Discounted Least Squares Curve Fitting

by

Hendrik J. Blok
<blok@physics.ubc.ca>

20 July, 1997

1. Introduction

I recently wrote some code to make simple forecasts in a time series (a steadily accumulating set of (x,y) data points). For its simplicity, I chose a least-squares fit to a straight line. The underlying behaviour of the system was continuously changing so it was unreasonable to expect the same parameters to be valid for all the data. As new data came in, I expected old data to become irrelevant, and I handled this by only fitting over the last N data points. Unfortunately, it became evident that this arbitrary parameter N I had chosen was very important to the fit, often producing pathological results: as N new data points were accumulated after an *outlier* (a strongly atypical y-value), it would suddenly be dropped from consideration and the forecast would undergo a discontinuous "jump". I began to wonder if there was a way of steadily *discounting* the relevance of past data in a smoother and more natural way…

(Note: much of the text written here is a blatant copy from Press *et al*. (1992). They said it so well, I could not word it any better myself. In Chapter 16 they derive the least-squares technique I described above.)

2. Solution

We use the index i to label our data points where i=0 indicates the most recently acquired data and i=1,2,3,… indicate successively older data. Each data point consists of a triplet (x,y,σ) where x is the independent variable (eg. time), y is the dependent variable, and σ is the associated measurement error in y.

As new data arrives (x_0,y_0,σ_0) we shift the indices of prior data to make room, and scale up the errors by some factor $\gamma \in (0,1)$:

$$(x_{i+1}, y_{i+1}, \sigma_{i+1}) \leftarrow (x_i, y_i, \sigma_i / \gamma) \tag{2.1}$$

If we define σ_i^* as the original value of σ_0 then after applying i of the above operations

$$\sigma_i = \sigma_i^* / \gamma^i \tag{2.2}$$

so, since γ<1, the historical deviations grow exponentially as new information is acquired.

We wish to fit data to a model which is a linear combination of *any M* specified functions of x. For example, the functions could be $1, x, x^2, \ldots, x^{M-1}$, in which case their general linear combination,

$$y(x) = a_1 + a_2 x + \cdots + a_M x^{M-1} \tag{2.3}$$

is a polynomial of degree M-1. The general form of this kind of model is

$$y(x) = \sum_{j=1}^{M} a_j X_j(x) \tag{2.4}$$

where $X_1(x), \ldots, X_M(x)$ are arbitrary fixed functions of x, called the *basis functions*. Note that the functions $X_j(x)$ can be wildly nonlinear functions of x. In this discussion "linear" refers only to the model's dependence on its *parameters* a_j.

For these linear models we define a merit function

$$\chi^2 = \sum_{i=0}^{N} \left[\frac{y_i - \sum_j a_j X_j(x_i)}{\sigma_i} \right]^2 \tag{2.5}$$

We will pick as best parameters those that minimize χ^2. There are several different techniques for finding this minimum. We will focus on one: the *singular value decomposition* of the *normal equations*. To introduce it we need some notation.

Let **A** be a matrix whose $N \times M$ components are constructed from the M basis functions evaluated at the N abscissas x_i, and from the N measurement errors σ_i, by the prescription

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i} \tag{2.6}$$

The matrix **A** is called the *design matrix* of the fitting problem. Notice that in general **A** has more rows than columns, $N \geq M$, since there must be more data points than model parameters to be solved for.

Also define a vector **b** of length N by

$$b_i = \frac{y_i}{\sigma_i} \tag{2.7}$$

and denote the M vector whose components are the parameters to be fitted, $a_1, \ldots, a_M$, by **a**.

If we take the derivative of (2.5) with respect to all M parameters a_j, we obtain M equations that must hold at the chi-square minimum,

$$0 = \sum_i \frac{1}{\sigma_i^2} \left[y_i - \sum_j a_j X_j(x_i) \right] X_k(x_i) \quad k = 1, \ldots, M \tag{2.8}$$

Interchanging the order of summations, we can write (2.8) as the matrix equation

$$\sum_j \alpha_{kj} a_j = \beta_k \tag{2.9}$$

where

$$\alpha_{kj} = \sum_i \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2} \quad \text{or equivalently} \quad [\alpha] = \mathbf{A}^T \cdot \mathbf{A} \tag{2.10}$$

an $M \times M$ matrix, and

$$\beta_k = \sum_i \frac{y_i X_k(x_i)}{\sigma_i^2} \quad \text{or equivalently} \quad [\beta] = \mathbf{A}^T \cdot \mathbf{b} \tag{2.11}$$

a vector of length M.

The equations (2.8) or (2.9) are called the *normal equations* of the least-squares problem. They can be solved for the vector parameters **a** by *singular value decomposition* (SVD). SVD solves fixes many difficulties in the normal equations, including susceptibility to round-off errors. SVD can be significantly slower than other methods; however, its great advantage, that it (theoretically) *cannot fail*, more than makes up for the speed disadvantage. A good review of SVD techniques can be found in Press *et al*. (1992) §2.6. In matrix form, the normal equations can be written as

$$[\alpha] \cdot \mathbf{a} = [\beta]. \tag{2.12}$$

3. Covariance Matrix

Let us define

$$[C] = [\alpha]^{-1}. \tag{3.1}$$

Then

$$\mathbf{a} = [C] \cdot [\beta] \quad \text{or} \quad a_j = \sum_k C_{jk} \beta_k \tag{3.2}$$

which allows us to determine **a**'s dependence on the y_i values. From (2.10) we see that $[C]$ is independent of y_i so

$$a_j = \sum_j C_{jk} \sum_i A_{ik} \frac{y_i}{\sigma_i} \tag{3.3}$$

and

$$\frac{\partial a_j}{\partial y_i} = \sum_k C_{jk} \frac{A_{ik}}{\sigma_i}. \tag{3.4}$$

The covariance between two parameters a_j and a_k is defined as

$$\begin{aligned} \text{Covar}\left[a_j, a_k\right] &\equiv \sum_i \sigma_i^2 \frac{\partial a_j}{\partial y_i} \frac{\partial a_k}{\partial y_i} \\ &= \sum_i \sigma_i^2 \sum_{lm} C_{jl} \frac{A_{il}}{\sigma_i} C_{km} \frac{A_{im}}{\sigma_i} \\ &= \sum_{lm} C_{jl} C_{km} \alpha_{ml} \end{aligned} \tag{3.5}$$

but since $[C] = [\alpha]^{-1}$ so

$$\sum_m C_{km} \alpha_{ml} = \delta_{kl} \quad \text{or} \quad [C][\alpha] = \mathbf{1} \tag{3.6}$$

where δ_{kl} is the Kronecker delta function. Hence (3.5) reduces to

$$\text{Covar}\left[a_j, a_k\right] = C_{jk} \tag{3.7}$$

and we find that [C] is the covariance matrix. The variance of a single parameter a_j is simply defined as

$$\mathrm{Var}\left[a_j\right] = \mathrm{Covar}\left[a_j, a_j\right] = C_{jj} \,. \tag{3.8}$$

4. Storage and Updating

So far we have made no mention of N, the number of data points to be fit. As we will see an advantage of the discounted least squares method is that N becomes irrelevant. As data points are accumulated the oldest data becomes decreasingly relevant and eventually contribute negligibly to the fitting procedure. Hence we can *theoretically* apply this data to an infinite data set. But can this be *practically* implemented? The answer is…Yes!

Notice that as we acquire new data (x_0,y_0,σ_0) according to (2.10) and (2.11) the matrix [α] and vector [β] update as

$$\alpha_{kj} \leftarrow A_{0j}A_{0k} + \gamma^2\alpha_{kj} = \frac{X_j(x_0)X_k(x_0)}{\sigma_0^{\,2}} + \gamma^2\alpha_{kj} \tag{4.1}$$

and

$$\beta_j \leftarrow A_{0j}b_0 + \gamma^2\beta_j = \frac{X_j(x_0)y_0}{\sigma_0^{\,2}} + \gamma^2\beta_j \tag{4.2}$$

so it becomes clear that we need not even store the history of data points, but should rather store just [α] and [β] and update them as new data is accumulated.

A useful measure we have neglected to calculate so far is χ^2,the chi-square statistic itself. In matrix notation (2.5) can be written

$$\begin{aligned}
\chi^2 &= (\mathbf{a}^T\cdot\mathbf{A}^T - \mathbf{b}^T)\cdot(\mathbf{A}\cdot\mathbf{a} - \mathbf{b}) \\
&= \mathbf{a}^T\cdot\mathbf{A}^T\cdot\mathbf{A}\cdot\mathbf{a} - \mathbf{b}^T\cdot\mathbf{A}\cdot\mathbf{a} - \mathbf{a}^T\cdot\mathbf{A}^T\cdot\mathbf{b} + \mathbf{b}^T\cdot\mathbf{b} \\
&= \mathbf{a}^T\cdot([\alpha]\cdot\mathbf{a} - [\beta]) - [\beta]^T\cdot\mathbf{a} + \mathbf{b}^T\cdot\mathbf{b} \\
&= \mathbf{b}^T\cdot\mathbf{b} - [\beta]^T\cdot\mathbf{a}
\end{aligned} \tag{4.3}$$

which appears to still depend on the data history in the first term. Let us define this term as a new variable δ,

$$\delta \equiv \mathbf{b}^T\cdot\mathbf{b} = \sum_i b_i^{\,2} \,. \tag{4.4}$$

Then, similarly to (4.1) and (4.2) δ can be updated as more information is accumulated

$$\delta \leftarrow b_0^{\,2} + \gamma^2\delta = \frac{y_0^{\,2}}{\sigma_0^{\,2}} + \gamma^2\delta \,. \tag{4.5}$$

So, to store all relevant history information we need only remember [α], [β], and δ as well as the latest data triplet (x_0,y_0,σ_0) for a total of M^2+M+4 numbers, regardless of how many data points have been acquired. Figure 4.1 shows that for many practical problems discounted least-squares fitting requires less storage than other methods. Although it has not been tested, we expect a similar condition to hold for processing time because the

number of calculations depend only on M instead of M and N as in traditional least-squares methods.

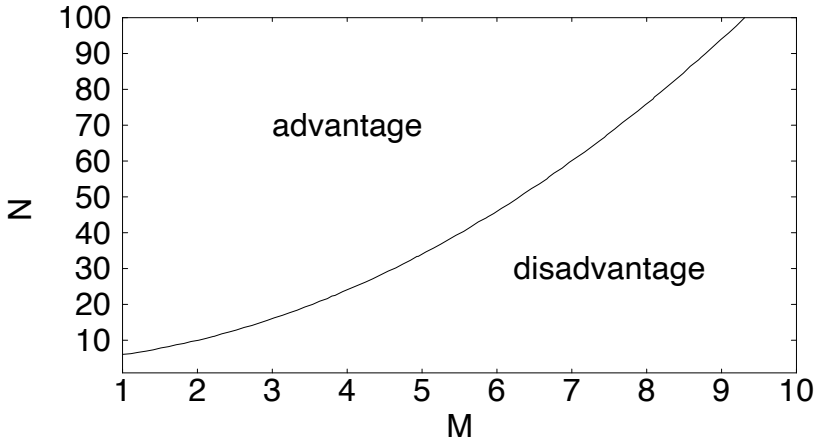

Figure 4.1 Discounted least-squares fitting has a computational storage advantage over traditional least-squares when $N>M^2+M+4$ where N is the number of data points and M is the number of parameters to be fitted.

As the reader can justify, all of these values should be initialized (prior to any data) with null values: $[\alpha]=\mathbf{0}$, $[\beta]=\mathbf{0}$, and $\delta=0$.

5. Memory, Effective Number of Data Points

For traditional least-squares fitting it is well known that if the measurement errors of y_i are distributed normally then the method is a *maximum likelihood estimation* and the expectation value (average) of (2.5) evaluates to

$$\left\langle \chi^2 \right\rangle = N - M \ . \tag{5.1}$$

This arises because $(y_i - y(x_i))/\sigma_i$ is distributed normally with mean 0 and variance 1, so the sum of N variances should equate to N. The subtraction of M is necessary because M parameters can be adjusted to actually reduce the variances further. For instance, with $N=M=1$ we can adjust the single parameter such that the curve passes precisely through the point (x_0,y_0), with zero variance. Similarly, for our method

$$\begin{aligned} \left\langle \chi^2 \right\rangle &= \sum_{i=0}^{\infty} \gamma^{2i} \left\langle \left[\frac{y_i - y(x_i)}{\sigma_i^*} \right]^2 \right\rangle - M \\ &= \sum_{i=0}^{\infty} \gamma^{2i} - M \\ &= \frac{1}{1-\gamma^2} - M \end{aligned} \tag{5.2}$$

which strongly suggests an *effective number of data points*

$$N_{eff} \equiv \frac{1}{1-\gamma^2}. \tag{5.3}$$

5.1 Memory

We now undertake a thought experiment to understand how N_{eff} comes into play. Consider a single parameter fit $X(x_i)=1$ or $y(x_i)=a$ to a long history of values $y_i=1$ (regardless of x_i). Updating according to (4.1), (4.2) and (4.5) gives

$$\begin{aligned} \alpha &\leftarrow 1+\gamma^2\alpha \\ \beta &\leftarrow y+\gamma^2\beta \\ \delta &\leftarrow y^2+\gamma^2\delta \end{aligned} \tag{5.4}$$

so after a long history of $y=1$,

$$\begin{aligned} \alpha &= 1+\gamma^2\left(1+\gamma^2(\ldots)\right) = 1+\gamma^2+\gamma^4+\ldots \\ &= \frac{1}{1-\gamma^2} = \beta = \delta \end{aligned} \tag{5.5}$$

which has a solution, in one dimension, of

$$a = \frac{\beta}{\alpha} \tag{5.6}$$

or $y(x)=a=1$.

Now consider a sudden shift in the data stream to $y_i=0$ (like a step function). How will this change our curve fit? The value of α remains unchanged but β and a change as follows:

y	β	a
⋮	⋮	⋮
1	α	1
0	$\gamma^2\alpha$	γ^2
0	$\gamma^4\alpha$	γ^4
⋮	⋮	⋮

Notice that a decays exponentially to the new equilibrium $a=0$. The time constant τ for the system (time for a to decay to $1/e$) is

$$\begin{aligned} \left(\gamma^2\right)^{\tau} &= e^{-1} \\ \Rightarrow \tau &= \frac{-1}{2\ln\gamma} \end{aligned} \tag{5.7}$$

which is, as the figure below shows, almost identical to N_{eff} for all γ.

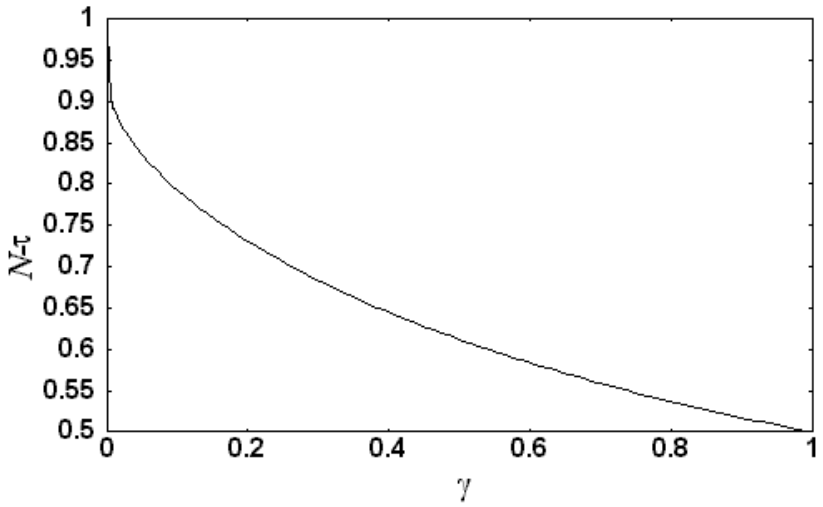

Figure 5.1 The difference between N_{eff} and τ is strictly less than one. The main difference between the two is that $\sigma_{N_{eff}} \to \infty$ as $N_{eff} \to 1$ (as it should) while $\sigma_\tau = \sqrt{e}\sigma_\tau^*$ for all τ. They converge as $N_{eff} \to \infty$.

Thus, N_{eff} is indeed a practical measure of the effective number of data points in a fit. The fit is dominated by the most recent data $i \le N_{eff}$, and N_{eff} acts as the *memory* of the fitting procedure.

6. Unknown Measurement Errors

On occasion measurement uncertainties are unknown and least-squares fitting can be used to recover an estimate of these uncertainties. Be forewarned that this technique assumes normally distributed y data with identical variances. If this is not the case, the results become meaningless. It also precludes the use of a "goodness-of-fit" estimator (such as the incomplete gamma function, see Press *et al*. (1992) §6.2) because it *assumes* a good fit.

We begin by assuming $\sigma_i^*=1$ for all data points and proceeding with our calculations of **a** and χ^2. If all (unknown) variances are equal $\sigma^*=\sigma_i^*$ then (5.2) actually becomes

$$\left\langle \chi^2 \right\rangle = (N_{eff} - M)\sigma^{*2} \tag{6.1}$$

so the actual data variance should be

$$\sigma^{*2} = \frac{\chi^2}{N_{eff} - M}. \tag{6.2}$$

We can update our parameter error estimates by recognizing that, from (3.3) and (2.6), the covariance matrix is proportional to the variance in the data, so

$$C_{jk} \leftarrow \frac{\chi^2}{N_{eff} - M} C_{jk} \,. \tag{6.3}$$

7. Forecasting

Forecasting via curve fitting is a dangerous proposition because it requires extrapolating into a region beyond the scope of the data, where different rules may apply, and hence, different parameter values. Nevertheless, it often used simply for its convenience. We assume the latest parameter estimations apply at the forecasted point x and simply use (2.4) to predict $y(x)$.

The uncertainty in the prediction can be estimated from the covariance matrix. The definition of variance for *any* distribution is the expectation value of the squared difference from the mean:

$$\mathrm{Var}\left[z\right] \equiv \left\langle \left(z - \langle z \rangle\right)^2 \right\rangle \tag{7.1}$$

and the covariance between two variables is defined as

$$\mathrm{Covar}\left[z_1, z_2\right] \equiv \left\langle \left(z_1 - \langle z_1 \rangle\right)\left(z_2 - \langle z_2 \rangle\right) \right\rangle \tag{7.2}$$

so (2.4) has a variance

$$\begin{aligned} \mathrm{Var}\left[y(x)\right] &= \mathrm{Var}\left[\sum_j a_j X_j(x)\right] \\ &= \left\langle \left(\sum_j a_j X_j(x) - \sum_j \langle a_j \rangle X_j(x) \right)^2 \right\rangle \\ &= \sum_{jk} X_j(x) \left\langle \left(a_j - \langle a_j \rangle\right)\left(a_k - \langle a_k \rangle\right) \right\rangle X_k(x) \\ &= \sum_{jk} X_j(x) \mathrm{Covar}\left[a_j, a_k\right] X_k(x) \\ &= \sum_{jk} X_j(x) C_{jk} X_k(x) \end{aligned} \tag{7.3}$$

where $[C]$ is the covariance matrix discussed in Section 3 with possible updating, in the absence of measurement errors, according to (6.3).

The above gives the uncertainty in $y(x)$, but in our derivation we have assumed the observed y-values were distributed normally around the curve where $y(x)$ represents the mean of the distribution. Similarly for our prediction, $y(x)$ is the prediction of the mean with its own uncertainty—on top of which there is the measurement uncertainty of data around the mean. If the expected measurement uncertainty is given by σ' then the predicted observation $y' \sim N\left(y(x), \sigma'\right)$ or, with the substitution

$$z' = y' - y(x) \tag{7.4}$$

we assume $z' \sim N(0, \sigma')$ *regardless of the prediction* $y(x)$. In other words, z' and $y(x)$ are mutually independent.

$$
\begin{aligned}
\mathrm{Var}\left[y'\right] &= \mathrm{Var}\left[y(x)+z'\right] \\
&= \left\langle \left(y(x)-\left\langle y(x)\right\rangle + z' - \left\langle z'\right\rangle\right)^2 \right\rangle \\
&= \left\langle \left(y(x)-\left\langle y(x)\right\rangle\right)^2 + \left(z'-\left\langle z'\right\rangle\right)^2 + 2\left(y(x)-\left\langle y(x)\right\rangle\right)\left(z'-\left\langle z'\right\rangle\right) \right\rangle \\
&= \left\langle \left(y(x)-\left\langle y(x)\right\rangle\right)^2 \right\rangle + \left\langle \left(z'-\left\langle z'\right\rangle\right)^2 \right\rangle + 2\left\langle \left(y(x)-\left\langle y(x)\right\rangle\right)\left(z'-\left\langle z'\right\rangle\right) \right\rangle \\
&= \mathrm{Var}\left[y(x)\right] + \mathrm{Var}\left[z'\right]
\end{aligned}
\qquad . \qquad (7.5)
$$

The last step of dropping the covariance term is allowed because when two distributions are independent

$$
\left\langle \left(z_1-\left\langle z_1\right\rangle\right)\left(z_2-\left\langle z_2\right\rangle\right) \right\rangle = \left\langle z_1-\left\langle z_1\right\rangle \right\rangle \left\langle z_2-\left\langle z_2\right\rangle \right\rangle = 0 \qquad (7.6)
$$

so our final results for the forecast y' at x are

$$
\begin{aligned}
\left\langle y'\right\rangle &= y(x) = \sum_j a_j X_j(x) \\
\mathrm{Var}\left[y'\right] &= \sum_{jk} X_j(x) C_{jk} X_k(x) + \sigma'^2
\end{aligned}
\qquad . \qquad (7.7)
$$

If σ' is unknown it should be set to the same scale as historical measurement errors. If they are unknown, σ' should be estimated from (6.2).

8. Implementation

A sample implementation of the above method, using a polynomial fitting function is included in the DOS program `dpolyfit.exe`. It is designed to make continuously updated parameter fits from data in the input stream, and output these parameters, or forecasts derived therefrom, to the output stream. It takes as a command-line parameter an initialization file containing preset parameters for the input format, fitting procedure, and output display. Common usages of this program are:

```
dpolyfit.exe inifile.ini <in.dat >out.dat
dpolyfit.exe inifile.ini <in.dat >>out.dat
```

Figure 8.1 Sample command-line parameters for program, where "inifile.ini", "in.dat", and "out.dat" are replaced with desired configuration, input and output files, respectively. The second version appends output to "out.dat" rather than overwriting it.

The behaviour of the program is controlled by the initialization file which contains the following options:

```
[Input]
Errors=No                ; input σ values?

[Fit]
Memory=14                ; effective # data points, N_eff in (5.3)
Parameters=7             ; # of parameters a to fit, M in (2.4)

[Output]
Input=Yes                ; print input (x,y,σ) triplet to output?
Parameters=No            ; print parameters a and errors to output?
Forecast=Yes             ; print forecasted y to output?
Forecast Distance=0      ; forecast y at x=x0+"Forecast Distance"

[Abort]                  ; end program when all of the following are true:
x=0
y=0
sig=0                    ; only compare σ if "[Input] Errors=Yes"
```

Figure 8.2 Sample initialization file "inifile.ini".

If "`[Input] Errors=Yes`" then the program expects three numbers per data point (x_0,y_0,σ_0), each separated by white spaces (eg. "`1.0  1.2  0.4`"). Otherwise, only two are expected (x_0,y_0).

The option "`[Fit] Memory=...`" contains N_{eff} as in (5.3). Generally, this must be strictly >0, but if a negative number is entered it is interpreted as $N_{eff} = \infty$, producing a traditional fit over all data points with no rescaling of the measurement errors.

If "`[Output] Input=Yes`" then the input values (x_0,y_0,σ_0) are reproduced as the first three columns of the output file. Even if σ_0 is unspecified, a best estimate is output, based on (6.2).

If "`[Output] Parameters=Yes`" then the next $2M$ columns of output are the parameters **a** and their respective uncertainties, eg. "a_1 δa_1 … a_M δa_M".

If "`[Output] Forecast=Yes`" then the program generates another two columns of output. It calculates the forecast of y at $x=x_0+$"`[Output] Forecast Distance`" from (7.7) and prints out the forecasted y and its uncertainty.

The program ends when the input (x_0,y_0,σ_0) or (x_0,y_0) matches the values in "`[Abort] x=... y=... sig=...`" or "`[Abort] x=... y=...`", respectively.

The design of this program might cause some confusion: even though the input data file is a complete set of data, the program only analyzes each data point successively, and all output is based on just this point and prior data. For example, forecasts are based only on past data, so by the end of the output file, all data points are considered but in the beginning results are based only on a single data point. The technique derived herein is better suited to time series with slowly accumulating data, rather than a complete data set on startup.

9. Summary

As in traditional least-squares methods, by differentiating the χ^2 merit function (2.5) of a linear model (2.4) we were able find a set of linear equations (2.12) which allowed us to solve for the optimal parameters which minimize χ^2. By scaling up the error tolerance of old data as new data is acquired (2.2) we were able to compact the information such that only an $M \times M$ matrix $[\alpha]$ (4.1), an $M \times 1$ vector $[\beta]$ (4.2), and a scalar δ (4.5) need be stored to retain the full history of accumulated data. We found that the inverse of $[\alpha]$ (3.1) held the covariance matrix of the fitted parameters (3.7) and (3.8). We were able to compute a *memory* for this technique (5.3) which is comparable to the number of data points used in traditional least-squares fitting. By assuming a good fit we were able to estimate the measurement uncertainties if they were unknown, and apply this to reconstruct reasonable deviations in the fitted parameters (6.3). Finally, by extrapolating with the latest parameter estimates (assuming they were valid in the forecasted domain), we were able to make forecasts and estimate their uncertainties (7.7).

As my list of references will attest, I have done virtually no research to see whether the *discounted least-squares* method has already been discovered (as is undoubtedly the case) and what name it goes by. In deriving this, I did not care if I "reinvented the wheel" because my goal was to introduce myself to some of the statistical techniques, particularly those used in the derivation of uncertainties.

10. References

Press W H, Teukolsky S A, Vetterling W T and Flannery B P (1992) *Numerical Recipes in C: The Art of Scientific Computing, Second Edition* (Cambridge University Press)