Modeling for Parameter Estimation with a Compartmental Model Example

Gary D. Knott, Ph.D.
Civilized Software, Inc.
12109 Heritage Park Circle
Silver Spring MD 20906
Tel.: (301)-962-3711
email: knott@civilized.com
URL: www.civilized.com

abstract: The practical issues involved in parameter estimation via weighted least-squares minimization are reviewed, with emphasis on fitting compartmental models. An example is given and solved using the *MLAB* mathematical and statistical modeling package. The problems of weight generation and of guessing an initial starting point are discussed and various heuristic methods for dealing with these problems are presented. Finally, the pitfalls of round-off error, algorithm instability, algorithm ineffectiveness, and ill-conditioning are described, together with some amelioration devices.

Parameter estimation methods generally consist of minimizing (or maximizing) some *objective function.* The case of maximum likelihood estimation entails maximizing a so-called *likelihood* or *log-likelihood* objective function. The case of weighted least-squares estimation entails minimizing a so-called *weighted sum-of-squares* objective function. (Note that computing x to maximize a function $f(x)$ is the same as computing x to minimize $-f(x)$, so we may focus on minimization algorithms without loss of generality.) Most, but not all, minimization algorithms are variations of the Newton iteration method derived from introducing a quadratic model for the objective function at each iteration.

Suppose we have a vector of random variables, $e = (e_1, e_2, \ldots, e_n)^T$ which represent "errors" in some observations. Let y be a vector of random variables, $y = (y_1, y_2, \ldots, y_n)^T$, representing the observations such that

$y_i = b_0 x_{i0} + b_1 x_{i1} + \ldots + b_k x_{ik} + e_i$, where $b_0, b_1, \ldots, b_k$ are unknown constants, and $x_{i0}, \ldots, x_{ik}$ are specified values. The random variable y_i is merely a translation of the random variable e_i. Often $x_{i0} = 1$, and in general $x_{i0}, x_{i1}, \ldots, x_{ik}$ may be defined in terms of some lesser number of independent variables. For example, x_{i1} may be an independent variable, and $x_{i0} = 1$ and $x_{i2} = x_{i1}^2$.

Define the vector $b = (b_0, b_1, \ldots, b_k)^T$, and the $n \times (k+1)$ matrix

$$X = \begin{bmatrix} x_{10} & x_{11} & \ldots & x_{1k} \\ x_{20} & x_{21} & \ldots & x_{2k} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n0} & x_{n1} & \ldots & x_{nk} \end{bmatrix}.$$

In matrix terms, we have $y = Xb + e$. This defines each random variable y_i as a linear expression in $x_{i0}, x_{i1}, \ldots, x_{ik}$ plus an error term e_i.

Any stochastic relationship between the random variable y_i and the values $x_{i0}, \ldots, x_{ik}$ can be expressed by the choice of the random variable e_i. Note that $E(y) = Xb + E(e)$. Suppose that $E(e) = 0$. In this case $E(y) = Xb$ and we say that $y = Xb + e$ is a *linear model* for y, since the random variable y_i is a linear function of the constants $b_0, \ldots, b_k$ and the random variable e_i.

Now, given X and estimates, $\tilde{y}$, of $E(y)$, we wish to estimate the constants, b, which are the parameters of interest.

Let $\text{cov}(e) = V$, so that V is a symmetric $n \times n$ matrix with $V_{ij} = \text{cov}(e_i, e_j)$. Usually we assume V is a diagonal matrix. Then we may define the weighted sum-of-squares objective function $S(b) := (y - Xb)^T V^{-1} (y - Xb)$. $S(b)$ is a random variable which depends upon the parameters b. Note when b is the parameter vector such that $E(y) = Xb$, then $S(b) = e^T V^{-1} e$. Now define $\hat{b} = (\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_k)^T$ as the vector of random variables such that $\hat{b} = (X^T V^{-1} X)^{-1} X^T V^{-1} y$. The vector $\hat{b}$ is the solution to the system of equations $\partial S / \partial b = 0$, so that $E(S(\hat{b}))$ is minimal. $\hat{b}$ is our *estimator* of b.

Often we have a situation where $E(e) \neq 0$ when we choose $y = Xb + e$. In this case, the linear model is not appropriate, instead, we may have a real-valued function, $f(x; b)$, of vector arguments x and b, such that $y_i = f(x_i; b) + e$ with $E(e) = 0$. The function $f(x_i; b)$ is in general a non-linear function of the parameters $b = (b_0, b_1, \ldots, b_k)^T$, as well as the arguments

$x_i = (x_{i0}, x_{i1}, \ldots, x_{im})$. Note the number of parameters, $k+1$, need not be the same as the number of independent variables, $m+1$.

Our problem is still that of estimating the parameters b, given estimates $\tilde{y}$ of $E(y)$. As before we may define the vector of random variables $\hat{b}$ so that $[\partial S/\partial b](\hat{b}) = 0$ and so that $E(S(\hat{b}))$ is minimal, where $S(b) = (y - y^*)^T V^{-1}(y - y^*)$, with $V = cov(e)$, and $y_i^* = f(x_i; b)$. Note, if we have *several* separate functions with shared parameters modeling several sets of data, we can combine the separate sum-of-squares objective functions into one by defining $S(b)$ as their sum, where b is the vector whose components are the local and global parameters appearing in the separate model functions.

There are numerical methods for solving the normal equations $[\partial S/\partial b](\tilde{\hat{b}}) = 0$ given a particular starting estimate for $\tilde{\hat{b}}$ and the particular sampled estimates $\tilde{y}$ for $E(y)$. These methods are iterative in nature, and may sometimes fail to converge to a correct solution.

To solve $\partial S/\partial b = 0$, treating the random vector y as constant, we may expand $\partial S/\partial b$ in a Taylor series about a point $b^{(j)}$. Thus, we *linearize* the normal equations, to obtain:

$$0 = [\partial S/\partial b](b^{(j)} + \beta^{(j)}) = M^{(j)}\beta^{(j)} + w^{(j)} + O(|\ \beta^{(j)}\ |^2)$$

where $M^{(j)} = [\partial^2 S/\partial b^2](b^{(j)})$, $w^{(j)} = [\partial S/\partial b](b^{(j)})$, and $\beta^{(j)} = (\beta_0^{(j)}, \ldots, \beta_k^{(j)})^T$.

Now the *Newton-Raphson procedure* is the iteration formula:

$b^{(j+1)} = b^{(j)} - \pi(M^{(j)})^{-1}w^{(j)}$, where π is a parameter, usually near 1. Under appropriate conditions this iteration will converge to a vector, b^*, such that $[\partial S/\partial b](b^*) = 0$.

The *Method of Steepest Descent* is: $b^{(j+1)} = b^{(j)} - \pi w^{(j)}$, where π is the minimum of S along the antigradient, given by $\partial S(b^{(j)} - \pi w^{(j)})/\partial \pi = 0$.

Davidon's quasi-Newton method is: $b^{(j+1)} = b^{(j)} - \pi H^{(j)}w^{(j)}$, where π is determined as in the method of steepest descent, and $H^{(j)}$ is a positive-definite matrix such that $H^{(j)}w^{(j)} - w^{(j-1)} = b^{(j)} - b^{(j-1)}$, and $H^{(0)} = I$.

Another approach to solving $\partial S/\partial b = 0$ is to *linearize the model* to obtain linear normal equations. The *Gauss-Newton procedure* uses this idea iteratively. We have the model $y_i = f(x_i; b) + e_i$, and we may expand f in a Taylor series to obtain:

$f(x; b+\beta) \approx f(x,b) + (\partial f(x;b)/\partial b)^T \beta$, where $\beta = (\beta_0, \ldots, \beta_k)^T$.

Define $g(x;\beta) = (\partial f(x;b)/\partial b)^T \beta$. Now, for any point b, we suppose β is the correction vector such that $f(x_i, b+\beta) = E(y_i)$. Thus $g(x_i;\beta) \approx E(y_i) - f(x_i;b)$, so $\beta \approx (X^T V^{-1} X)^{-1} X^T V^{-1}(E(y) - (f(x_1;b), \ldots, f(x_n;b))^T)$, where $X_{st} = \partial f(x_s;b)/\partial b_t$, with $1 \leq s \leq n$, and $0 \leq t \leq k$. Now β depends upon the guess, b. Thus we have the iteration: $b^{(j+1)} = b^{(j)} + \beta^{(j)}$, where $\beta^{(j)} = (X^T V^{-1} X)^{-1} X^T V^{-1}(E(y) - (f(x_1;b^{(j)}), \ldots, f(x_n;b^{(j)}))^T)$, with $X_{st} = \partial f(x_s;b^{(j)})/\partial b_t$.

The matrix $X^T V^{-1} X$ may be singular or ill-conditioned; moreover the approximation $f(x;b+\beta) \approx f(x;b) + g(x;\beta)$ may hold only for vectors, β, which are very small. The *Marquardt-Levenberg method* is a variation of the Gauss-Newton procedure which deals with these difficulties.

Let D be a diagonal $n \times n$ matrix with $D_{ii} = (X^T V^{-1} X)_{ii}$. Now define:

$$\beta = (X^T V^{-1} X + \varepsilon D)^{-1} X^T V^{-1}(E(y) - (f(x_1;b), \ldots, f(x_n;b))^T).$$

The matrix $X^T V^{-1} X + \varepsilon D$ is non-singular for $\varepsilon > 0$ and its condition depends upon ε.

Now, the Marquardt-Levenberg iteration is: $b^{(j+1)} = b^{(j)} + \beta^{(j)}$, where $\beta^{(j)} = (X^T V^{-1} X + \varepsilon D)^{-1} X^T V^{-1}(E(y) - (f(x_1;b^{(j)}), \ldots, f(x_n;b^{(j)}))^T)$,
with $X_{st} = \partial f(x_s;b^{(j)})/\partial b_t$, and $D_{st} =$ if $s = t$ then $(X^T V^{-1} X)_{st}$ else 0.

For $\varepsilon = 0$, we have the Gauss-Newton procedure, while for $\varepsilon \to \infty$, we obtain a vanishing correction vector, $\beta^{(j)}$, in the direction of steepest descent. When the matrix $X^T V^{-1} X$ is ill-conditioned or of deficient rank, the magnified-diagonal property of the Marquardt-Levenberg method usually overcomes this difficulty. Unless we start at or exactly hit a maximum, we will not be troubled by the fact that $[\partial S/\partial b](\hat{b}) = 0$ at both minima and maxima, since we always require that a "downhill" step be preferred.

Our approach employs the Marquardt-Levenberg method, with $\varepsilon = 10^{-10}$ initially. Rather than keep ε fixed, at each iteration ε is varied so as to seek out a smaller $S(b)$ for b lying on an appropriate curve between the current estimate of b and the next estimate predicted by the Gauss-Newton iteration. This is a form of line-search with a curved "line". The evaluations of $S(b)$ involved for different trial values of ε constitute subiterations.

The practical issues involved in parameter estimation are:

1. Choice and justification of the mathematical model.

2. Choice and justification of the statistical error model.

3. Choice of the initial parameter values (guesses) and the "guidance control" of the minimization process.

Let us consider these issues in the context of a concrete example. The following data and model was proposed by Nicholas Holford as a challenge for compartmental modeling software packages. The data is widely disparate in scale, and we discuss below how to use weight vectors to handle this. There is also missing data at different time points.

We shall employ the *MLAB* mathematical and statistical modeling system[1][2] for this challenge problem. The solution discussed herein shows several important features of MLAB. These features include simultaneously fitting several functions with shared parameters to different data sets. The functions which make up the model are defined by a set of differential equations. These differential equations turn out to be stiff and thus require a suitable implicit method such as *Gear*'s method[3] to solve them numerically in a reasonable amount of time.

The problem setting is as follows: 48.15 milligrams of a drug D is given by mouth, and blood concentrations of the drug D and also of its only metabolite M are measured. Also the cumulative amounts of D and M in the urine are measured. Thus, we have the following data.

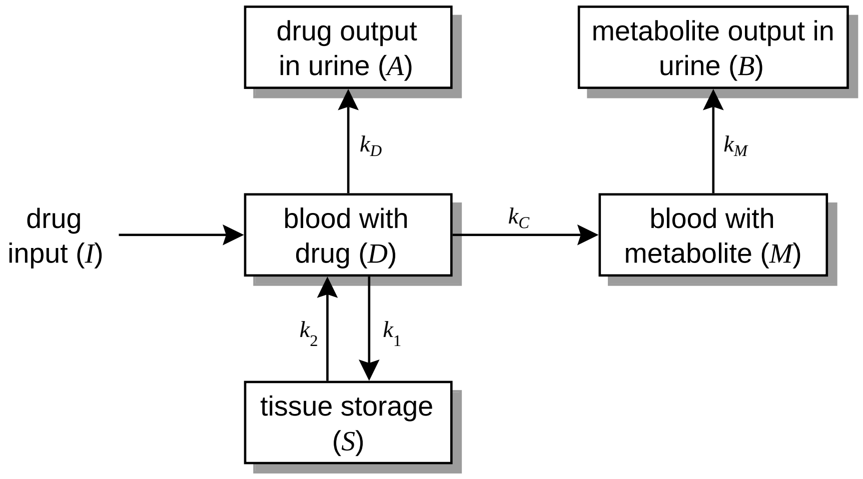
time (hours)	blood-D (mg/liter)	blood-M (mg/liter)	urine-D (mg)	urine-M (mg)
.82	.1746	.822051		
1			1.87	7.23
1.2	.166	1.143786		
1.4	.1264	1.152462		
2	.1092	.859647	3.23	15.53
2.4	.0904	.648531		
2.9	.0828	.601536		
3			4.02	21.15
3.38	.0704	.381744		
3.92	.0591	.402711		
4			4.59	25.88
4.42	.0511	.30366		
5.18	.0355	.252327		
6			5.77	32.42
6.35	.0148	.143154		
8			6.3	34.89
8.3	.0081	.063624		
10			6.51	36.16
10.28	.0047	.033258		
12			6.65	37.06
12.4	.0026	.020967		
24			6.92	38.7
24.57	.0009	.006507		
48			7.3	40.29
72			7.38	40.77

Note: In the data table above, blanks represent missing data. In order to prepare the data for input, some value must be supplied at each place where a number is missing. Any unique value may be used for these missing values since we will remove them later. For this example, zero will be entered for the missing table values. *MLAB* handles the problem of missing data automatically during fitting (zero weights are generated internally to correspond to missing data).

The errors in the blood concentration measurements have variances which are roughly proportional to the squares of the true measurement values. The errors in the urine amounts have more-nearly constant variances. Whatever model we use to predict $D(t)$ (blood drug concentration at time t), $M(t)$

(blood metabolite concentration at time t), $A(t)$ (urine drug amount at time t), and $B(t)$ (urine metabolite amount at time t), we will want to weight our observations by weights which are proportional to the reciprocals of the variances.

A compartmental model for the uptake, metabolic conversion and excretion of this drug is given below, and we wish to curve-fit to adjust the model to fit the observed data.


drug output in urine (*A*)
metabolite output in urine (*B*)
k_D
k_M
drug input (*I*)
blood with drug (*D*)
k_C
blood with metabolite (*M*)
k_2
k_1
tissue storage (*S*)

This is a highly simplified model; the path from the blood drug compartment to the blood metabolite compartment should probably include a metabolic conversion compartment, and perhaps the metabolite should go in and out of tissue as does the drug D, but this five-compartment model is already near the limit of what we can usefully fit to the data.

Let V_B be the volume in liters of the blood and let V_S be the volume in liters of the tissue of the subject being studied. We let V_M denote the volume in liters of the blood-metabolite compartment; we would expect that $V_M = V_B$, but we may obtain a better fit when this constraint is not honored.

Let $D(t)$ = the concentration of the drug D in the blood at the time t, let $M(t)$ be the concentration of the metabolite M in the blood at time t, let $S(t)$ be the concentration of the drug D in the tissue at time t. Also, let $A(t)$ be the cumulative amount of drug D which has appeared in the urine by time t, and let $B(t)$ be the cumulative amount of the metabolite M which has appeared in the urine by time t.

We can define the functions that comprise the above compartmental model by specifying a first-order ordinary differential equation for each compartment.

$$\begin{aligned}
D' &= (I(t) - (k_1 + k_D + k_C)D + k_2 S)/V_B \\
S' &= (k_1 D - k_2 S)/V_S \\
M' &= (k_C D - k_M M)/V_M \\
A' &= k_D D \\
B' &= k_M M
\end{aligned}$$

with $D(0) = 0$, $M(0) = 0$, $S(0) = 0$, $A(0) = 0$, and $B(0) = 0$.

The choice of the input function, I, is somewhat arbitrary. However, if the drug is absorbed as fast as it passes at a constant rate into the small intestine, we may choose $I(t) = \mathit{if}\, t < E_T \ \mathit{then}\ 48.15/E_T \ \mathit{else}\ 0$, so 48.15 mg of the drug is introduced at a constant rate over E_T hours. The absorbance rate of the drug is not known; thus we shall let E_T be a fitting parameter. If we needed a continuous input function, we could instead use the form $I(t) = 48.15 \cdot H \cdot \exp(-H \cdot t)$, and introduce the constraint $H > 0$. It turns out this makes little difference in the final results. Another method of handling the input, which is especially suitable for a bolus injection of an unknown amount of drug, is to specify the initial condition for D as $D(0) = d_0$, and then make d_0 a fitting parameter. This requires that initial values can be adjusted by our curve-fitting software, as is possible in *MLAB*.

Note that k_M, k_C, k_1 and k_2 are in units of liters/hour, the derivatives D', S', and M' are in units of mg/liter/hour, A' and B' are in units of mg/hour, D, S, and M are in units of mg/liter, A and B are in units of mg, V_B, V_S and V_M are in units of liters, and $I(t)$ is in units of mg/hour, and these units are dimensionally consistent.

Like almost all compartmental models, this is a gross over-simplification of reality. In many cases, such as chemical kinetics modeling, it is often possible to employ a theoretically-justifiable model. Physiological compartmental models however, are necessarily inaccurate in regard to mechanism and spatial mass distribution and flow. In spite of their theoretical inadequacy, compartmental models can be useful descriptive and predictive tools, as long as they are not mistaken as physically and mechanistically-complete and valid models.

The model above is not the minimal model that might be usefully employed for the given data. Indeed it is near the limit of complexity that can be usefully fit to the given data. The complexity of a compartmental model is generally a function of the number of compartments and the number of edges indicating flow between compartments, which together determine the number of unknown parameters to be estimated. A further determinant of complexity is the interconnection density which is measured by the number of alternate paths that connect separated compartments. Part of the art of compartmental modeling consists of judging the trade-offs between model adequacy versus model complexity in light of the measured data, the types and sizes of errors in the observations, and the research goals motivating the modeling being undertaken.

The next concern is that of the nature of the measurement error in our data. Each observed value $\tilde{y}_i$ can be considered to be the sum of the "true" value $E(y_i)$, plus an error value $\tilde{e}_i$ which may be taken as a sample of a random variable e_i. Ideally we would like to know the joint distribution of the error random variables e_i so that maximum likelihood estimation can be employed when desired. Usually, however, such distributional information is not known, although the investigator's experience can often play a role here.

Maximum likelihood estimation is often the preferred tool when suitable non-elliptically-contoured distribution functions are plausibly known; in this case, we could use the *MLAB* `maximize` operator. The results of weighted least squares minimization are generally close to the results of maximum likelihood estimation when the observational errors are independent and approximately normal with 0 mean. It is important to remember, however, that least-squares estimators can be *inconsistent* for some non-normal error distributions. With normal error, when the model is linear in the parameters, the least squares estimators are the *same* as the maximum likelihood estimators, provided that the correct weights are used; and these estimators are themselves jointly normally-distributed.

For least squares estimation, it is important that the error in the independent variable or variables be neglible. When both the independent and dependent variables are measured with error, we can sometimes "assign" all the error for a data point to reside with the dependent variable value, but usually the correlation present between such errors makes this assignment of error problematical. One important example where such correlated errors occur is in fitting the Scatchard model to ligand binding data[4]. The Scatchard model predicts the ratio of bound ligand concentration to free ligand concentration as a linear function of bound ligand concentration; typically all of these quantities are measured with error, and the error is clearly correlated. In this case, the difficulties can be overcome by using an alternate (non-linear) saturation model which relates the total amount of ligand present to the amount bound; the total amount of ligand can generally be measured much more accurately than can the amount of bound ligand.

This device of transforming the model and the data finds use in the situation where the error in a variable deviates substantially from normal. For example, if an observed variable is seen to behave approximately like a shifted log-normal random variable, we may use a logarithm transformation to obtain the more desirable situation where the error in the transformed data appears to be approximately normal. The opposite effect must also be taken into account; when data is transformed for more convenient modeling (either mathematically or physically via a different measuring procedure); the error may be adversely affected, *i.e.* made noticeably non-normal. Algebraic transformations can also be useful in other ways. For example, if a parameter K is necessarily positive, it may be more convenient to substitute $\exp(L)$ for K, and estimate L, because the least-squares estimator for L is likely to be distributed more like a normal random variable than is the least-squares estimator for K, and thus the linear normal error theory is more likely to be of use.

The most fundamental consideration of error, that we ignore at our peril, is the issue of *scale*. We need to take account of the varying magnitudes of the errors that often occur in observations of distinct variables or of the same variable at different independent variable values. In the data given above, the magnitudes of the errors in the drug-in-blood data are much larger than the magnitudes of the errors in the metabolite data. In this case, the difference is mainly due to the differing units of measurement used (molar concentration vs. grams per milliliter), but often differing scales of error magnitude are unavoidable. The purpose of weights in least squares parameter estimation is exactly to accomodate the differing error magnitudes that may arise. We want to *weigh* the observation $\tilde{y}_i$ with the reciprocal of the variance of the associated error random variable e_i; thus we wish to assign the weight $1/Var(e_i)$ to the observation $\tilde{y}_i$. Some such weighting is necessary when we have data with error of substantially different magnitudes.

Even when the data of greatest magnitude is accurately measured, we may wish to use weights which give data with lower magnitudes a chance to be fit;

the use of such artifical weights are useful and defensible when the model is known to be inexact; we might say that the error due to the model is "transferred" to the high magnitude data by using lowered weights.

There are several useful ways to estimate the variances of a sequence of observations. Often we may know from experience that the standard deviation of an observation $\tilde{y}_i$ is adequately modeled by a known constant, or by a fixed fraction or some other function h of the unknown true value $E(y_i)$. In this latter case, we might estimate the variance to be associated with the observation $\tilde{y}_i$ as $h(\tilde{y}_i)^2$. In *MLAB*, it is also possible to iteratively change the weights during curve-fitting so that a function of the form $h(f(x_i), x_i, \tilde{y}_i, i)$ is used as our *dynamically changing* variance estimate, where x_i is the independent variable vector associated with the observation $\tilde{y}_i$ and f is the model function being fit to the data points $(x_i, \tilde{y}_i)$. This device goes by the name *iterative reweighting*; it can be used for various purposes such as implementing "robust" curve-fitting algorithms.

When we have enough data, we may estimate the associated variance values by computing a moving standard deviation with respect to a sliding "window" passing over the data; this entails computing a moving mean as well. Both the moving mean and the moving standard deviation can be computed with a weighting function applied to the data in the window so as to give more importance to the central points within the window interval and to insure that the resulting moving standard deviation curve is continuous in theory. Moreover we may smooth the resulting moving standard deviation curve using various schemes when this is desired. There are many slight variations of this basic idea for generating variance estimates automatically, having to do with the scheme used to compute the non-parametric central trend curve. Besides weighted moving averages, moving linear or quadratic models can be successively fit to our data sequence, smoothing splines or kernel estimation methods can be used, and even moving medians can be useful for some data sets.

For our example problem, we will use the *MLAB* `EWT` operator which employs the deviations from a smoothed form of the data to estimate the errors in data values. Using `EWT` on the various data sets produces weights based on error estimates scaled comparably to the underlying error of the data sets themselves. This has the effect that the deviations will each be approximately sized so that each of our sets of observed data is given more or less correct weight in the sum of squares to be minimized.

Once the weights or the weighting model is established, we have only a minimization problem to solve, with the proviso that we can impose all necessary constraints on the admissable parameter values. Constraints are an important part of many curve-fitting problems. Sometimes, the constraints are physically dictated; certain parameters *must* be positive (or negative), or must lie in some fixed range. Other times the constraints that we impose are "plausibility" constraints; for example, the investigator's judgment may dictate that a certain parameter is limited to lie in a certain range.

We may want to impose constraints to guide the curve-fitting process; such constraints may be dropped in a final "fiduciary" curve-fit once we have discovered the region of interest in parameter space. This use of constraints may be considered to be a tactical device for obtaining a good starting point in parameter space rather than as part of the model itself.

Non-linear constraints such as $a^2 + b^2 < 10$ pose special difficulties. Most systems allow only linear constraints (if that), and non-linear constraints must be generally implemented "by hand" by including a *penalty function* term in the model. For example, if we are fitting the model $f(t; a, b)$ to the data points (t_i, v_i) with the constraint $a^2 + b^2 < 10$, we may minimize the objective function $\sum_{i=1}^{n}(v_i - f(t_i; a, b))^2 + \max(0, , a^2 + b^2 - 10)^8$; whenever $a^2+b^2 > 10$, the penalty term becomes increasingly-positive, and a suitably-sensitive search process will move back into the region in parameter space where $a^2 + b^2 < 10$.

It is necessary to use constraints for fitting our example; without them, the parameters may well be assigned foolish values where the differential equations cannot be integrated numerically. Let us assume the following set of constraints.

$$\{ .1 < E_T < 70,\ 0 < k_1,\ 0 < k_2,\ 0 < k_D,\ 0 < k_C,\ 0 < k_M,\ 3 < V_B < 7, \\ 10 < V_S < 100,\ V_M > .01\}$$

Once we have established our model and weights, we can undertake the parameter estimation process. This can be easy or difficult depending upon the nature of our weighted least-squares objective function and the initial starting point chosen for the vector of parameters. Moreover this entire process is often provisional (we may be exploring various weights or various models), and may want to be repeated with revisions in the weights or the model or in certain other controlling choices.

Guessing the initial starting point "correctly" is sometimes vital to obtaining a sucessful fit. But most often the initial starting point makes little difference; the shape of the objective function is essentially bowl-shaped, albeit with some irregular features, and we will “roll” to the minimum at the bottom of the bowl from any reasonable initial starting point. Indeed, for a linear model, the quadratic approximation used with Newton-based minimization methods is exact, and we will reach the minimum immediately. For some non-linear models, however, the objective function has various complicated features such as several separate “bowls” and/or various ridges and valleys that make starting from a good guess important. Even when we have a good initial starting point, the progress to the desired minimum can be slow; a “banana”-shaped bowl is not well-modeled by an elliptical bowl, and the fitting process may zig-zag slowly toward the minimum.

Choosing an initial starting point for a sensitive problem is generally an iterative process; we usually must try several starting points and thereby gradually learn about our objective function. Knowledge of the physical meaning of parameters can be very helpful, but this becomes less important as the model deviates from reality.

There are a variety of tricks for obtaining “good” parameter-value guesses; the unifying theme is to start with a simplified model and/or simplified data for which guessing the initial starting point is easier, and then to reintroduce the complexities of the original situation bit by bit. This can be formally done by using a homotopic mixture of our desired model and some simplified model, and then gradually adjusting the mixing coefficient; this is a kind of *continuation* device. Often it suffices to fix some parameters and adjust the others, and then vary the fixed parameters in a subsequent fit. For example, it can be useful to iteratively fix the non-linear parameters, adjust the linear parameters (generally with no difficulty,) and then adjust all the parameters, repeating this process as necessary. Occasionally, we may want to modify the data or the weights being used, or even “make up” a set of data so as to guide the fitting process toward a desired solution; finally, of course, we undo all such data modifications for a final fit.

It is often useful to do random probes and graph the resulting model together with the associated data; even though the parameter space is huge and of high dimension, an investigator’s intuition is sometimes sharpened by looking at a few such graphs. Indeed, one should *always* graph the data and the model as determined by the initial starting point in order to visually check the data and to verify that the model is correctly formulated for computation. Sometimes initial starting points can be usefully obtained by using minimization methods such as the simplex method or some form of simulated annealing.

For compartmental models, we can often use a simplified model by dropping some edges or some compartments and then estimate the remaining parameters. Then we can return to our original model, and make more informed guesses of its parameters. For example, in our problem, we might drop the tissue storage compartment, estimate the remaining parameters, and then reintroduce it, knowing that the effect of this reintroduction is to reduce the estimated flow rate parameters going out of the drug-in-blood compartment. Another device for compartmental models with the parameters occurring linearly in the associated differential equations, due to Judah Rosenblatt[5], is as follows. To use Rosenblatt's device, we must have data for each function (*i.e.*, each compartment) in our model; if needed, we might "make-up" such data. Now, we fit the data we have with smooth curves (*e.g.* smoothing splines), and then differentiate those curves to get derivative function curves for each model function. With these smooth curves for both the model functions and their associated derivative functions, our system of differential equations becomes an over-determined system of *linear* equations, and we may solve the corresponding normal equations to obtain the initial starting parameter values that we are seeking.

When we are fitting a distribution function model to data, we can make use of old tricks like using the formulas for the moments, and fitting them to the computed moment estimates. For example, Suppose we are fitting a family of gamma distributions determined by a covariate a to data, so that the model is $F(s;a) = G(s;\alpha_1(a),\alpha_2(a),\alpha_3(a))$, where $G(s;m,b,c)$ is the general Gamma distribution function, with $G(s;m,b,c) := \Gamma(m)^{-1}\int_{0\le t\le \frac{s-c}{b}} t^{m-1}e^{-t}dt$ for $s > c$ and 0 for $s < c$. The associated mean is $mb + c$, and the associated variance is mb^2. Note the Gamma-distribution parameters m and b must both be non-negative values. Suppose the functions $\alpha_1(a), \alpha_2(a)$, and $\alpha_3(a)$ are all taken as quadratic expressions with distinct coefficients, so that $\alpha_i(a) = \lambda_{1i}a^2 + \lambda_{2i}a + \lambda_{3i}$ for $i = 1, 2, 3$. In order to obtain an initial estimate for the coefficients λ_{ji}, we may simultaneously fit $\alpha_1(a)\alpha_2(a) + \alpha_3(a)$ to the appropriate sample means, and $\alpha_1(a)\alpha_2(a)^2$ to the appropriate sample variances. We may now use the obtained λ_{ji} values and refine these estimates by fitting F to the ensemble of empirical cumulative distribution values obtained from the data.

The curve-fitting process is essentially characterized by making sucessive informed guesses for the starting parameter values and accepting those parameter value estimates which please us the most; generally, *but not necessarily always*, these estimates correspond to the least value for our sum-of-squares objective function. Note that when, as in the case of a compartmental model, we have a model involving functions defined by differential equations, we must numerically solve these differential equations in order to compute a single value of our objective function. Also, for most minimization algorithms, the partial derivatives of the objective function with respect to each of the parameters must be occasionally computed. When these partial derivatives can be obtained symbolically, it is useful to do so, since this avoids the common problems associated with numerical differentiation that sometimes arise. (*MLAB* computes such symbolic derivatives automatically for non-differential equation models.) In the case of a differential-equation-based model, it is necessary to use numerical methods, either by numerically solving an auxillary system of differential equations, or by solving our given system of differential equations twice for the purpose of computing a centered-difference numerical derivative estimate. Clearly then, curve-fitting a differential-equation-based model can be a computationally demanding process.

As with any complex computational process, we may be the victims of (1) round-off error, (2) algorithm instability, (3) algorithm ineffectiveness, and/or (4) ill-conditioning. Round-off error often arises when we have terms of widely-disparate magnitude arising in our objective function; data of differing magnitudes can also be a cause of round-off error.

One particularly egregious round-off error problem is the situation where numerical overflow occurs; this happens when we generate a number whose magnitude is greater than the computers' floating-point representational capacity. (On a PC, approximately $1.7 \cdot 10^{308}$ is the largest directly-representable value.) Overflows sometimes occur during curve-fitting when an unsuitable parameter vector is generated. Many computer programs either stop or produce an "infinity"-code which fatally contaminates all subsequent calculations. It is preferable to produce the largest *computationally-valid* number of the correct sign, since even excessive round-off error is better than termination, because, if allowed to proceed, the search process may well eventually go in a more felicitous direction into a region of parameter space where overflows do not occur, so that the transient occurrence of overflows is inconsequential. It is often unnecessarily hard (and sometimes impossible) to program this behaviour in a practical manner, but it is an effective adjunct for some problems, and MLAB handles overflows as described above on most processors.

Algorithm instability occurs when the values produced by various computations are garbage because certain assumptions are violated or because certain iterations do not converge. One important example is when the algorithm used for solving differential equations introduces a error-magnifying effect in the solution because the problem that we are solving lies outside the "stability region" for the numerical method being employed. It is sometimes possible to program tests which detect such problems. The most appropriate response is to switch to another numerical method which is, hopefully, more suitable to the problem at hand.

Algorithm ineffectiveness is basically the issue of speed. If it takes too long to estimate our parameters, we have gone outside the class of problems for which our methods are effective. Sometimes this is merely a matter of size; all methods are ineffective when the number of parameters to be estimated becomes too large or when the objective function becomes too complicated. One common situation that occurs with compartmental models is that of a *stiff* system of differential equations. A stiff system of differential equations is a system with at least one "super-stable" equation which requires many very small steps to track the solution accurately with so-called explicit methods. In such a circumstance, there are other *implicit* methods[3] which sometimes allow the use of a large-enough step-size to restore effectiveness. Since the overhead of an implicit method is greater than the overhead of an explicit method, we don't want to use an implicit method routinely. It is convenient for the investigator if we program tests for stiffness and automatically switch to an implicit method that is effective for stiff differential equations when this is appropriate.

The classic situation of ill-conditioning is an inherent defect in our model, our data, or both. If the objective function, or its derivatives, are so sensitive to small changes in the parameters that we get wildly different function values for almost identical parameter values, then we have an ill-conditioned problem. This includes the case where a solution of a differential equation becomes very large in magnitude except for a small range of parameter values. Such instability generally manifests itself by overflows. This "divergent" behavior in a differential equation is the opposite of stiffness.

Ill-conditioning often manifests itself by yielding extremely non-unique parameter estimates, in the sense that small changes in the data, or in choices such as the integration error-per-step limit lead to very different parameter estimates. Geometrically, we have a “flat” place in the graph of our objective function; the parameters can move a relatively-large distance in certain directions with only a small change in the objective function, and where we end-up in this flat region is a matter of our initial starting point, and the computational errors engendered during searching. Such a flat place is modeled by an elliptical bowl defined by a quadratic form, where some of the eigenvalues of the matrix of the quadratic form are relatively very large, and some are relatively very small; this is the hallmark of ill-conditioning. Since a joint-confidence region of our estimated parameters is obtained as a contour of the objective function in the neighborhood of the minimum that corresponds to our estimated parameter vector, these eigenvalues are also useful in obtaining the linear normal theory error estimates for the estimated parameters; essentially, the less flat the approximating bowl containing the minimum is, the smaller an approximate elliptic joint-confidence region will be.

The condition of a curve-fitting problem can change as the parameter values change; our problem can be ill-conditioned in some regions of parameter space and not in others. If we are lucky, we will find suitable parameter values in a region of the parameter space where ill-conditioning does not occur. If we are unlucky, however, we may be driven to a sub-optimal local minimum corresponding to an unacceptable fit due to encountering transient or persistent ill-conditioning. For differential-equation-based objective functions, the occurrence of ill-conditioning may be connected to entering a region of parameter space where the character of our differential equations is markedly different, this is a *bifurcation* event occurring.

Even if none of the above issues present us with serious difficulties, we may have a minimization problem which is just plain *hard*. That is, the objective function with k parameters may correspond to a surface in $(k + 1)$-space which has many complex features such as ridges and valleys and local depressions that violate the successive quadratic (elliptic bowl) approximation approach used by Newton minimization methods.

Now let us return to our compartmental modeling example. We need initial guesses for all the parameters. These guesses must be suitable; arbitrary guesses can lead to unreasonable final fit values, or even cause the fitting process to be unable to proceed due to excessive stiffness of the differential equations!

Suppose a unit amount of drug diffuses from blood into tissue so that half of it is transfered in one hour. Then if y is the amount of drug in the blood, we have $y' = -k_1 y$ with $y(0) = 1$, and $y(1) = .5$, and so $k_1 \approx .7$. Let us also guess that $k_2 = .7$.

If half of a unit amount of drug is cleared from the blood and transferred to the urine by the kidneys in about 4 hours, then $k_D \approx .17$. Let us also guess that $k_M = .17$. Similarly, let us guess $k_C = .17$.

Finally we choose $V_B = 5$, $V_M = 5$, $V_S = 40$, and $E_T = 1$.

Now we may proceed in MLAB as follows. First we enter the data listed above, with zeros for missing values, and then we construct the corresponding weight vectors `WD`, `WM`, `WA`, and `WB` using the built-in `EWT` (estimated weights) operator. (`compress(m,2)` removes all the rows of the matrix `m` where the value in column 2 is 0.)

```
n = read(datafile, 100, 5)
tv = n col 1;  "tv = time values"
dv = n col 2;  "dv = blood drug data."
mv = n col 3;  "mv = blood metabolite data."
av = n col 4;  "av = urine drug data."
bv = n col 5;  "bv = urine metabolite data."

dv = tv &' dv;  dv = compress(dv,2); wd =ewt(dv)
mv = tv &' mv;  mv = compress(mv,2); wm =ewt(mv)
av = tv &' av;  av = compress(av,2); wa =ewt(av)
bv = tv &' bv;  bv = compress(bv,2); wb =ewt(bv)
```

Now we enter our model, our constraints, and our inital guesses.

```
function d't(t) = (i(t) - (k1 + kd + kc)*d + k2*s)/vb
function s't(t) = (k1*d - k2*s)/vs
function m't(t) = (kc*d - km*m)/vm
function a't(t) = kd*d
function b't(t) = km*m
function i(t) = if t<et then dose/et else 0

initial d(0) = 0
initial s(0) = 0
initial m(0) = 0
initial a(0) = 0
initial b(0) = 0

dose = 48.15

k1=.7;k2=k1;kd=.17;km=kd;kc=kd;vb=5;vs=40;et=1;vm=5

constraints c = {k1>0, k2>0, kc>0, km>0, et>.1, et<70, vb>3,
                 vb<7, vs>10, vs<100, vm>.01}
```

Now we proceed to fit. To reduce the amount of time needed to fit this stiff model, we use Gear's method with a tolerance of .01.

```
method = gear;
maxiter = 100
errfac = 0.01

fit(k1,k2,kc,kd,km,et,vb,vs,vm),
  d to dv with weight wd, m to mv with weight wm,
  a to av with weight wa, b to bv with weight wb, constraints c

final parameter values
      value              error              dependency    parameter
    19.55670378        12.41128315        0.7277144348   K1
    4.829029615        56.05648908        0.9970952774   K2
    95.71231349        18.05875113        0.9611351847   KC
    17.29009355        3.174400652        0.9601255658   KD
    13.57478156        2.526143866        0.6637941471   KM
    4.835844445       0.8222361079         0.945128974   ET
    4.168221663         13.3558721        0.5651377681   VB
      88.309337        930.8309603         0.997317646   VS
    3.929260682        8.161760496        0.9041479451   VM
22 iterations
CONVERGED
best weighted sum of squares = 3.054944e+02
weighted root mean square error = 2.665431e+00
weighted deviation fraction = 5.505088e-02
```

```
R squared = 9.959130e-01
no active constraints
```

Now we will graph our four data sets together with the best-fit curves produced by solving our system of differential equations with the parameter values obtained above.

Note that we must beware of assuming that our obtained parameters have any physical significance. It is unlikely, for example, that the actual compartment volumes are close to the values we have for V_B, V_M and V_S. Our model may be useful for prediction purposes, but it is not useful for gaining insight into any actual physiological mechanisms.

```
tv=0:75!120
draw points(d,tv) color brown
draw dv color red pt xpt lt none
image color white
top title " Drug concentration in Blood" font 11 size .03
left title "'-90AD" font 11 size .03
bottom title "time"
frame 0 to .5, 0 to .5
w1=w

draw points(m,tv) color blue
draw mv color blue pt octagon lt none
image color yellow; frame color green
top title "Metabolite concentration in Blood" font 11 color brown size .03
left title "'-90AM" font 11 size .03
bottom title "time"
frame .5 to 1, 0 to .5
w2=w

draw points(a,tv) color purple
draw av color red pt square lt none
image color grey; frame color brown
top title " Drug amount in Urine" font 11 size .03
left title "'-90AA" font 11 size .03
bottom title "time"
frame 0 to .5, .5 to 1
w3=w
```

```
draw points(b,tv) color brown
draw bv color blue pt crosspt lt none
image color aqua; frame color red
top title " Metabolite amount in Urine" font 11 size .03
left title "-90AB" font 11 size .03
bottom title "time"
window 0 to 80, 0 to 45
frame .5 to 1, .5 to 1
w4=w

view
```

This is not the only reasonable fit. Starting from other guesses, for example: $K_1 = 223$, $K_2 = 14.7$, $K_C = 66$, $K_D = 11.7$, $K_M = 9.7$, $E_T = 1.95$, $V_B = 6.24$, $V_S = 12.35$, and $V_M = 0.04$, we can obtain other, quite different, results; this fit is presented below. The large dependency values of the parameters indicate that this problem does not have a unique answer. Probably the problem is over-parameterized.

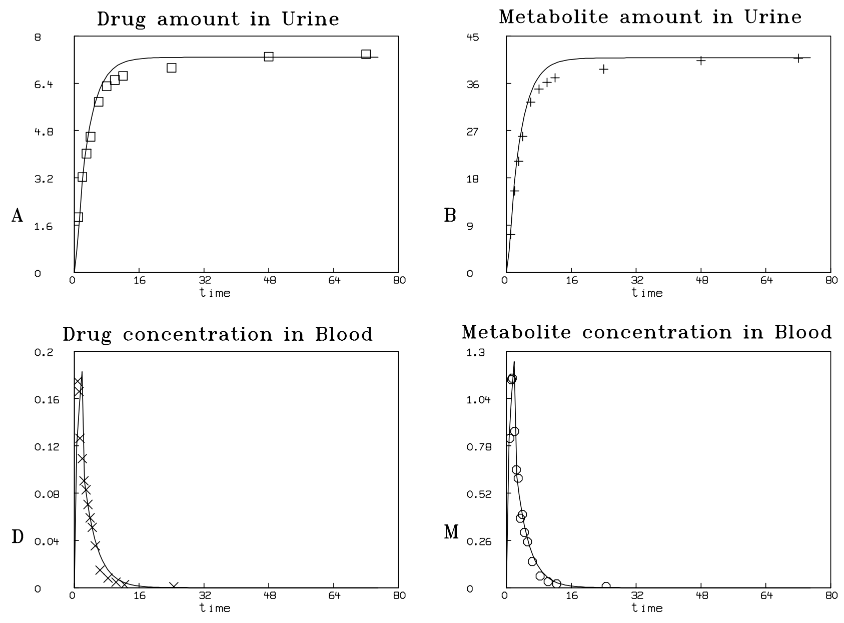
```
k1 = 223; k2 = 14.7
kd = 11.7; km = 9.7; kc = 66
vb = 6.24; vs = 12.35; vm =.04; et = 1.95;

fit(k1,k2,kc,kd,km,et,vb,vs,vm),
  d to dv with weight wd, m to mv with weight wm,
  a to av with weight wa, b to bv with weight wb, constraints c

final parameter values
      value              error              dependency    parameter
    222.3296966        27.76132266          0.8337053755   K1
     14.7359648        3117.341849          0.9999999383   K2
    65.98333895        3.836887237          0.9399933157   KC
     11.7407663        0.692533352           0.938315949   KD
    9.697523939       0.6754487655          0.6928345824   KM
    1.960006851        0.059767016          0.9654962599   ET
    5.964737228        3.990424231           0.866044195   VB
    12.31946483        2605.653695          0.9999999384   VS
  0.04208395842      0.01012174536          0.9703620604   VM
```

```
3 iterations
CONVERGED
best weighted sum of squares = 4.491649e+01
weighted root mean square error = 1.022042e+00
weighted deviation fraction = 2.993774e-02
R squared = 9.941458e-01
no active constraints
```

The graphical results of this fit are shown below. Note we fit the blood-drug and blood-metabolite concentrations more closely at the expense of urine-drug and urine-metabolite fitting, indeed this was our goal in searching for an alternate local minimum. This minimum is smaller than the first minimum we found, and seems to be a generally-preferable fit as well.



Since our model is not physically accurate, we may feel free to choose our parameter values so that the four curves are adequately predicted, without concern for the physical meanings of the parameters.

In conclusion, note that parameter estimation via curve-fitting is *not* always an automatic procedure. Considerable physical and mathematical judgment may be required to coax the generation of suitable estimates. This is a fertile area for research into heuristic algorithms to aid investigators in doing curve-fitting, but it is unlikely that parameter estimation via model-fitting with a curve-fitting process will ever be routine for all the various modeling tasks we may wish to undertake.

[1] Civilized Software Inc., URL: http://www.civilized.com

[2] Nash, John C.; Quon, Tony K., *Software for Modeling Kinetic Phenomena*, The American Statistician, Vol. 50, No. 4, pp. 368-378, 1996.

[3] Gear, C. W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs NJ, 1971.

[4] Knott, Gary D., *MLAB* Applications Manual, Civilized Software, Bethesda MD, 1996.

[5] Rosenblatt, Judah, *A More Direct Approach to Compartmental Modeling*, Progress in Food and Nutrition Science, Vol. 12, pp. 315-324, 1988.