Since I don't know of a good reference that lays out model-fitting etc. the way I like to think about it, I've started writing up notes. They are currently a work in progress and may be periodically be updated/extended.

The problem of "what model fits my data" is as old as experimental data [1]. An important category is when measurement errors have Gaussian distributions. Not only are Gaussians some of the easiest distributions to work with, but due to the central limit theorem, if we have enough data we'll end up at a Gaussian distribution anyways. The aim of this note is to show, starting from the probability distribution function (PDF) of a Gaussian, that we can compare the relative probabilities that different models would have given rise to our observed data.

1. From Gaussians to χ^2

If we have a Gaussian random variable with mean μ and standard deviation σ, then we want to know how likely we are that a realization of this variable gives us a value x. While the probability of a single value is zero for any continuous definition, we instead use the *probability density function* or PDF, where the probability of observing an event between x and $x+\delta_x$ is $\mathrm{PDF}(x)\delta_x$. Properly normalized PDFs should always be non-negative and integrate to unity. For a Gaussian, the PDF is:

$$\exp(-\frac{(x-\mu)^2}{2\sigma^2})/\sqrt{2\pi\sigma^2}$$

where the factor of $\sqrt{2\pi\sigma^2}$ is needed so the PDF integrates to one.

In traditional model fitting, we are usually interested in figuring out that the data "should have been," which mean we want to figure out μ. We also usually take the errors σ as given. While we can certainly try to estimate the errors at the same time as a model, but it's quite a bit more complicated and so we don't try to do that here. We also usually have many data points, which may have their own errors and expected means. One simple case would be fitting a line to data taken at different times t, where the expected value of each data point would be $at+b$. In this case, if we measure the i^{th} data point at time t_i, then we have $\mu_i = at_i + b$ for a slope a[2] and intercept b . Each data point would have its own value, taken at its own time, but if a line is a good description of the data, they should all agree on the values for the slope and intercept.

If we have many data points, with *uncorrelated* errors, then the probability density of observing several data values is just the product of the probability density of the individual data points. In otherwords, if d_i are our n data points, then the PDF is

$$\prod_{i=1}^{n} \exp(-\frac{(d_i-\mu_i)^2}{2\sigma_i^2})/\sqrt{2\pi\sigma_i^2}.$$

Of course, the product is rather awkward to work with, so let us instead work with the log of the

[1] citation needed

[2] We are using a here and not the usual m to avoid confusion down the line.

PDF:

$$\log \mathrm{PDF} = \sum \frac{-(d_i - \mu_i)^2}{2\sigma_i^2} - \sum \frac{1}{2}\log(2\pi\sigma_i^2)$$

We can also pull a factor of $-\frac{1}{2}$ to the left hand side to further simplify matters. The PDF is also often referred to as the likelihood $\mathcal{L}$, so we make that change as well, giving:

$$-2\log(\mathcal{L}) = \sum \frac{(d_i - \mu_i)^2}{\sigma_i^2} + \sum \log(2\pi\sigma_i^2)$$

With this expression for the likelihood in hand, we can now say how likely it is that given a model μ_i and errors σ_i would have given rise to the observed data d_i. Of course, what we really want to ask is "given the data, what is the true model." Unfortunately, that question is rather ill-posed (see *e.g.* the long literature on Bayesian statistics). Instead, the question we know how to answer is "given a model, how likely is it to have given rise to the data we got." We can ask this question about many different models, and can compare the relative probabilities of two different models giving rise to the observed data. That is a well-defined question and the answer is just the ratio of the likelihoods/PDFs. In log space, the ratio becomes a difference, and we have for models μ_i and $\mu_i^\dagger$

$$-2\delta\log(\mathcal{L}) = \sum \frac{(x_i - \mu_i)^2}{\sigma_i^2} - \sum \frac{(x_i - \mu_i^\dagger)^2}{\sigma_i^2}$$

Happily, the second term in the likleihood has gone away, because it doesn't depend on the model and so gets cancelled in the difference. Note that if we ever try to fit for the σ's, though, we would need to keep that term around.

Looking at the likelihood difference expression, we can see that the full information we need to know for a given model is just the sum, which is called χ^2

$$\chi^2 \equiv \sum \frac{(x_i - \mu_i)^2}{\sigma_i^2}$$

Once I have the value for χ^2 for a set of models in-hand, I can quickly compare the relative likelihoods by noting that $-2\delta\log(\mathcal{L}) = \delta\chi^2$, which gives a likelihood ratio for the two models of $\exp(-\frac{\delta\chi^2}{2})$. If the difference between two models is large, the model with the larger χ^2 is exponentially less likely to have given arise to the observed data. Similarly, is $\delta\chi^2$ is small (where small means much less than 1), then the relative probabilities are very similar.

In short, we use χ^2 because for Gaussian random variables, everything we need to know about the relative probailities of two models producing the observed data is encapsulated in χ^2.

2. Searching for the "Best" Model

While it is useful to be able to compare two different models, usually what we want to do is find the model that does the "best" job fitting the data. In more general cases, even defining what we mean by this question can get quite complicated, and reasonable people can disagree. We can, however, get to an answer by defining the "best" model as the one that had the highest likelihood of producing the observed data, which is usually described as the frequentist approach. Since we know that low values of χ^2 mean those models have relatively higher probabilities of producing our observed data, we can say that the "best" model is the one that minimizes χ^2. Very often, we have models that depend on one or more continuous parameters. One example of this that we have arleady seen is fitting a line to data, where we can set the slope and intercept to be whatever we want. The problem then becomes, of all the infinite possible values, how do we find the ones that minimize χ^2?

In searching for the parameter values that minimize χ^2, which from now on we will refer to as the "best-fit" values, we could directly differentiate χ^2 as we have written it, and possibly find a solution. However, it is usually simpler to understand what is going on, and how best to proceed, if we re-write χ^2 in the language of linear algebra. We can start by putting the observed data points and their expected values into column vectors d and μ. Furthermore, we can define a diagonal matrix N where $\mathrm{N}_{i,i} = \sigma_i^2$. In this case, N^{-1} is just trivially $\mathrm{N}^{-1}_{i,i} = 1/\sigma_i^2$. If we then take $\mathrm{N}^{-1}(d-\mu)$ we can see that this makes a column vector, with i^{th} element just $(d_i - \mu_i)/\sigma_i^2$. Recalling that a row vector times a column vector is just a dot product, we can take $(d-\mu)^T\left(\mathrm{N}^{-1}(d-\mu)\right)$. This becomes $\sum \frac{(d_i-\mu_i)^2}{\sigma_i^2}$, which was just our original expression for χ^2. So, in linear algebra notation, we have

$$\chi^2 = (d-\mu)^T\mathrm{N}^{-1}(d-\mu)$$

. Of course, we usually have some form for what the data values should be, and usually that depends on some set of model parameters. In this case, $\mu_i =< d_i >= A_i(m)$ where A_i is some function that depends on both the model parameters and which data point we're looking at. Again, in the case of fitting a line to data, we need to know both the slope/intercept *and* the x-value of a point in question before we can predict the y-value. Of course, we can as usual turn these things into vectors, giving us

$$\chi^2 = (d - A(m))^T\mathrm{N}^{-1}(d - A(m)) \tag{1}$$

where if we knew the true model parameters, we'd have $< d >= A(m_{true})$. If we were arbitrarily good at math, we could differentiate χ^2 with respect to m, and find the global minimum, which would give us our best fit parameters. Unfortunately, this is usually very hard to do.

3. Linear Least Squares

Fortunately, there is a very broad class of models where we can analytically solve for our best-fit parameters. If we make the important simplification that we'll restrict ourselves to models that depend *linearly* on the model parameters, it turns out we can indeed solve for the global minimum of χ^2. What do we mean by a linear dependence? We mean

$$< d >= \mathrm{A}m \tag{2}$$

for a potentially arbitrary matrix A and fit parameters m. Generally, we decide on A, which corresponds to selecting which class of models we're going to look at, and then finding the parameters that give us the best model in that whole class. Going back to the line-fitting example where $< d_i >= at_i + b$, we could write A as an n-data by 2 matrix where the first column is t_i and the second colum is just a vector of ones. Our fit parameters m (for model parameters) is then just a column vector consisting of the slope and intercept $m = [ab]$. If you carry one the multiplication $\mathrm{A}m$ you can see that the individual elements are $at_i + b$, so line-fitting fits nicely into this formalism. It's important to note that once we said we were going to fit a line to our data, that specified A. We could have instead said we'd fit a quadratic or cubic polynomial, or sum of exponentials, or sum of sines and cosines, and that each of the choices would have specified a different version of A. For now, we will not worry about how to select between different choices of A, but just worry about how to pick the best parameters once we have selected A.

We can rewrite χ^2 to reflect the fact we have restricted ourselves to the linear case:

$$\chi^2 = (d - \mathrm{A}m)^T \mathrm{N}^{-1} (d - \mathrm{A}m) \tag{3}$$

. We now have to find the values of m that minimize χ^2. As usual, we do this by taking the derivative of χ^2 with respect to m. As shown in the appendix, this is just

$$\nabla \chi^2 = -2\mathrm{A}^T \mathrm{N}^{-1} (d - \mathrm{A}m)$$

. At the minimum of χ^2, we know this must be equal to zero and so we have

$$-2\mathrm{A}^T \mathrm{N}^{-1} (d - \mathrm{A}m) = 0$$

or

$$\mathrm{A}^T \mathrm{N}^{-1} \mathrm{A}m = \mathrm{A}^T \mathrm{N}^{-1} d \tag{4}$$

This is the fundamental equation of linear least-squares and people who analyze data would do well to ruminate on it periodically. When tackling large problems, I find it more useful to think about this equation in terms of operators. A takes a model and maps it to data. N^{-1} inverse variance-weights data. A^T takes data and projects it back into a somethign that looks like a model, but is not normalized. As an example, if we are using least-squares to solve for a map of the sky given data from a telescope scanning repeatedly over a patch of sky, A^T times a vector of ones would give us the "hit counts" – the map of how many times each map pixel was observed by a detector. We can then interpret Equation 4 as follows: The right-hand side is the inverse variance-weighted data, projected onto our fit parameters. The left hand side maps parameters into data, then inverse variance weights them and projects onto parameters. Our maximum likelihood solution is where these two are equal. If we somehow *knew* (or guessed) the solution, we could check it very quickly, just by operating on the data/model.

It is tempting to re-write Equation 4 as

$$m = \left(\mathrm{A}^T \mathrm{N}^{-1} \mathrm{A}\right)^{-1} \mathrm{A}^T \mathrm{N}^{-1} d$$

Indeed, often that is exactly what we *should* do. However, in a wide range of common cases, $\mathrm{A}^T \mathrm{N}^{-1} \mathrm{A}$ will not lend itself to inversion. Either it is too large to invert (let alone construct) or is singular. Often it will be both. In such cases, iterative solutions to Equation 4 are preferred, where we guess a solution, run it through the left-hand side of Equation 4, and compare it to the right-hand side. Based on the difference, we update our guess and repeat until we're close enough.

3.1. Bias

: We have shown that Equation 4 gives the maximum-likelihood fit to our data provided that we know the noise in the data, and that our model can describe the data (Equation 2). Let's calculate the expected difference between our recovered model m and the true one m_t. We will also set $d = d_t + n$ where d_t are the noise-free data, and n is the (unknown) realization of the noise we happened to get. Since $d_t = \mathrm{A}m_t$, is is obviously true that:

$$\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}m_t = \mathrm{A}^T\mathrm{N}^{-1}d_t$$

Note that this depends on Equation 2 being true, but critically *not* on N describing the variance in the data. We can then subtract this result from Equation 4 giving:

$$\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}(m - m_t) = \mathrm{A}^T\mathrm{N}^{-1}(d - d_t) \tag{5}$$

Of course $d - d_t$ is just n, so (assuming invertability):

$$m - m_t = \left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}\mathrm{A}^T\mathrm{N}^{-1}n$$

Usually this will average to zero, since the noise is positive and negative with equal probability, meaning our maximum-likelihood parameters are unbiased. This is relied on Equation 2 being true, but not on N being an accurate description of the variance. The closer N is to the true variance, the closer m will be to the maximum-likelihood solution, but it should remain unbiased. One note of caution, however: extreme caution must be used when estimating the noise directly from the data, since we can inadvertently couple *errors* in the noise model to signal in the data, which can make the right-hand side of Equation 5 non-zero. A simple example would be using d^2 as our variance when estimating the mean of the data. If the mean is positive, data that happened to fluctuate high will be assigned higher variance, and hence lower weight, than data that happened to fluctuate low. The result will be an estimate of the mean biased low. The usual solution to this problem is to subtract the model from the data before estimating the noise, then fitting the model. After multiple iterations this process should converge, but it can sometimes be slow when the SNR per-point is high. Other solutions are possible, including making a (possibly extremely) non-optimal but unbiased estimate of the model before estimating noise for the first time. While solutions vary, just being aware of the issue is most of the battle.

3.2. Model Variances

While the maximum-likelihood model is unbiased (subject to the warnings from the previous section), the parameters will still have errors. How do we describe these? If $\delta_m \equiv m - m_t$, then from Equation 5 we have

$$\delta_m\delta_m^T = \left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}\mathrm{A}^T\mathrm{N}^{-1}n\left[\left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}\mathrm{A}^T\mathrm{N}^{-1}n\right]^T$$

$$= \left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}\mathrm{A}^T\mathrm{N}^{-1}nn^T\mathrm{N}^{-1}\mathrm{A}\left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}$$

where we have used the fact that N is symmetric. If we take the expectation, then $< nn^T >= N$ which we can use to cancel out one factor of the noise matrix in the middle. This leaves us with:

$$\left\langle \delta_m \delta_m^T \right\rangle = \left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}$$

We can carry out one further cancellation, leaving us with

$$\left\langle \delta_m \delta_m^T \right\rangle = \left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1} \tag{6}$$

Fortunately, if we have directly solved our least-squares problem, we already have this matrix sitting around! While this gives us the full covariance of our model errors, note that the standard deviation of the model parameters is just the square root of the diagonal entries.

One further case of interest is when we care about the error in our prediction for the data and the truth. An instance of this would be fitting data with a polynomial, where the difference between our model and truth might be more interesting than the formal errors on the polynomial coefficients themselves. We know our estimate for the data is given by $d = \mathrm{A}m$, so

$$d - d_t \equiv \delta_d = \mathrm{A}(m - m_t) = \mathrm{A}\delta_m$$

The expectation of this should be zero if our model is unbiased, but we can again take the covariance:

$$\left\langle \delta_d \delta_d^T \right\rangle = \mathrm{A}\delta_m \left(\mathrm{A}\delta_m\right)^T = \mathrm{A}\left(\mathrm{A}^T\mathrm{N}^{-1}\mathrm{A}\right)^{-1}\mathrm{A}^T \tag{7}$$

4. Correlated Noise

So far, we have formulated everything assuming the errors in the data are uncorrelated $< n_i n_j > \propto \delta_{i,j}$. Life is often not so kind to us, and frequently errors are correlated. A random walk is one case, where one knows that neighboring samples are highly correlated - if sample i is high, then sample $i+1$ must also be high. How can we deal with this? We start by adding in an extra invertible matrix S into χ^2 along with its inverse, so χ^2 remains unchanged:

$$\chi^2 = (d - \mathrm{A}m)^T\mathrm{S}^T\mathrm{S}^{T,-1}\mathrm{N}^{-1}\mathrm{S}^{-1}\mathrm{S}(d - \mathrm{A}m)$$

Since S always shows up with its inverse, this *has* to have the same solution, model errors etc. as Equation 4. We can now group terms as follows:

$$\chi^2 = (\mathrm{S}d - \mathrm{SA}m)^T\left(\mathrm{SNS}^T\right)^{-1}(\mathrm{S}d - \mathrm{SA}m)$$

We can now introduce new variables:

$$\tilde{d} = \mathrm{S}d$$

$$\tilde{\mathrm{A}} = \mathrm{SA}$$

$$\tilde{\mathrm{N}} = \mathrm{SNS}^T$$

We can now write χ^2 with the new variables:

$$\chi^2 = \left(\tilde{d} - \tilde{\mathrm{A}}m\right)^T\tilde{\mathrm{N}}^{-1}\left(\tilde{d} - \tilde{\mathrm{A}}m\right)$$

By multiplying Equation 2 on the left by S, it is obviously true that

$$< \tilde{d} >= \tilde{\mathrm{A}} m$$

Less obvious is that

$$\tilde{\mathrm{N}}_{ij} =< \tilde{n}_i \tilde{n}_j >$$

We know that $\tilde{n}_i = \sum_k \mathrm{S}_{ik} n_k$. So

$$\langle \tilde{n}_i \tilde{n}_j \rangle = \left\langle \sum_k \mathrm{S}_{ik} n_k \sum_l \mathrm{S}_{jl} n_l \right\rangle$$

However, the cross terms are all zero unless $k = l$, so this transforms into a single sum:

$$\langle \tilde{n}_i \tilde{n}_j \rangle = \left\langle \sum_k \mathrm{S}_{ik} \mathrm{S}_{jk} n_k^2 \right\rangle = \sum_k \mathrm{S}_{ik} \mathrm{S}_{jk} \sigma_k^2 \tag{8}$$

Now, let's see what $\tilde{\mathrm{N}}_{ij}$ is equal to:

$$(\mathrm{SN})_{ik} = \mathrm{S}_{ik} \sigma_k^2$$

Now, since S^T is the final term in $\tilde{\mathrm{N}}$, we can get the rotated noise elements:

$$\tilde{\mathrm{N}}_{ij} = \sum_k (\mathrm{SN})_{ik} \mathrm{S}^T_{kj}$$

But S^T_{kj} is just S_{jk}, so we have

$$\tilde{\mathrm{N}}_{ij} = \sum_k \mathrm{S}_{ik} \mathrm{S}_{jk} \sigma_k^2$$

This result is just Equation 8, so as long as we can calculate the $< \tilde{n}_i \tilde{n}_j >$ in our new coordinate system, we can calculate χ^2 without ever referring to the original, uncorrelated data! So, our expression for χ^2 (Equation 3) is true even if our noise is correlated as long as $\mathrm{N}_{ij} =< n_i n_j >$. Since that is true, the entire framework we have developed for solving least-squares problems carries through directly. While writing the equations down using linear algebra started out as a notational convenience, it has become a powerful tool letting us analyze data with correlated errors as easily as uncorrelated errors.

5. Nonlinear Least-Squares

Sadly, we occasionally meet non-linear functions in the real world. Naturally, this will be more complicated than the linear case, but often manageably so. The linear case guarantees a single global minimum of χ^2, and we can solve for that global minimum directly. The non-linear case makes no such guarantee. Instead, we will start at a (hopefully well-chosen) point in parameter space, and try to reason from the local properties of chi^2 about where a minimum might lie. We can almost always find *a* minimum, but it could very well be a local minimum.

5.1. Newton's Method

Continuing to assume Gaussian errors, we can Taylor expand chi^2 to second order. That Taylor expansion will now have a quadratic form, so we can find the global minimum of χ^2 of the Taylor expansion. If we're "close enough", that minimum should be close to the actual desired minimum. Of course, it won't be exact, but we're now (hopefully) closer to the true minimum. If we repeat the process, we should be able to take an even more accurate step, and we can just keep doing this until the changes in χ^2 are small enough we get bored and quit. This procedure is the multidimensional version of Newton's method.

To carry out Newont's method, we will need to calculate the gradient and curvature of χ^2. Starting from the general expression for χ^2 (Equation **??**) we can take the gradient:

$$\nabla\chi^2 = -2\mathrm{A}^{',T}\mathrm{N}^{-1}((A)(m) - d)$$

where $\mathrm{A}^{'}$ is a matrix with entries $\mathrm{A}^{'}_{i,j}$ equal to the derivative of the model $\mathrm{A}(m)$ for data point i with respect to parameter j. If we define the *residual* $r \equiv d - \mathrm{A}(m)$ then we can simplify the expression for $\nabla\chi^2$ a bit more:

$$\nabla\chi^2 = -2\mathrm{A}^{',T}\mathrm{N}^{-1}r$$

We're also going to need the curvature, which we get by differentiating the gradiant:

$$\nabla\nabla\chi^2 = -2\mathrm{A}^{'',T}\mathrm{N}^{-1}r + 2\mathrm{A}^{',T}\mathrm{N}^{-1}\mathrm{A}^{'}$$

. Before continuing, let's look at the two terms in the curvature. The second term should look familiar - it is the same as the linear case. The first term however has got second derivatives of our function. It also has the residual in it. If we are close to the minimum, the residual should 1) be small and 2) be positive and negative with (roughly) equal probability. For these reasons, the first term is *usually* smaller than the second term, so we *usually* approximate the curvature with just the second term. An added bonus is that we already have everything we need to calculate the second term, so an easy (and often productive) plan is to forge boldly ahead ignoring the first term, and see if we converge. If yes, great! If not, you might consider calculating the first term, but we will see another technique in the next section where we can usually still get away with just the second term.

With the gradient and curvature in hand, we can now write down Newton's method. As a reminder from one dimension, if we want to find a zero of $f(x)$, we take $x_{n+1} = x_n - f(x_n)/f'(x_n)$. In our case, we want to set the gradient of χ^2 equal to zero, so $f \to \nabla\chi^2$, $f^{'} \to \nabla\nabla\chi^2$, and the division gets replaced by a multiplication by a matrix inverse. We can now string together all our terms to get the multidimensional Newton's method to minimize χ^2. If our position in parameter space after step n is m_n, we have:

$$m_{n+1} = m_n + \left(\mathrm{A}^{',T}\mathrm{N}^{-1}\mathrm{A}^{'}\right)^{-1}\left(\mathrm{A}^{',T}\mathrm{N}^{-1}r\right)$$

This process may converge, in which case your'e done. When chi^2 quits changing appreciably (if the noise is properly normalized, that happens for $\delta\chi^2 << 1$, but if it's not you may need to set up some other convergence check) then you stop iterating, and take the parameters m_n as your maximum likelihood model. To the extent that the Taylor expansion is a good one, the results from linear least squares on error bars and model uncertainties just carry directly over. You'll have an idea if this is the case based on how quickly Newton's method converges once χ^2 gets within a few of the final minimum. If it leaps to very close to the minimum in a single step, it's reasonable to have confidence in the Taylor expansion. If it takes many steps, the Taylor expansion is known to not be very good, so you may wish to poke around parameter space a bit more.

5.2. Levenberg-Marquardt

Newton's method is great *when it works.* If it converges, it is almost certainly the fastest (in terms of computer time) and easiest (in terms of your time) way to fit nonlinear models. Sadly, it frequently fails, and the failures are usually spectacular. What's a poor student, with a thesis to write, to do? If we start close enough to a minimum, Newton's method will proabbly work. It's hard to do that, though, since if you knew where the minimum was, you'd already be done. A more robust, but exceedinly clunky, technique would be to pick a starting point, and try Newton's method. If it converges, go home. If it doesn't, try to find a better starting point. As we saw in the previous section, we can calculate the local gradient of χ^2, so we could find a better starting point by following the gradient downhill for a while, then try Newton's method again. Do this enough times, and eventually Newton's method should converge. Wouldn't it be nice to have a more graceful way of doing something similar?

Fortunately, there exist such techniques. The most common was developed by Levenberg and Marquardt. Qualitatively, it works by smoothly transitioning between downhill-type steps and Newton's method-type steps based on how well convergence has been going. If convergence has been bad, take a lot of short steps in a downhill direction, while if it has been good, try to jump directly to the minimum. To do this, we introduce an extra parameter λ, which lets us transition between downhill and Newton's steps. When $\lambda \rightarrow 0$ we approach Newton's method, while when $\lambda \rightarrow \infty$, we take increasingly short downhill steps. That sounds nice, but it raises two critical questions: 1) how do we actually implement this, and 2) what order of magnitude should λ even be? Let's start by addressing the second question. If we had a single variable, our Newton's method step would be $-f'/f''$. In our multidimensional case, if we fit for parameter i while holding all other parameters fixed, our natural step size would be the i^{th} element of the gradient divided by the i^{th} element along the curvature of the diagonal. A way to do this is have $m_{i,n+1} = m_{i,n} - grad(i)/curve(i,i)/(1+\lambda)$. If $lambda = 0$, this is just the Newton's method step, and as λ gets large, we take increasingly smaller steps in the downhill direction. For $\lambda = 1$, we take exactly half of our Newton's method step. By including the curvature, we've been able to set the scaling of λ so that it is dimensionless, and for $\lambda << 1$ we're taking Newton's method steps, and for $\lambda >> 1$, we're taking very short but still downhill steps.

Now that we've answered our second question, about the scaling of λ, a solution to the first problem suggests itself. If we rescale the diagonal of the covariance matrix ($\mathrm{A}^{',T}\mathrm{N}^{-1}\mathrm{A}'$) by a factor of $1 + \lambda$, then when λ approaches zero we take the usual Newton's method step, and when λ is large, then the covariance approaches a diagonal matrix, and the step approaches $1/\lambda$ times the downhill step. This, coupled with a scheme to update λ, is the Levenberg-Marquardt algorithm. If we take a "bad" step (where bad is usually defined by an increase in χ^2), then increase λ and try again from the same starting point. If we take a "good" step, then move to the new point and decrease λ. There is considerable freedom in how you update λ. My preference is to start with $\lambda = 0$. If every step succeeds, then we just leave λ at zero, and we end up with Newton's method. Once we take a failed step, immediately increase λ to order unity. The motivation is that if χ^2 has increased, we probably missed the minimum by at least a factor of 2, in which case we would want to take no more than a half-sized step. As long as we keep failing, increase λ by a factor of a few each iteration. Eventually (modulo numerical innacuracies) this has to give us a step where χ^2 shrinks. Once we've taken a successful step, decrease λ. I usually decrease λ slower than I increase it because in my experience, even though you may have taken a successful step you aren't yet out of the woods. If that's the case, being greedy usually backfires. Of course, other schemes are possible. Numerical Recipes suggests not penalizing the fitter after the first failed step since Newton's frequently recovers.

Levenberg-Marquardt is quite robust in the sense that it will converge to a minimum. That minimum may or may not be the global minimum - if you don't understand your problem well enough to know if there's only one minimum, a standard thing to do is restart LM from several different points in parameter space and see if they end up finding the same minimum. If yes, you're probably at the global minimum. Given its robustness and relative ease of coding, LM is the standard "fast" tool for nonlinear model-fitting. If you can code your derivatives (or have the patience to take numerical derivatives), LM is a good default starting point to get maximum-liklihood parameters and (approximate) error bars.

6. Markov Chain Monte Carlo

Sometimes we are faced with nonlinear fitting problems with complicated likelihood surfaces. Levenberg-Marquardt works very well when there is a single dominant minimum that is reasonably well described by a second order Taylor expansion. For broad classes of problems, though, this can fail, and fail spectacularly. A classic example is a χ^2 surface with multiple minima. LM is unlikely to notice the multiple minima, and its reported error bars will be derived from the curvature around the minimum it found. If you blindly use those error bars, you can underestimate your uncertainties by orders of magnitude. A very robust (and easy to code) tool that can handle complicated likelihood surfaces is Markov Chain Monte Carlo (MCMC). The downside is that it's hundreds of times slower than Newton's method or LM, and that's on a good day.

The inspiration for MCMC comes from thermodynamics. I'll go through the motivation, but don't worry if you don't follow all the details. You'll still be able to happily and usefully use MCMC. We'll start with the seemingly unrelated problem of a two-state system with populations n_1 and n_2, and transition probabilities p_{12} for a particle in state 1 to transition to state 2, and p_{21} for the reverse. The time evolution of the system is then

$$\frac{\mathrm{d}n_1}{\mathrm{d}t} = -n_1 p_{12} + n_2 p_{21}$$

and similar for state 2. If we let the system come into equilibrium, then the derivative must be equal to zero, so we have

$$\frac{n_1}{n_2} = \frac{p_{21}}{p_{12}} \tag{9}$$

That is, the ratio of the population of the two states is equal to the ratio of their transition probabilities. We could carry out a simulation of this with a single particle. At every time step, we roll a die, and based on our roll and on the transition probabilities, we decide if the particle jumps states or not. If we wait long enough, we know that the fraction of the time the particle spends in state 1 divided by the fraction of the time it spends in state 2 is just the ratio of p_{21} to p_{12}. A fact that will become crucial is that the absolute magnitudes of the transition proabilities don't matter - it is just the ratio. If the probabilities are very low, we will have to wait a very long time for the particle's behavior to approach the average, but if we wait long enough, it will.

How does this extend to higher numbers of states? Well, we can convert our two state system into a matrix equation:

$$\begin{pmatrix} \dot{n_1} \\ \dot{n_2} \end{pmatrix} = \begin{pmatrix} -p_{12} & p_{21} \\ p_{12} & -p_{21} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \tag{10}$$

In steady state, this must equal zero. We can now build up a many-state transition matrix by adding together all the two-state matrices. As long as the phase space densities are correct (*i.e.* we set $n_1, n_2, n_3 ...$ correctly), then every two-state matrix will give zero when we multiply by the phase space density, and so *every* possible sum of two-state matrices will give zero. As a more concrete example, let's take a three state system and explicitly write down all the transition matrices:

$$\left[\alpha_{12} \begin{pmatrix} -p_{12} & p_{21} & 0 \\ p_{12} & -p_{21} & 0 \\ 0 & 0 & 0 \end{pmatrix} + \alpha_{13} \begin{pmatrix} -p_{13} & 0 & p_{31} \\ 0 & 0 & 0 \\ p_{13} & 0 & -p_{31} \end{pmatrix} + \alpha_{23} \begin{pmatrix} 0 & 0 & 0 \\ 0 & -p_{23} & p_{32} \\ 0 & p_{23} & -p_{32} \end{pmatrix} \right] \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} \tag{11}$$

As long as Equation 9 holds for each pair of states, then this must be equal to zero for *any* values of α_{ij}. You can think of the α's as mixing rates between the states. Large values of α mean that particles are likely to bounce back and forth between those two states quickly, while small values of α mean particles will only slowly transition between the two states. Note also that there is nothing special about the number of states. In particular, we can reproduce continuous distributions by taking the limit as the number of states approaches infinity.

With these results in hand, we are now ready to make the key leap. We can simulate a system by starting in a state, picking a new state the system *might* jump to, and then deciding if the system actually makes the jump or not. By convention, if the phase space density is higher at the trial point, we always take the jump. If it is lower, we *sometimes* take the jump, with probability $\frac{n_i}{n_j}$. If we wait long enough, then the the path our system took in our simulation (called a Markov chain) will have its observed phase space density converge to the underlying average. How we pick trial state j when starting at state i is called the trial distribution. Crucially, this doesn't rely on the trial distribution. A better trial distribution means our Markov chain converges *faster*, but it will still converge even with a poor trial distribution. The only formal constraint is that the probability for state i to attempt to transition to state j has to be the same for state j to transition to state i.

How do Markov chains relate to parameter fitting? Let's say we want to calculate the expectation of a model parameter. If we know how to calculate the likelihood of a given set of model parameters, we can write down the expectation of model parameter m_i:

$$\langle m_i \rangle = \int m_i \mathcal{L}(m) \mathrm{d}^n m$$

We could in principle just calculate this integral, but these sorts of integrals can rapidly become intractable. Let's say we have 10 parameters, and want to do a numerical integration going from -3σ to 3σ in steps of 0.2σ. In that case, each parameter can have one of 31 possible values, so to brute force this integral we would need to evaluate the likelihood at 31^{10} different points in parameer space. That works out to over 800 trillion points. If each likelihood took 1 microsecond, we would need 25 years just to do this integral! Now let's say we had a Markov chain - we could take the expectation of m_i just by averaging the values of m_i in the chain. If the Markov chain converges at the level we care about in many fewer steps, then we can calculate parameter expectations, or indeed *any* statistical quantity, very quickly by averaging over samples in the chain.

This then is the power of using Markov chains to carry out Monte Carlo integrations over complicated likelihood surfaces (hence the name of this technique: Markov Chain Monte Carlo or MCMC). With a good trial distribution, MCMC will converge faster, but even with a poor trial distribution, it will eventually converge. The number of chain samples required to converge tends to grow only slowly with increasing numbers of parameters, and so MCMC is useful for high dimensional problems. They are particularly when the likelihood surface is non-Gaussian, since methods based on Taylor expansions can give wildly misleading results.

We can now write down the steps to carry out MCMC:
Step -1: Decide on a trial distribution. It must be symmetric, but otherwise can be (nearly) anything.
Step 0: Pick a random point in parameter space and calculate its likelihood. This is your current location.
Step 1: Take a trial step, drawn from the trial distribution.
Step 2: Evaluate the likelihood at the trial location.
Step 3: If the new likelihood is larger, move to the new location with probability 1. If the new likelihood is smaller, move to the new location with probability $\mathcal{L}_{new}/\mathcal{L}_{old}$ (you'll need to generate a random number for this).
Step 4: Save your current location, and go back to step 0.
Keep going until you're converged.

Calculus with Linear Algebra': It is likely that many people reading this note have taken linear algebra at some point but perhaps not have done calculus with it. To first order, doing calculus with linear algebra is just like doing regular calculus, as long as one is careful not to switch the order of any matrices. For instance, the derivative of the linear least-squares equations $\chi^2 = (d - \mathrm{A}m)^T \mathrm{N}^{-1}(d - \mathrm{A}m)$ is just

$$\nabla\chi^2 = -\mathrm{A}^T \mathrm{N}^{-1}(d - \mathrm{A}m) + (d - \mathrm{A}m)^T \mathrm{N}^{-1} A$$

. We've played a bit loose here since we've ignored the difference between taking the derivative of m and m^T, but remember that underneath we can think about this as taking the derivative of χ^2 with respect to the individual m_i's, each of which is a scalar. Since the two terms in the sum are just conjugates of each other, the individual elements are the same and so we can just add their elements and pick one of the transposes to use. The usual choice is to then say

$$\nabla\chi^2 = -2\mathrm{A}^T \mathrm{N}^{-1}(d - \mathrm{A}m)$$

.