero. Now, the variance of the calculated quantity can be found to be

$$\sigma_x^2 \simeq \sigma_u^2 f_u(\bar{u}, \bar{v})^2 + \sigma_v^2 f_v(\bar{u}, \bar{v})^2. \qquad (3.10)$$

Let's try to recover the quadrature rules that we already know.

Let $f(u, v) = u \pm v$.

$$\begin{aligned}
\sigma_x^2 &\simeq \sigma_u^2 + \sigma_v^2 \\
\Rightarrow \sigma_x &\simeq \sqrt{\sigma_u^2 + \sigma_v^2}. \qquad (3.11)
\end{aligned}$$

Let $f(u, v) = u \cdot v$.

$$\begin{aligned}
\sigma_x^2 &\simeq \sigma_u^2 \bar{v}^2 + \sigma_v^2 \bar{u}^2 \\
\frac{\sigma_x^2}{\bar{u}^2 \bar{v}^2} &\simeq \frac{\sigma_u^2}{\bar{u}^2} + \frac{\sigma_v^2}{\bar{v}^2} \\
\Rightarrow \frac{\sigma_x}{\bar{x}} &\simeq \sqrt{\frac{\sigma_u^2}{\bar{u}^2} + \frac{\sigma_v^2}{\bar{v}^2}}. \qquad (3.12)
\end{aligned}$$

Let $f(u, v) = u/v$.

$$\begin{aligned}
\sigma_x^2 &\simeq \frac{\sigma_u^2}{\bar{v}^2} + \frac{\sigma_v^2 \bar{u}^2}{\bar{v}^4} \\
\frac{\sigma_x^2 \bar{v}^2}{\bar{u}^2} &\simeq \frac{\sigma_u^2}{\bar{u}^2} + \frac{\sigma_v^2}{\bar{v}^2} \\
\Rightarrow \frac{\sigma_x}{\bar{x}} &\simeq \sqrt{\frac{\sigma_u^2}{\bar{u}^2} + \frac{\sigma_v^2}{\bar{v}^2}}. \qquad (3.13)
\end{aligned}$$

3.3 Functions of n Variables

If extended to n independent variables, we can approximate the uncertainty in a calculated quantity in exactly the same way as outlined for one and two variables. Supposing that $x = f(\mathbf{U})$, where $\mathbf{U} = (u_1, u_2, \ldots, u_n)$, it can be shown that the variance in x is approximately

$$\sigma_x^2 \simeq \sum_{i=1}^{n} \sigma_{u_i}^2 \left(\frac{\partial f}{\partial u_i} \right)_{\bar{\mathbf{U}}}^2 . \tag{3.14}$$

4 Data Fitting

In the laboratory, there are three major tasks which accompany most experiments. They are **calibration**, **verification**, and **characterization**. All of these tasks involve reducing the data we collect and meticulously characterize with uncertainty to a set of parameters. The purposes of the three tasks are different with respect to the goals of the data analysis.

A calibration requires that you try to find a relation between physical quantities and numerical output so that if you get an output, you can determine the physical quantity with confidence.

A verification requires that you test a model against your data and check whether the model appropriately describes the physical measurements you have made. In this case you must find a way of determining whether a model is invalid.

A characterization requires that you use a model to measure a characteristic of a phenomena or thing through the model's parameters. In this case you must find a way to quote this characteristic precisely with confidence and ensure the model is valid for your data.

4.1 The Method of Maximum Likelihood

Most fitters work in a conceptually simple way. They attempt to minimize or maximize some metric of fit goodness through a set of model parameters. At the extrema, the model parameters identify a curve of best fit.

For least-squares fitting, the fitter will try to find the least sum of squared deviation

from the fit curve. The function that it would minimize is

$$L(\mathbf{A}) = \sum_{i=1}^{N} (f(\mathbf{A}, x_i) - y_i)^2, \tag{4.1}$$

where $\mathbf{A} = (a_1, a_2, \ldots, a_M)$ is a vector of M parameters for the model $f(\mathbf{A}, x)$, and (x_i, y_i) are the N collected data points. So, now you know how a least squares fitter works, in principle. As it turns out, least-squares is a reasonable (but not great) metric for fitting, but why is this so?

The method of maximum likelihood provides a solid foundation for choosing an appropriate metric function for your data [2, p.180]. Suppose that you have a set of data, and you know what the PDFs are for every single data point, and you wish to fit this data to a curve. What exactly do you want the fit to indicate or accomplish? It is reasonable to suggest that the most probable curve is the best curve.

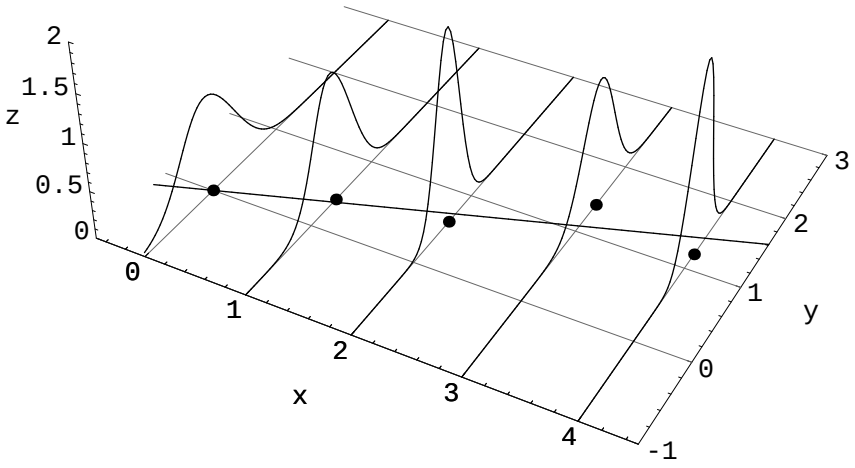

Figure 4.1: An illustration of maximum likelihood.

As illustrated in Fig. 4.1, for each x_i, the fit curve has a corresponding $f(\mathbf{A}, x_i)$, which which in turn has a corresponding probability density with respect to the data point (x_i, y_i). The total likelihood of the curve is the product of all the probability densities for each data point evaluated where the fit curve intersects x_i,

$$\mathcal{L}(\mathbf{A}) = \prod_{i=1}^{N} P_i(f(\mathbf{A}, x_i)). \tag{4.2}$$

Notice that this likelihood function isn't restricted to any kind of PDF function. Each data point could have different statistics and it would work just as well. It is a very general method of judging the goodness of fit. Typically, the logarithm of the likelihood is taken instead of just the likelihood because the likelihood becomes vanishingly small for large numbers of data points. As well, probability densities are normalized so that each peak value is 1 and a perfect fit corresponds to a log likelihood of 0. The result is negated and multiplied by 2 so that maximum likelihood corresponds to a minimization of a metric. The result is

$$L(\mathbf{A}) = -2\log(\bar{\mathcal{L}}(\mathbf{A})) = -2\log\Big(\prod_{i=1}^{N}\bar{P}_i(f(\mathbf{A},x_i))\Big) \tag{4.3}$$

$$L(\mathbf{A}) = \sum_{i=1}^{N} -2\log(\bar{P}_i(f(\mathbf{A},x_i))). \tag{4.4}$$

Note that the scripted $\mathcal{L}(\mathbf{A})$ is used for likelihood, $\bar{\mathcal{L}}(\mathbf{A})$ is used for peak normalized likelihood and $L(\mathbf{A})$ is used for metric.

4.1.1 Metric for Gaussian Distribution

If we assume that all data points exhibit Gaussian error statistics, then we arrive at a nice form of the fitting metric. Using Eq. (4.4), and substituting the peak normalized Gaussian PDF for P_i, we find

$$L(\mathbf{A}) = \sum_{i=1}^{N} -2\log\Big(\exp\Big(\frac{-(f(\mathbf{A},x_i)-y_i)^2}{2\sigma_i^2}\Big)\Big) \tag{4.5}$$

$$L(\mathbf{A}) = \sum_{i=1}^{N} \frac{(f(\mathbf{A},x_i)-y_i)^2}{\sigma_i^2} \tag{4.6}$$

$$= \chi^2(\mathbf{A}). \tag{4.7}$$

So the metric that should be minimized for a fit with gaussian distributed errors is the χ^2.

4.1.2 Metric for Poisson Distribution

We can arrive at a metric for data with Poisson error statistics by following the same recipe as above. One problem is that the maximum of the Poisson distribution cannot be solved analytically, so the normalization is difficult. However, we know that the

Poisson distribution approaches the Gaussian distribution in the limit of large μ, so we can estimate the maximum probability density with

$$P_{\max} \approx \frac{1}{\sqrt{2\pi\mu}} \qquad \mu \gg 1. \tag{4.8}$$

However, for small μ, there must a correcting factor which approximates the deviation. The following equation was found to be a good approximation to within 0.1%

$$P_{\max} \approx \frac{1}{\sqrt{2\pi\mu}} + \frac{1}{60}\mu^{-3/2}. \tag{4.9}$$

So, substituting the peak normalized poisson distribution into Eq. (4.4) we find,

$$\begin{aligned} L(\mathbf{A}) &\simeq \sum_{i=1}^{N} -2\log\left[\frac{e^{y_i} y_i^{f(\mathbf{A},x_i)}}{\Gamma(f(\mathbf{A},x_i)+1)(\frac{1}{\sqrt{2\pi y_i}} + \frac{1}{60}y_i^{-3/2})}\right] & (4.10)\\ &\simeq 2\sum_{i=1}^{N}\Bigg[\log(\Gamma(f(\mathbf{A},x_i)+1)) + \log\left(\frac{1}{\sqrt{2\pi y_i}} + \frac{1}{60}y_i^{-3/2}\right) \\ &\qquad - y_i - f(\mathbf{A},x_i)\log(y_i)\Bigg]. & (4.11)\end{aligned}$$

If $y_i \gg 1$, then the metric approaches χ^2 where all the uncertainties are $\sigma_i = \sqrt{y_i}$.

4.1.3 Metric for Lorentzian Distribution

The metric for Lorentzian error statistics is calculated in the same way as before. The Lorentzian distribution is easily normalized, and the metric is found to be

$$\begin{aligned} L(\mathbf{A}) &= \sum_{i=1}^{N} -2\log\left(\frac{(\Gamma/2)^2}{(f(\mathbf{A},x_i)-y_i)^2 + (\Gamma/2)^2}\right) & (4.12)\\ &= 2\sum_{i=1}^{N}\Big[\log((f(\mathbf{A},x_i)-y_i)^2 + (\Gamma/2)^2) - \log((\Gamma/2)^2)\Big]. & (4.13)\end{aligned}$$

4.2 Calculating Uncertainty in Parameters

Presumably, we have all the tools needed to generate a fit curve to data. We'd certainly be able to come up with the parameter vector $\mathbf{A}$ if we had a set of (x_i, y_i) and a model function $f(\mathbf{A}, x)$ by minimizing one of the metrics. Furthermore, a

minimum must exist since all the metrics are bounded from below by 0. However, parameters are not meaningful unless we can quote an uncertainty to them.

Calculating uncertainty in parameters can be quite difficult, particularly if the model function is nonlinear and has many parameters. Certain assumptions are made about the behavior of parameters which simplify the analysis. First, all functions can be approximated as linear in their parameters over small deviations of the parameters. Second, all parameters are uncorrelated and independent.

A detailed derivation of the following results are available in the cited references [2, p.122]. The covariances of the parameters are related to the metric through the matrix equation

$$\boldsymbol{\epsilon} = \boldsymbol{\alpha}^{-1} \quad \text{where} \quad \epsilon_{ij} = \sigma_{ij}^2, \quad \alpha_{ij} = \frac{1}{2} \left. \frac{\partial^2 L}{\partial a_i \partial a_j} \right|_{\mathbf{A}_0}, \tag{4.14}$$

L is the metric, $\mathbf{A}_0$ are the parameters of minimum L, and $i, j \in [1, M]$ where M is the number of parameters in the model. The matrix $\boldsymbol{\alpha}$ is known as the curvature matrix as it describes the curvature of the metric surface along every pair of coordinates in parameter space. The matrix $\boldsymbol{\epsilon}$ is known as the error matrix since it contains estimates of uncertainty for all the parameters.

Fortunately, finding a precise approximation of the second derivatives of the metric is easy using finite differencing, and matrix inversion is a simple task given a computer. So, the error matrix can be determined without much effort.

To get an idea of what we are finding from the curvature matrix, we consider an idealized case of perfectly uncorrelated parameters. If perfectly uncorrelated, we expect that the covariances of the parameters are all zero. So, our error matrix is diagonal in variances. Thus, the curvature matrix must also be diagonal and we have the set of equations,

$$\frac{1}{2} \left. \frac{\partial^2 L}{\partial a_i^2} \right|_{\mathbf{A}_0} = \frac{1}{\sigma_i^2}, \quad i = 1 \ldots M \tag{4.15}$$

where σ_i^2 is the variance of parameter a_i, not the uncertainty of data point y_i.

Since we are evaluating a function near a minimum, we can approximate the function as parabola for small perturbations. With the curvatures given in the above equations, and integrating twice we find that

$$L(a_{1_0}, \ldots, a_i, \ldots, a_{M_0}) = \frac{(a_i - a_{i_0})^2}{\sigma_i^2} + L(\mathbf{A}_0). \tag{4.16}$$

A convention for the error in a parameter is the change in the parameter that increases the χ^2 by 1 from the minimum. Clearly, if we substitute $|a_i - a_{i_0}| = \sigma_i$, we find that the metric increases by 1.

4.3 The Quality of a Fit

The goodness of a fit is a term used to describe the operation of a fitter, the quality of a fit is a term used to describe how well the parameters model the data, and whether you can state confidence in your results.

There are several basic checks for the quality of a fit. The first is to check if the minimum metric you found was an appropriate size. For Gaussian statistics, a general rule is that

$$\frac{\chi^2(\mathbf{A}_0)}{N-M} \approx 1, \tag{4.17}$$

where N is the number of data points and M is the number of parameters. The same rule applies to Poisson data as long as most of the data points correspond to more than 10 counts.

You can also check the χ^2 cumulative distribution function for your value of χ^2. The cumulative distribution function tells you the probability that if you repeat the experiment, that you will get a higher χ^2. If your fit is high quality, then you'd expect that there would be about an equal chance of getting a higher χ^2 versus a lower χ^2. If you've artificially inflated your errors, then you'd expect to get a higher χ^2 much more often. Conversely, if your model doesn't fit, you'd expect to get a lower χ^2 every time. If the χ^2 cumulative distribution function is within $[5\%, 95\%]$, the fit is usually deemed to be acceptable, unless there are other reaons why it is may be suspect. One note is that the χ^2 test doesn't tell you if your model is correct, it can only tell you if it wrong. For example, you may fit data to a quadratic and it would have acceptable χ^2, but it may also be acceptable for a fit to an exponential function.

The second check is to look at the residuals. If you see a clear pattern in the residuals, then you may need to reconsider your model or check for systematic errors. Typically, if there is a striking pattern in the residuals, the metric will not be an acceptable value. If you find yourself with a pattern in the residuals, but with an acceptable metric, you have likely overestimated your errors because it is quite unlikely that the pattern in the residuals is coincidental. For example, if you fit a thermocouple calibration curve, nonlinear terms will likely be present, and they will be visible in the residuals of the fit curve. The residuals of a fit should be randomly distributed about zero with $\sim 70\%$ of the data points within quoted uncertainty of zero.

A third check is to look at how the fit parameters change as you fit different sections of the data set. If your model accurately describes the data over the entire range, then the fit parameters you obtain for different sections of the data set should all agree within uncertainties. This technique is particularly effective at investigating errors arising from instrument behavior.

5 The Advanced Physics Lab Fitter

5.1 Introduction and Principles of Operation

The advanced physics lab fitter has been based upon the principles in the previous sections. So, in theory, anyone who has a grasp of the material can build their own fitter, or better yet, help to improve the operation of the current fitter. The fitter operates in MATLAB and has a graphical user interface to shield the user from the tedium of handling command line options.

The fitter works under the principle that a set of measurements should be organized by variable. So, if we take a set of measurements of temperature and voltage, the voltage measurements can be given a name, say `volt` and the temperature measurement can be given a name, say `temp`. This is opposed to an analysis system that works under the principle that measurements should be organized by data point.

Every set of data must be represented in the workspace as an $N \times 2$ matrix. By convention, the first column is **always** the measurement and the second column is always the uncertainty. This is consistent with the idea that a measurement is not meaningful without an uncertainty. Another note is that to be able to fit or plot two variables, they must have the same number of rows.

So, suppose we perform an experiment and collect 3 data points of temperature and voltage, $(280 \pm 1\text{ K}, -0.45 \pm 0.03\text{ mV})$, $(290 \pm 1\text{ K}, 0.04 \pm 0.06\text{ mV})$, and $(300 \pm 1\text{ K}, 0.66 \pm 0.07\text{ mV})$.

These data should be defined in your workspace with two matrices,

$$\texttt{temp} = \begin{pmatrix} 280 & 1 \\ 290 & 1 \\ 300 & 1 \end{pmatrix} \text{ and } \texttt{volt} = \begin{pmatrix} -0.45 & 0.03 \\ 0.04 & 0.06 \\ 0.66 & 0.07 \end{pmatrix}$$

The software is broken into two parts that work independently. There is the **Data Preview and Browser** and the **Fitting Tool**. The Data Preview and Browser is executed with the command `databrowser`. The Fitting Tool is executed with the command `fitter(x,y)`, where `x` and `y` are the data corresponding to the x and y axes respectively.

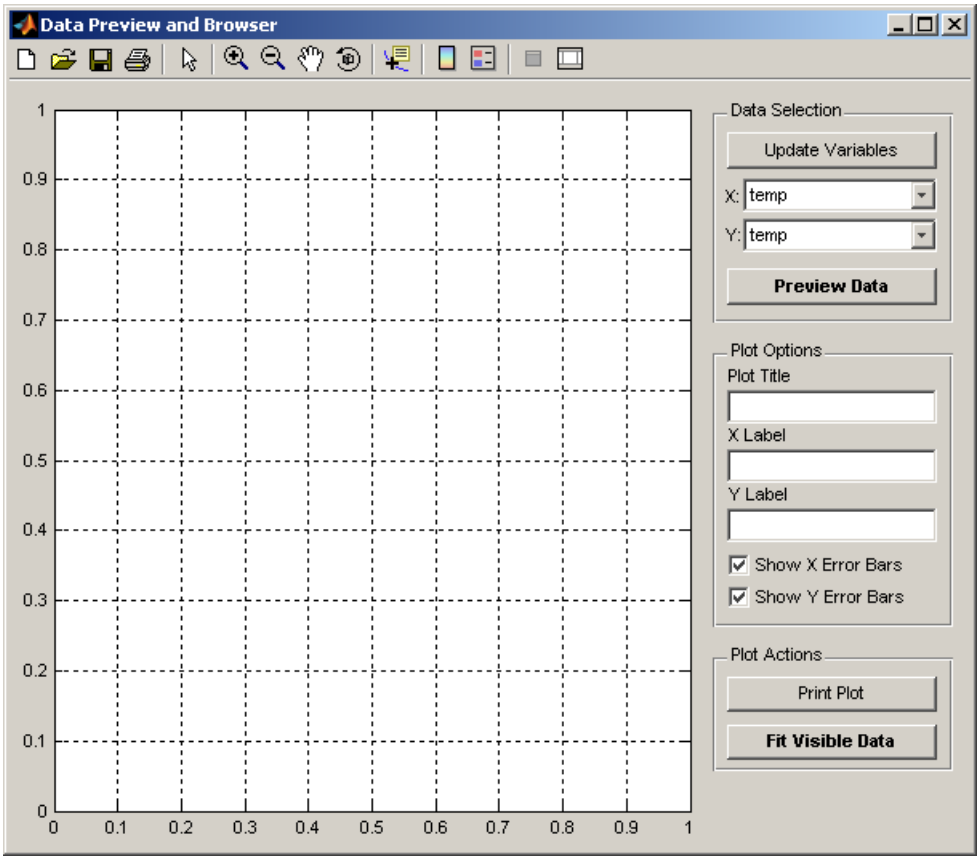

Figure 5.1: The Data Preview and Browser.

5.2 Data Preview and Browser

Upon running the command `databrowser`, you are presented with the interface shown in Fig. 5.1. There is one large pair of plot axes, three panels of buttons and edit boxes, and a toolbar at the top of the window.

The **Data Selection** panel allows you to select data from your workspace and using the dropdown menus and plot them against each other using the **Preview Data** button.

The **Plot Options** panel allows you to change the appearance of the plot. You can add labels to the axes and a title using the edit boxes and show or hide the error bars with the checkboxes. You will need to use the **Preview Data** button to show changes in error bar visibility.

The **Plot Actions** panel allows you to print or fit the data on the plot. To print a formatted plot which will fit nicely into a standard black lab book, click the **Print**

Plot button. Do not use the print button in the toolbar. You must have a preview of your data visible to print. To fit all of the data points that are visible within the limits of the preview plot, click the **Fit Visible Data** button and an instance of the **Fitting Tool** will be opened with the visible data.

Data can be selected for fitting by changing the view of the data. This can be accomplished by using the **zoom** and **pan** tools from the toolbar. The **zoom in** tool is activated by clicking on the button with the positive magnifying glass. The **zoom out** tool is activated by clicking the button with the negative magnifying glass. **pan** tool is activated by clicking the button with the white hand. All tools can be deactivated by clicking on their respective buttons a second time (they toggle). They are used by clicking and dragging on the plot.

A sample of a labelled preview plot is shown in Fig. 5.2.

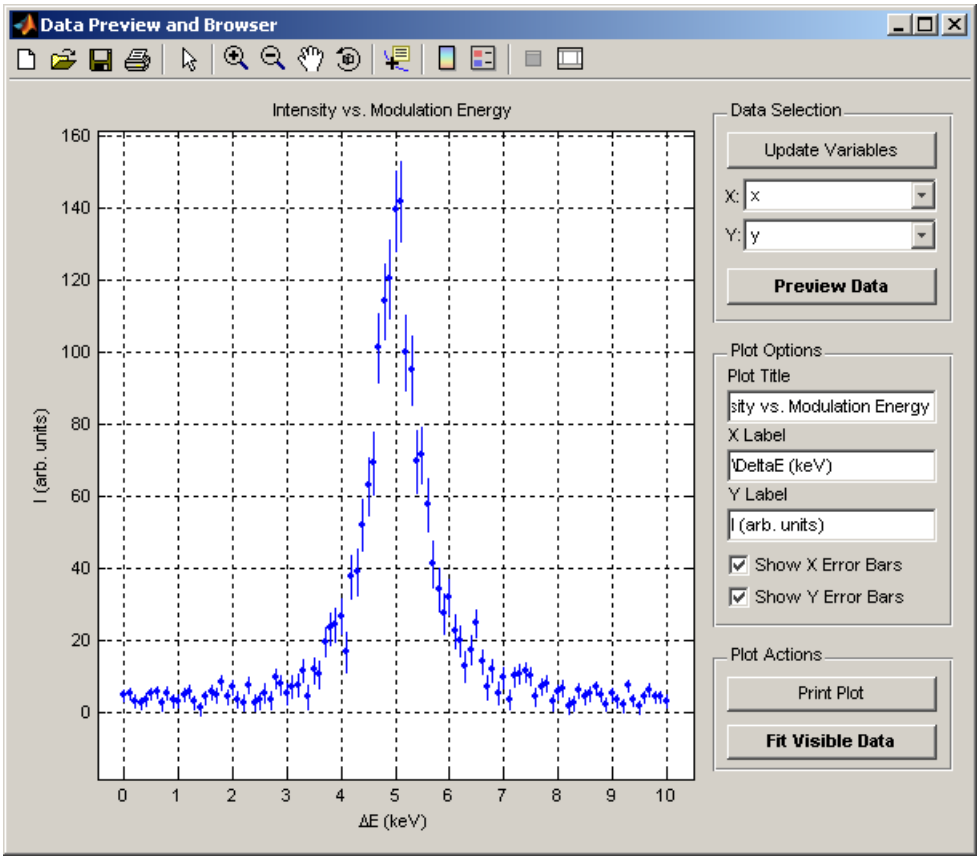

Figure 5.2: Sample labelled data preview.

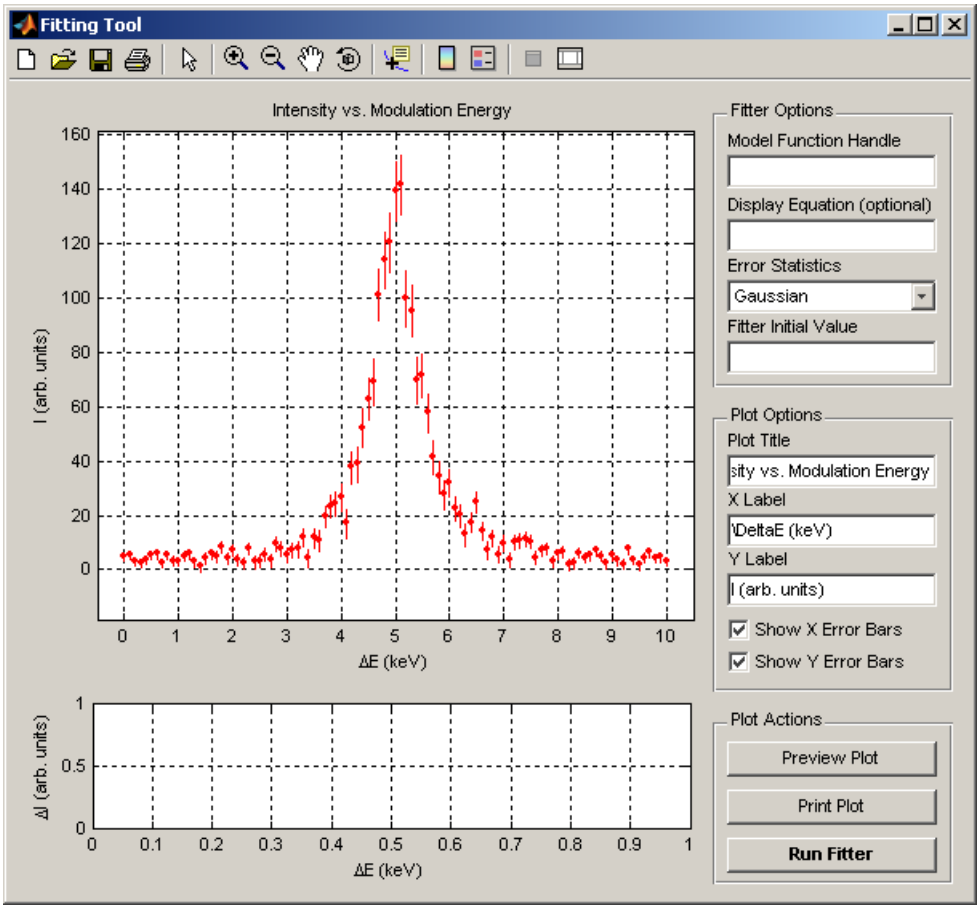

Figure 5.3: The Fitting Tool.

5.3 Fitting Tool

Upon running the command `fitter(x,y)` or clicking the **Fit Visible Data** button in the Data Preview and Browser, you are presented with the interface shown in Fig. 5.3. There are two pairs of plot axes, three panels of buttons and edit boxes, and a toolbar at the top of the window.

The large top axes displays a plot of the data, the fit curve, and whatever plot annotation is generated. The smaller bottom axes displays a plot of the fit residuals if they are available. Upon opening the Fitting Tool, there are no residuals since no fit has been attempted.

The **Fitter Options** panel allows you to change the important parameters of the fitter. The **Model Function Handle** edit box requires the name of the model you wish to fit. Each function is specified by a unique string such as "linear" or

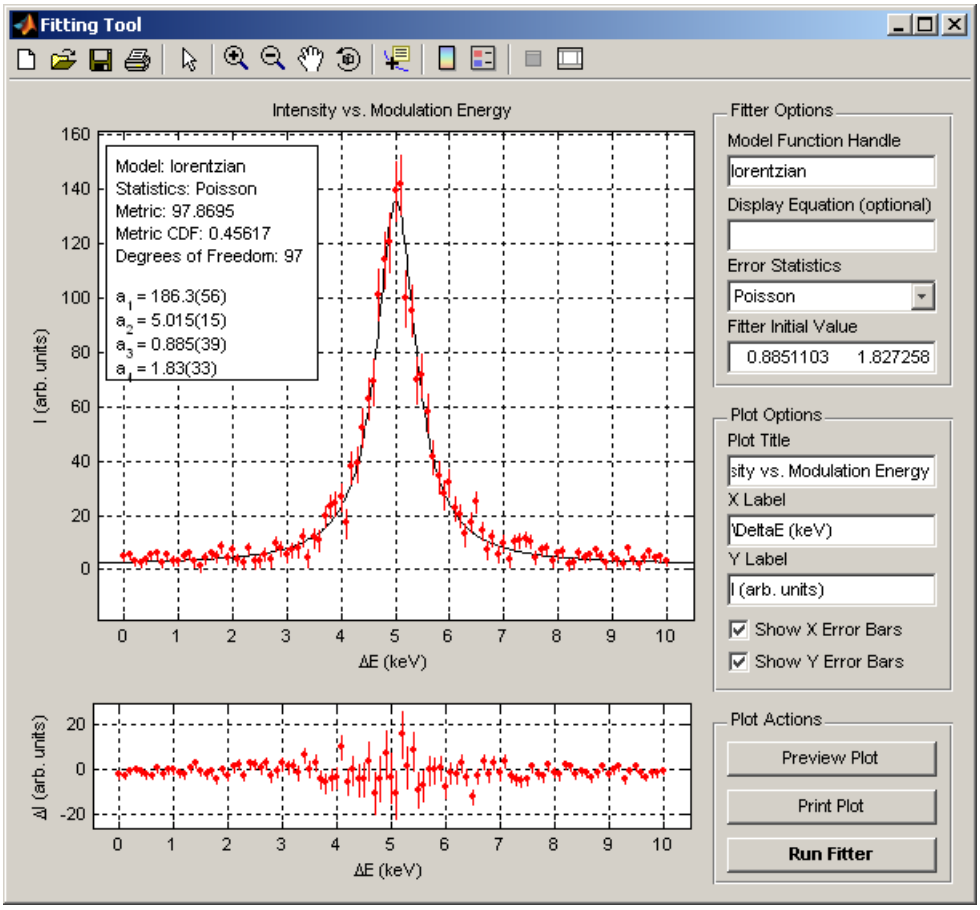

Figure 5.4: Successful fit using the Fitting Tool.

"quadratic". A whole library of functions is available and their specifications are outlined in Appendix B. The **Display Equation** edit box can be filled with a LaTeX equation such as "y = a_1 + a_2 x" which is displayed on the plot as an annotation after a fit has been made. This field is optional. The **Error Statistics** popup menu allow you to select the kind of probability distribution function that best describes the uncertainties in all the data points. The options are Gaussian, Poisson, Lorentzian, and None (blind least squares). The **Fitter Initial Value** edit box contains a series of numbers that tells the fitter where in parameter space to start its search. The input corresponds to an initial value of parameter vector **A** specified by a list of numbers separated by either spaces or commas with the i^{th} number corresponding to a_i.

The **Plot Options** panel allows you to change the appearance of the plot with the same options as the Data Preview and Browser.

The **Plot Actions** panel allows you to preview fits, print plots, and execute the fitter. The **Preview Fit** button will calculate and display a curve generated from

the fitter initial values. It will also calculate the residuals to the data points and the metric with the given parameters. This feature is useful in finding a good initial value for the fitter or to manually perform a fit. The **Print Plot** button will generate a nicely formatted plot, which will fit nicely into a black lab notebook. Do not use the print button in the toolbar since it will print the panels and buttons, which are unpleasant. The **Run Fitter** button will execute the fitting routine with the data that has been provided through the user interface. While fitting, the action buttons will be greyed out and disabled. The buttons are enabled once the fitter is complete. If there is an error, the fitter will try its best to recover and tell you what went wrong.

A sample of a successful fit using the Fitting Tool is shown in Fig. 5.4.

5.4 Your First Fit: A Walkthrough

To get you up and running, we have a detailed walkthrough of a typical calibration fit. The data was taken from Example 7.1 of Bevington [2, p.119-121]. As quoted from the text,

> A student measures the difference in output voltage between the two junctions of a thermocouple for a temperature variation in the sample junction from 0 to 100°C in steps of 5°C. The measurements are made on the 3 mV scale of an analog voltmeter, and the fluctuations of the needle indicate that the uncertainties in the measurements are approximately 0.05 mV for all readings.

There are no quoted errors for temperature. By document convention, all command code is displayed in fixed width font surrounded by a box. For example:

```
a = a.*b + inv(c) - d + meshgrid([0:0.01:0.4],[-3:-0.01:-3.4]);
```

Step 1: Get your data into MATLAB.

You will first need to get the data from your lab book into the computer. The easiest way to do this is to create text files corresponding to each variable with filenames indicating the variable name.

In Fig. 5.5, we have used notepad to create two text files each with two columns of numbers corresponding to measurement and error. The measurement is **always** in the first column. You can use spaces, tabs, a single comma, or a combination of these delimiters to separate the columns.

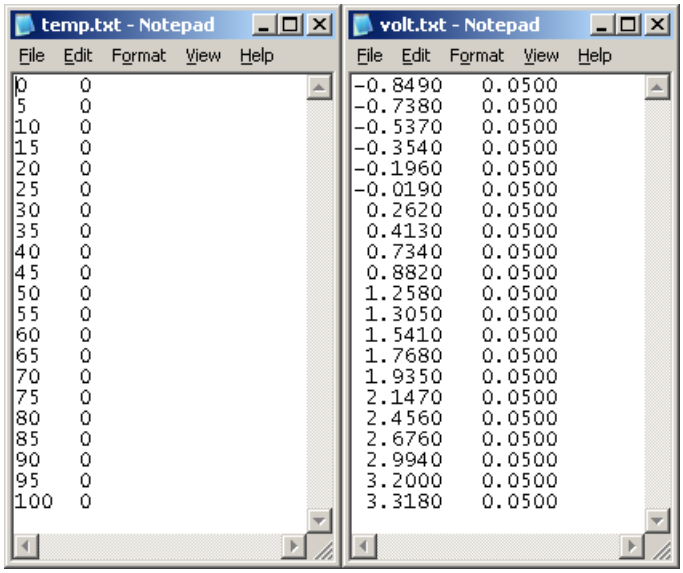

Figure 5.5: Creating text files for data variables.

Save these files into your MATLAB working directory. If in doubt, place them in `C:\MATLAB7\work\`, which is the default working directory. It is convenient to organize your work by experiment, so a directory per experiment is good practice, although not necessary.

Now, launch MATLAB if it is not already open. MATLAB can automatically read and parse the text files you've created, so type the commands:

```
load temp.txt
load volt.txt
```

These commands will create two variables in your workspace called `temp` and `volt` with the data that you have entered. To save the workspace as one file so that you can easily recall all your data, type the command:

```
save thermocouple.mat
```

This command will save a workspace file containing all of your workspace variables to your working directory with the filename `thermocouple.mat`. If you close MATLAB and start again, you can recall your saved workspace with the command:

```
load thermocouple.mat
```

Step 2: Select your data to fit using the Data Preview and Browser[4].

With variables in your workspace, you can select what data you wish to fit with the Data Preview and Browser. To open this tool, type the command:

```
databrowser
```

This will bring up a window as shown in Fig. 5.1. Select your X variable and Y variable from the two popup menus. In this example, select `temp` as the X variable and `volt` as the Y variable. Preview your data by clicking the **Preview Data** button. You must preview your data to be able to fit it, since the browser selects visible data points to send to the fitter. You can also label the data with a title and axes labels.

You can use the **zoom** and **pan** tools in the toolbar to change the plot view. This is how you select data to fit. If you want to fit the whole data set, this isn't necessary as the axes limits are chosen to suit the data set.

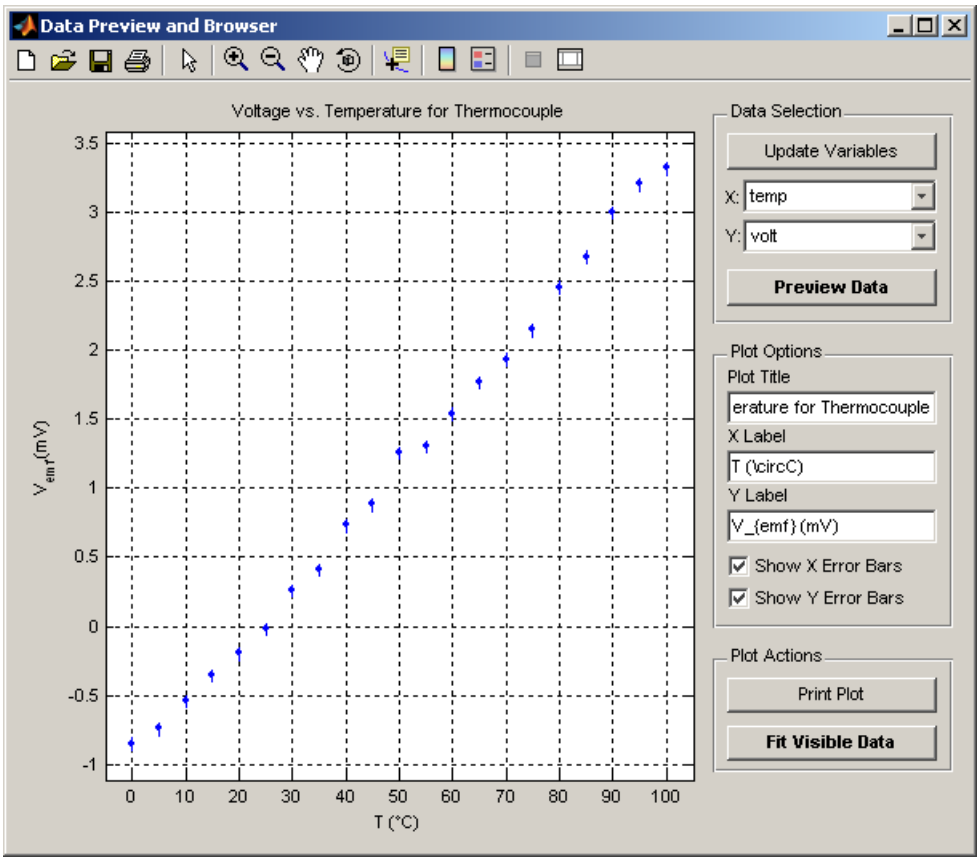

Figure 5.6: Previewed and labelled data.

[4]You can skip this step by running the command `fitter(temp,volt)` if you're sure you want to fit the entire data set

A labelled data set and preview plot is shown in Fig. 5.6. Once you're happy with the visible data, click the **Fit Visible Data** button to open the Fitting Tool. Titles and axes labels are also sent to the fitter, so that you don't need to type them in again.

Step 3: Fit your data using the Fitting Tool.

The Fitting Tool will be opened by the Data Preview and Browser. If you had clicked the **Fit Visible Data** button in Fig. 5.6, a new window will open as shown in Fig. 5.7.

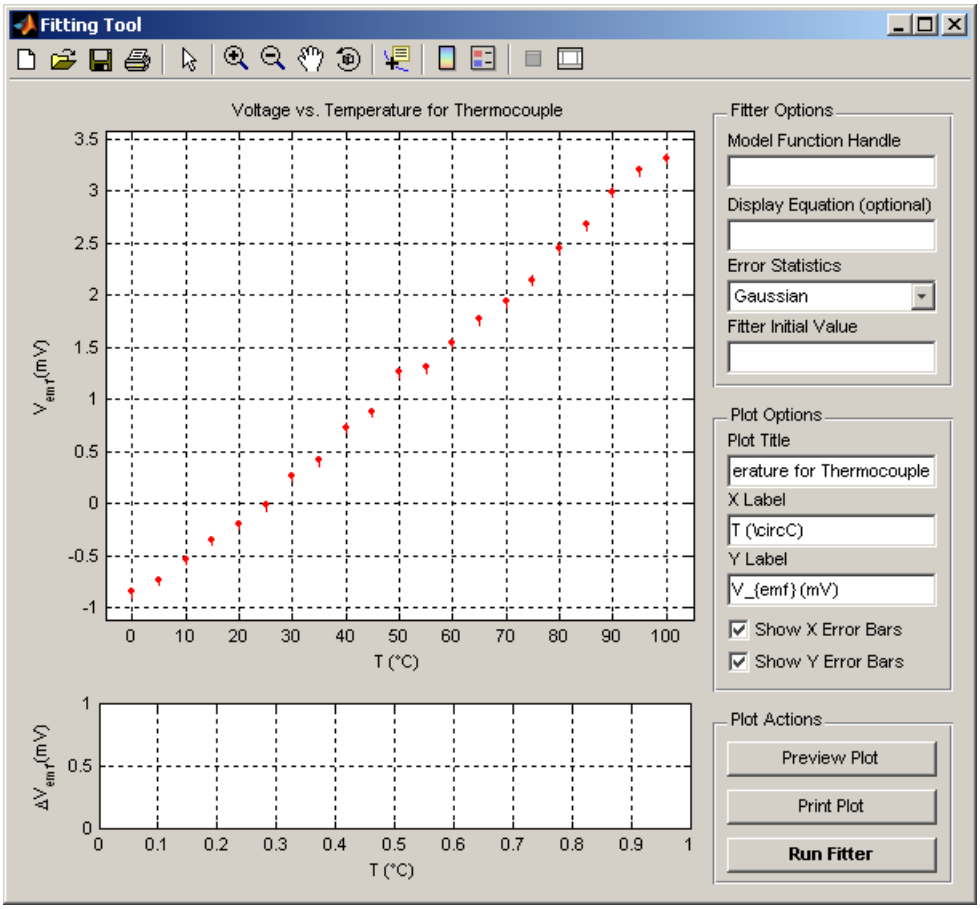

Figure 5.7: Initial fitting tool window.

To select a fit equation, refer to the standard function library in Appendix B. Each equation is identified by a string called a function handle. Enter your equation function handle into the **Model Function Handle** edit box. For this example, we will use the "quadratic" equation.

You may optionally add a LaTeX style expression for the equation which will be displayed on the plot if you wish. Enter the expression into the **Display**

Equation edit box.

Select the data statistics with the **Error Statistics** popup menu. Most lab data is assumed to have a Gaussian distribution, only choose other statistics if you know that they are justified.

You must specify initial values for each parameter so that the fitter knows where to start its search. Enter initial values as a single line of space or comma separated numbers corresponding to each parameter $a_1 \ldots a_M$ into the **Fitter Initial Value** edit box. You can preview your initial values by clicking the **Preview Plot** button. A preview plot with initial values $(a_1, a_2, a_3) = (-1, 0.03, 0.0002)$ is shown in Fig. 5.8

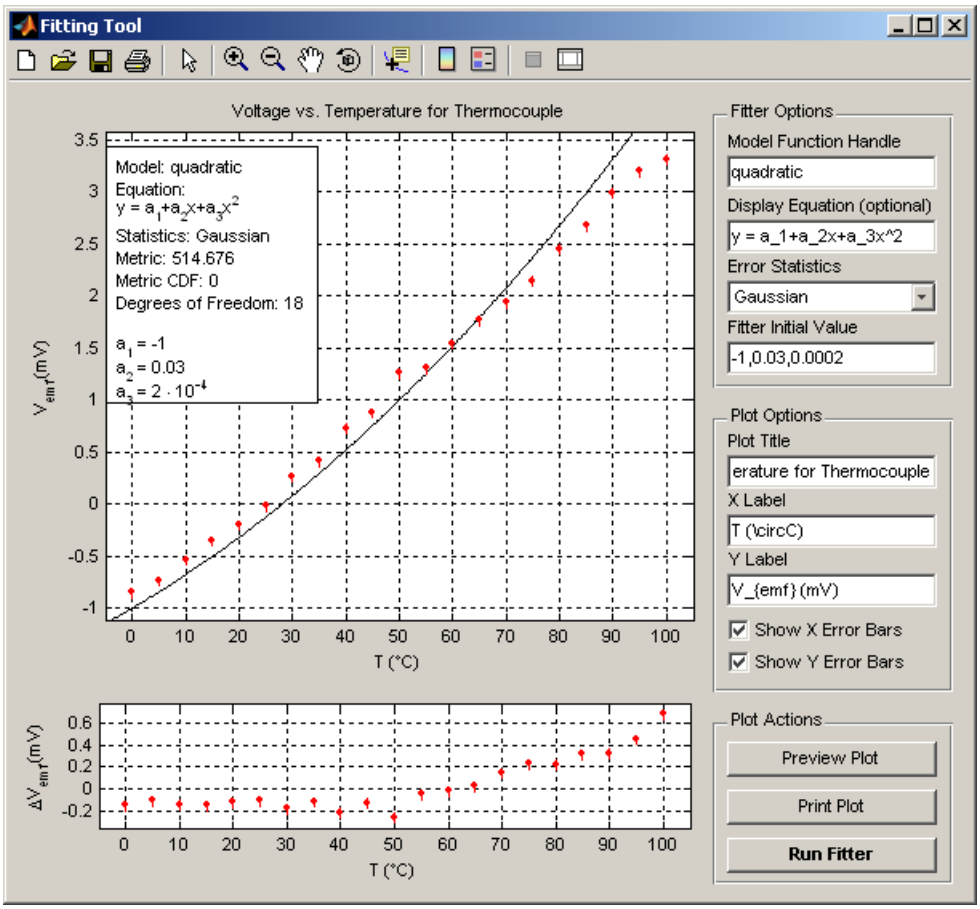

Figure 5.8: Preview of a quadratic fit curve with $[a_1, a_2, a_3] = [-1, 0.03, 0.0002]$.

To run the fitter, click the **Run Fitter** button. It will run the Levenberg-Marquardt gradient search algorithm through parameter space to find a minimum value of the metric. The results are presented on the plot as soon as the fitter completes executing. The best fit plot of this data is shown in Fig. 5.9.

Parameter errors are estimated and displayed with the parameters in the annotation box. Residuals are shown in the lower set of axes.

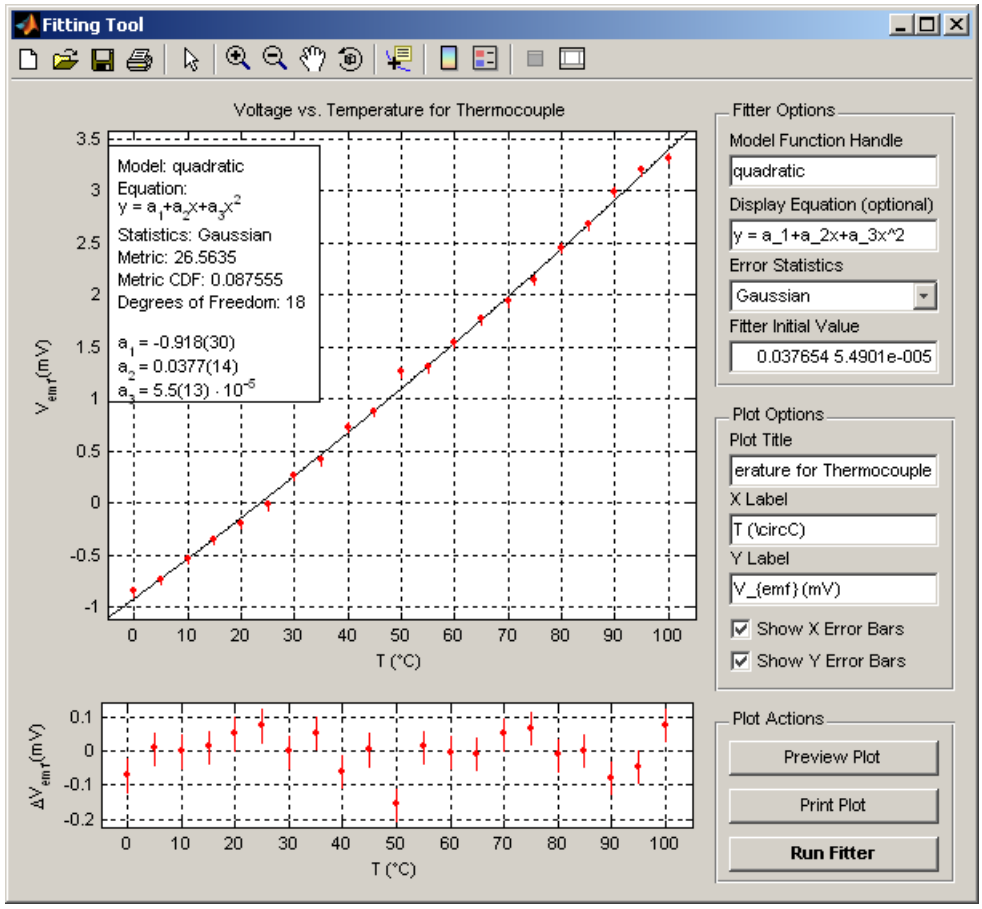

Figure 5.9: Best fit of thermocouple data.

If you want to print the fit, click the **Print Plot** button and it will open a print dialog and format a nice plot that will fit into your lab book.

5.5 Fitting Arbitrary Functions: A Walkthrough

The power of this fitting environment is that you can define your own fitting functions, which can be as simple or complex as you like. The fitter will theoretically work with functions with as many parameters that you wish to throw at it, and can generate a fit as long as you can create a MATLAB function that satisfies the basic requirement that if you give it a parameter vector, and data vector it will return a vector of identical dimension of the data vector with the results of the function.

For example, if you want to fit to a curve,

$$y = f(\mathbf{A}, x) = a_1 x^2 \sin(a_2 x) + a_3 \tag{5.1}$$

you must create a MATLAB function that will perform the operation,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} f(\mathbf{A}, x_1) \\ f(\mathbf{A}, x_2) \\ \vdots \\ f(\mathbf{A}, x_N) \end{pmatrix}. \tag{5.2}$$

Step 1: Write your mathematical function as a MATLAB function.

MATLAB functions are defined with M-files, which are collections of MATLAB commands with a function wrapper. For example, consider the M-file for the quadratic model.

```
function y = quadratic(a,x)
y = a(1) + a(2)*x + a(3)*x.^2;
```

The function is in the familiar $f(\mathbf{A}, x)$ format, where `a` is the parameter vector and `x` is the data vector.

The only issue in writing your mathematical function in MATLAB is when MATLAB tries to use its matrix operations. Consider the following commands:

```
y = (x + 1) / x;
y = (x + 1) * x;
y = (x + 1)^2;
```

At first glance you may think these are okay, but there is a problem. MATLAB thinks that `x` is a matrix, so what does the first line really mean?

The first line means: Take matrix $\mathbf{X}$ and add 1 to every element, then matrix multiply the result with $\mathbf{X}^{-1}$.

The second line means: Take matrix $\mathbf{X}$ and add 1 to every element, then matrix multiply the result with $\mathbf{X}$.

The third line means: Take matrix $\mathbf{X}$ and add 1 to every element, then matrix multiply the result with itself (matrix exponent).

Not quite what you expected? To get the desired behavior, there are MATLAB operators that work element-by-element. To make an operation element-by-element, prefix the operator with a period. i.e. (`x*y` → `x.*y`), (`x/y` → `x./y`), (`x^2` → `x.^2`). These operations calculate $z_{ij} = x_{ij} y_{ij}$, $z_{ij} = x_{ij}/y_{ij}$, and $z_{ij} = {x_{ij}}^2$, respectively.

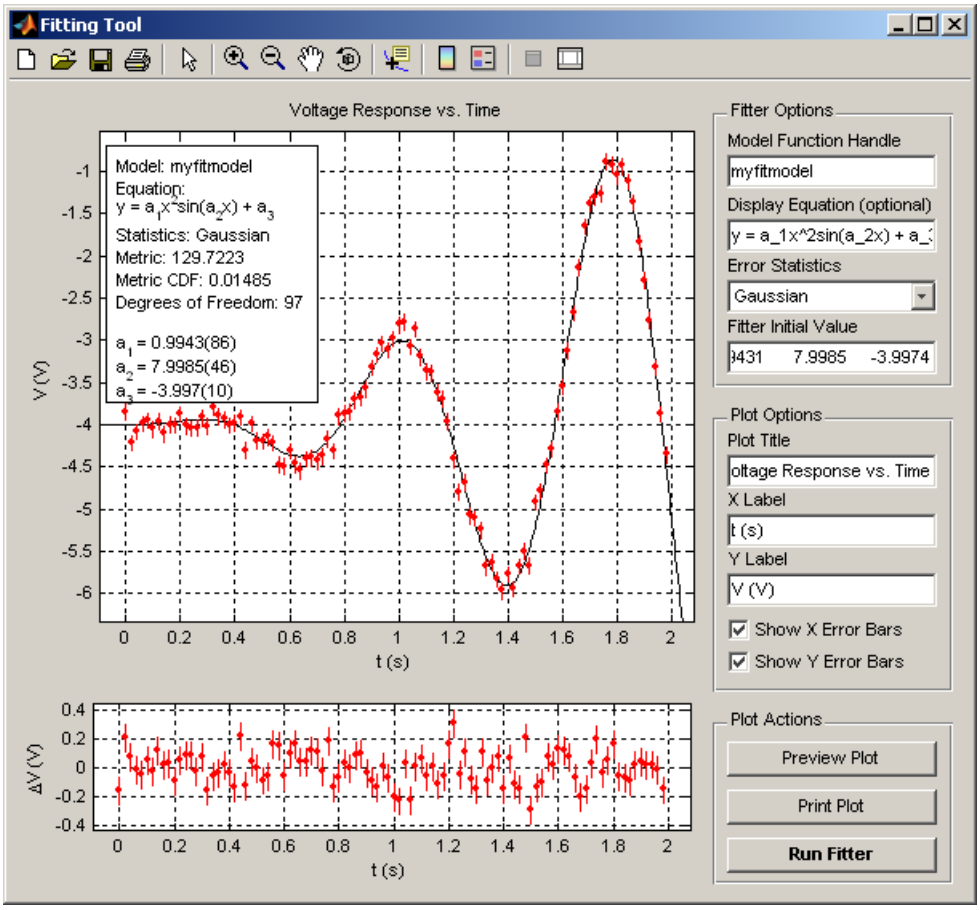

Figure 5.10: Sample fit using a custom function.

If one of the operands is a 1×1 matrix, it is treated as scalar, so isn't necessary to use the element-by-element operators. The following commands require no modification:

```
y = x + 1
y = x + a(3);
y = a(4) * x;
```

MATLAB's built in functions work element-by-element, so `sin(x)` calculates $z_{ij} = \sin(x_{ij})$. The same applies to basic functions like `cos`, `exp`, `log`, `besselk`, etc.

So, for our example,

$$y = f(\mathbf{A}, x) = a_1 x^2 \sin(a_2 x) + a_3 \tag{5.3}$$

the equivalent MATLAB function would be:

```
function y = myfitmodel(a,x)
y = a(1) * x.^2 .* sin(a(2)*x) + a(3);
```

Be sure to remember to finish lines with semicolons, otherwise you're going to get **a huge mess** in your command window because the fitter will run this function several thousand times, and every time the function runs, MATLAB will display the result matrix.

Save your function as `<function name>.m` in the working directory to give the fitter access to your function. Your function's function handle is simply the function name. So, we would save this example as `myfitmodel.m` and the function handle is "myfitmodel".

Step 2: Use your function in the fitter.

Once you've created your custom function, use it like any library function by using the appropriate function handle in the Fitting Tool. A fit of the example function in Step 1 with some fabricated data with Gaussian error is shown in Fig. 5.10.

A An Introduction to MATLAB

MATLAB is an abbreviation for **Mat**rix **Lab**oratory. It was originally written as an "easy to use" extension of the LINPACK and EISPACK linear equation and eigenvalue solving systems by Cleve Moler. Now it is one of the most popular technical computing environments in the world and is used throughout the sciences and engineering.

The basic philosophy of a MATLAB user is:

Everything is a matrix.

This is different than other technical computing environments such as *Mathematica* or *Maple* where everything is an expression, or you have to deal with a mish-mash of both data and expression. MATLAB is excellent for data analysis because its operation is based almost entirely in the manipulation of numerical data, not mathematical expressions.

The usage of MATLAB is based around the **workspace**. The workspace can be thought as local memory on your workstation where your data resides. When you load data from a file, or input it manually, the data resides in the workspaces as a **variable**. A variable is a matrix with some variable name that you can define that can be passed to functions or combined with other variables.

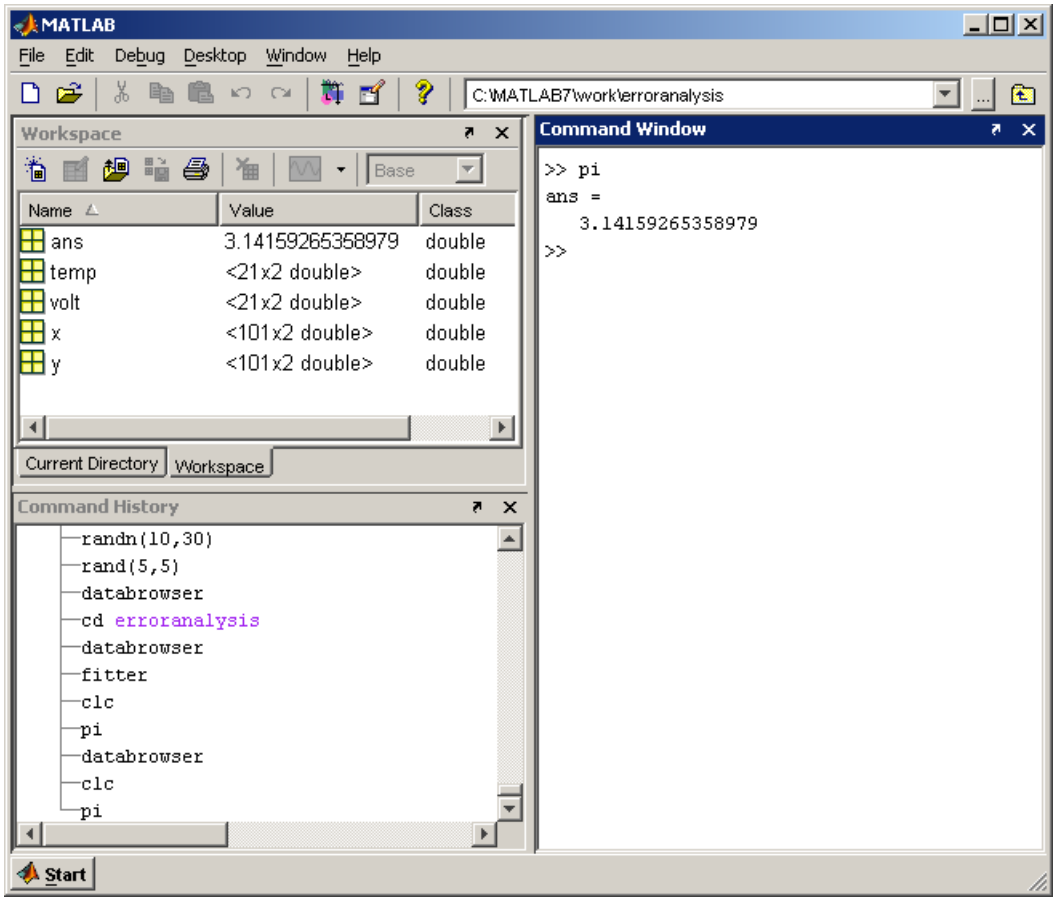

Figure A.1: The MATLAB desktop environment.

When you launch MATLAB you will be presented with an interface called the **desktop environment** as shown in Fig. A.1. It lets you look at what's in your workspace, shows your command history, and gives you access to the command line. You will need to use the command line to load data into your workspace and to perform manipulations on them.

We begin a simple example of the use of the command line. Commands and variables are case-sensitive (`volt` not the same as `VoLt`) and all commands must be followed by the enter key. Consider the following commands:

```
a = 1
b = [1, 2, 3]
c = [4, 5, 6]'
d = [1, 2, 3; 4, 5, 6]
```

The first line creates a new variable in the workspace called `a` and assigns it a 1×1

matrix with single element of value 1.

The second line creates a 1×3 row matrix called `b` with elements $\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$.

The third line has a matrix definition with an apostrophe. The apostrophe means transpose. So, the effect on this command is that we create a 3×1 column matrix called `c` with elements $\begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$.

The fourth line uses a semicolon in the matrix definition. This means "new row". So, the command creates a 2×3 matrix called `d` with elements $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$.

After running each command, MATLAB will display the result. If you want to make the commands "silent", follow the command with a semicolon, such as:

```
c = [4, 5, 6]';
```

It is also useful to know how to extract data from matrices. MATLAB has a clever operator to aid in extracting data called the colon (`:`) operator. The colon operator can be thought as defining a span of indices. If you use the colon operator alone, it means all indices, if you specify boundaries, it means all the values within that range of indices. Consider the following commands:

```
a = x(2,3)
b = x(:,1)
c = x(2:4,2:6)
```

Assume $\mathbf{X}$ is a 6×6 matrix.

The first line assigns the value of x_{23} (row 2, column 3) to variable `a`.

The second line assigns a vector of all rows of $\mathbf{X}$ at column 1 to variable `b`.

The third line assigns a matrix consisting of rows 2 to 4 and columns 2 to 6 to variable `c`.

Practical operations may involve removing offset from data or changing error due to systematics:

```
volt(:,1) = volt(:,1) - 2.30;
volt(:,2) = 2*volt(:,2) + 0.01;
```

B The Function Library

B.1 Polynomials

constant	$y = a_1$
proportional	$y = a_1 x$
linear	$y = a_1 + a_2 x$
quadratic	$y = a_1 + a_2 x + a_3 x^2$
cubic	$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3$
quartic	$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + a_5 x^4$
quintic	$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + a_5 x^4 + a_6 x^5$
poly6	$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + \ldots + a_7 x^6$
poly7	$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + \ldots + a_8 x^7$
poly8	$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + \ldots + a_9 x^8$
poly9	$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + \ldots + a_{10} x^9$

B.2 Exponentials and Logarithms

exponential	$y = a_1 \exp(a_2 x)$
transientpos	$y = \exp(a_1(x - a_2)) + a_3$
transientneg	$y = -\exp(a_1(x - a_2)) + a_3$
powerlaw	$y = x^{a_1}$
powerlawconst	$y = x^{a_1} + a_2$
logarithm	$y = a_1 \log(x + a_2) + a_3$

B.3 Peak and Background

gaussian	$y = a_1 \exp\Big(\frac{-(x-a_2)^2}{2a_3^2}\Big) + a_4$
gaussianlinear	$y = a_1 \exp\Big(\frac{-(x-a_2)^2}{2a_3^2}\Big) + a_4 + a_5 x$
gaussianquadratic	$y = a_1 \exp\Big(\frac{-(x-a_2)^2}{2a_3^2}\Big) + a_4 + a_5 x + a_6 x^2$
lorentzian	$y = \frac{a_1}{\pi} \frac{a_3/2}{(x-a_2)^2 + a_3^2/4} + a_4$
lorentzianlinear	$y = \frac{a_1}{\pi} \frac{a_3/2}{(x-a_2)^2 + a_3^2/4} + a_4 + a_5 x$
lorentzianquadratic	$y = \frac{a_1}{\pi} \frac{a_3/2}{(x-a_2)^2 + a_3^2/4} + a_4 + a_5 x + a_6 x^2$

Need to look through experiments to
find more functions to add to the function library.
Any suggestions?

References

[1] Alberto Leon-Garcia. *Probability and random processes for electrical engineering.* McGraw-Hill, second edition, 1994.

[2] Philip R. Bevington and D. Keith Robinson. *Data reduction and error analysis for the physical sciences.* McGraw-Hill, third edition, 2003.