A STUDY OF THREE ALGORITHMS FOR NONLINEAR LEAST SQUARES PARAMETER ESTIMATION

by

MICKEY LINN STILSON

B.S., Kansas State University, 1971

A MASTER'S REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Statistics

KANSAS STATE UNIVERSITY

1976

Approved by:

[Signature]
Major Professor
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Formulation of model and technique</td>
<td>4</td>
</tr>
<tr>
<td>2.1 General technique</td>
<td>4</td>
</tr>
<tr>
<td>2.1.1 Taylor series technique (Gauss-Newton procedure)</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2 Method of steepest descent</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Hartley's modified Gauss-Newton procedure</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Marquardt's procedure</td>
<td>12</td>
</tr>
<tr>
<td>2.4 Spiral algorithm</td>
<td>16</td>
</tr>
<tr>
<td>2.5 Distributional properties of parameter estimates</td>
<td>20</td>
</tr>
<tr>
<td>3. Some examples</td>
<td>22</td>
</tr>
<tr>
<td>4. Constrained parameter estimation</td>
<td>32</td>
</tr>
<tr>
<td>4.1 An example using growth curves</td>
<td>34</td>
</tr>
<tr>
<td>References</td>
<td>37</td>
</tr>
<tr>
<td>Appendix</td>
<td>38</td>
</tr>
<tr>
<td>A user's guide for NONLIN</td>
<td>38</td>
</tr>
<tr>
<td>NONLIN</td>
<td>41</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>64</td>
</tr>
</tbody>
</table>
1. Introduction.

Scientists are frequently interested in investigating the relationship between some response or dependent variable, denoted by \( y \), and a vector of independent variables, denoted by \( x \). Assume the relationship is postulated to be some function \( f \) involving a vector of parameters \( \theta \), as
\[
y = f(x, \theta).
\]
For an experimental situation where \( y \) is a random variable, the assumed relationship is
\[
E(y) = f(x, \theta)
\]
and the model with additive error structure can be expressed as
\[
y = f(x, \theta) + \epsilon.
\]
Thus over the course of \( n \) observations, we construct the model
\[
y_i = f(x_i, \theta) + \epsilon_i,
\]
where \( y, f(x, \theta) \), and \( \epsilon \) are \( n \times 1 \) vectors.

In order to more accurately classify the type of models to be analyzed, we present the following definitions ([5]):

**Definition 1.1** A model is \( y = f(x; \theta; \epsilon) \), where \( y \) is the value of an observed random variable which is to measure the phenomena under study (dependent variable), \( x \) is a vector of constants or other observed random variables (independent variables), \( \theta \) is a vector of unknown parameters, \( \epsilon \) is a vector of unobserved random variables with some assumed distribution, and \( f(\cdot,\cdot,\cdot) \) is some known mathematical function of \( x, \theta, \) and \( \epsilon \).

**Definition 1.2** A model is defined to be a **linear** model if \( y = f(x, \theta) + h(\epsilon) \), where \( f(x, \theta) \) is a linear function of the elements of \( \theta \).

Note the additive error structure implied in the above definition of a
linear model. An example is \( y = \theta_0 + \theta_1 x + \theta_2 x^2 + \epsilon \).

**Definition 1.3** A model is defined to be **essentially linear** if there exists a transformation \( \lambda(y) = \lambda(f(x, \theta, \epsilon)) \) such that \( \lambda(y) = g(x, \theta) + h(\epsilon) \), where \( g(x, \theta) \) is a linear function of \( \theta \), and \( h(\epsilon) \) is a function of \( \epsilon \) only. An example of an essentially linear model is

\[
y = (e^{\theta_0}e^{\theta_1 x})(\epsilon)
\]

as \( \ln(y) = \theta_0 + \theta_1 x + \ln(\epsilon) \). The model \( y = \exp(\theta_0 + \theta_1 x) + \epsilon \) is nonlinear because we cannot make a transformation to a linear model retaining an additive error structure. The additive error structure is important in order to apply least squares to making interval estimates about \( \theta \) and examining various distributional properties.

**Definition 1.4** Any model \( y = f(x, \theta, \epsilon) \) that is not linear or essentially linear is defined to be **nonlinear**.

Statistical theory offers many techniques for obtaining estimators of \( \theta \) from the model \( y = f(x, \theta) + \epsilon \), including maximum likelihood, Bayesian, and least squares. For an account of general methods of obtaining estimators see [10] (Chapter VII). For most techniques some objective function of \( \theta \), say \( \phi(\theta) \), is to be optimized. Examples of \( \phi \) include risk functions (Bayesian estimation), likelihood functions (maximum likelihood estimation), and sums of squares (least squares estimation). In this paper we restrict ourselves to obtaining least squares estimators for \( \theta \) from the model \( y = f(x, \theta) + \epsilon \), where \( f \) is generally a nonlinear function in \( \theta \). These estimators are also maximum likelihood estimators when normality is assumed. Since linear estimation is a special case of nonlinear estimation, all results discussed will apply equally to linear and nonlinear estimation.
In our study we will begin with a general formulation of a technique of finding a least squares estimator \( \hat{\theta} \) for \( \theta \) in \( y = f(x, \theta) + \epsilon \). The technique utilizes a Taylor series linear approximation to \( f \) and develops an iterative scheme to approach \( \hat{\theta} \). The scheme is generally referred to as the Gauss-Newton or Taylor series method. We then study three modifications to the general technique of the Gauss-Newton method. These include the modified Gauss-Newton [6], the Marquardt [8], and the Spiral [7] algorithms. These modifications are based on the premise that a procedure that converges in fewer iterations and/or with less computational effort is an improvement.

Section 3 is devoted to several examples and the problem of parameter estimation under constraints is considered in Section 4. The appendix documents a computer program developed by the author incorporating the algorithms of the modified Gauss-Newton, Marquardt, and Spiral techniques. The results of Section 4 are included in the program so that constrained estimation is possible using either the modified Gauss-Newton or Spiral algorithms. The appendix includes a user's guide to the program along with sample output.

In this study all theorems are quoted without proof and often with less than complete rigor. The reader is referred to the references for detail.
2. Formulation of model and technique.

2.1 General technique.

Consider the model
\[ y = f(x_1, x_2, \ldots, x_k; \theta_1, \theta_2, \ldots, \theta_p) + \epsilon. \]
By letting \( x = (x_1, \ldots, x_k)' \) and \( \theta = (\theta_1, \ldots, \theta_p)' \), the above model can be expressed as
\[ (2.1) \quad y = f(x, \theta) + \epsilon. \]

If there are \( n \) observations of the form \( y_i, x_{i1}, x_{i2}, \ldots, x_{ik} \) for \( i = 1, \ldots, n \), the above model can be written as
\[ y_i = f(x_{i1}, \ldots, x_{ik}; \theta_1, \ldots, \theta_p) + \epsilon_i, \]
or, following the notation of \((2.1)\),
\[ (2.2) \quad y_i = f(x_{i1}, \theta) + \epsilon_i. \]
Letting \( \epsilon = (\epsilon_1, \ldots, \epsilon_n)' \), we make the usual assumptions that \( E(\epsilon) = 0 \) and \( E(\epsilon \epsilon') = \sigma^2 I \), i.e., that the errors are identically distributed with zero means, equal variances, \( \sigma^2 \), and zero covariances. For the purpose of obtaining confidence intervals and testing hypotheses, we will later assume \( \epsilon \) is normally distributed as \( \epsilon \sim N(0, \sigma^2 I) \), but this assumption is not necessary for estimation purposes.

The least squares procedure involves determining a value of \( \theta \) which minimizes the sum of squared deviations of the observations from their expected value. The objective function can be expressed as
\[ (2.3) \quad \phi(\theta) = \sum_{i=1}^{n} (y_i - f(x_{i1}, \theta))^2. \]
Since \( y_i \) and \( x_{i1} \) are observations, only \( \theta \) is an unknown variable in \( \phi(\theta) \).
We define the least squares estimator of \( \hat{\theta} \), denoted by \( \hat{\theta} \), as that value which minimizes \( \phi(\theta) \). From [4] we note that under the assumption \( \varepsilon \sim N(0, \sigma^2 I) \), \( \hat{\theta} \) can also be shown to be the maximum likelihood estimator of \( \theta \).

To determine a suitable value for \( \hat{\theta} \), we differentiate (2.3) with respect to \( \theta \), providing the p normal equations in \( \hat{\theta} \), a solution,

\[
\sum_{i=1}^{n} \left( y_i - f(x_i, \hat{\theta}) \right) \left( \frac{\partial f(x_i, \theta)}{\partial \theta_j} \right)_{\theta = \hat{\theta}} = 0
\]

for \( j = 1, \ldots, p \), which are to be solved for \( \hat{\theta} \). This is generally not an easy task as direct solutions are not available and an iterative process must be used. Not only are the equations difficult to solve, but frequently multiple solutions exist ([4]).

2.1.1 Taylor series technique (Gauss-Newton procedure).

Suppose we have a preliminary estimate of \( \theta \), denoted by

\[
\theta^0 = (\theta_1^0, \theta_2^0, \ldots, \theta_p^0)
\]

obtained from previous experience or knowledge, or as the result of intelligent guessing. For \( \theta \) sufficiently close to \( \theta^0 \), expansion of \( f(x_i, \theta) \) in a Taylor series about \( \theta^0 \), keeping only first order terms, yields

\[
f(x_i, \theta) \approx f(x_i, \theta^0) + \sum_{j=1}^{p} \left( \frac{\partial f(x_i, \theta)}{\partial \theta_j} \right)_{\theta = \theta^0} (\theta_j - \theta_j^0).
\]

Letting \( f_i^0 = f(x_i, \theta^0), \delta_j^0 = (\theta_j - \theta_j^0), \) and \( \gamma_j^0 = \left( \frac{\partial f(x_i, \theta)}{\partial \theta_j} \right)_{\theta = \theta^0} \),

the first order approximation of the model in (2.2) becomes
(2.5) \[ y_i - f_0^i = \sum_{j=1}^{p} \delta^0_{ij} z_{ij} + \epsilon_i ; \quad i = 1, \ldots, n. \]

Writing \( Z^0 = \begin{bmatrix} z_{01} & \cdots & z_{0p} \\ \vdots & \ddots & \vdots \\ z_{n1} & \cdots & z_{np} \end{bmatrix} \),
\( y = (y_1, \ldots, y_n)' \), \( \delta^0 = (\delta^0_1, \ldots, \delta^0_n)' \), and \( f^0 = (f_1, \ldots, f_n)' \), a matrix form of (2.5) is

(2.6) \[ (y - f^0) = Z^0 \delta^0 + \epsilon. \]

which is a model linear in the unknown parameters \( \delta^0 \). The normal equations for (2.6) are

(2.7) \[ Z^0' Z^0 \delta^0 = Z^0' (y - f^0) \]

which, on assuming \( Z^0 \) is of full rank, have solutions

(2.8) \[ \hat{\delta}^0 = (Z^0' Z^0)^{-1} Z^0' (y - f^0). \]

In using (2.4), we have approximated the nonlinear function \( f \) by a plane in the region of \( \hat{\delta} \). Letting \( \delta^1 = \delta^0 + \hat{\delta}^0 \), the vector \( \hat{\delta}^0 \) can be thought of as a correction vector giving a new estimate, \( \hat{\delta}^1 \), of \( \hat{\delta} \), which is the best estimate obtainable under the linear approximation. If \( \delta(\delta^1) = 0 \), then \( \delta^1 = \hat{\delta} \) and we have a solution. In all likelihood this is not the case, so we now let \( \delta^1 \) be our new estimate, or initial value, and repeat the procedure, deriving a new correction vector and hence a new estimate \( \delta^2 \). In general

(2.9) \[ \delta^{j+1} = \delta^j + \hat{\delta}^j + (Z^j' Z^j)^{-1} Z^j' (y - f^j), \]

with obvious notation.

We continue the process until convergence is reached. Following [8] the process will be considered to have converged upon obtaining a correction vector \( \delta^j \) such that \( |\delta^j_i| \leq \epsilon \), for \( i = 1, \ldots, p \), and some small \( \epsilon > 0 \),

\[ \tau + |\delta^j_i| \]
say $10^{-5}$, and some small $\tau$, say $10^{-3}$. The presence of $\tau$ is to allow the test under the possibility $\hat{a}_i=0$ for some $i$.

For the special case where $y_i=f(x_i, \theta)+\epsilon_i$ is a linear model, $\delta^1=0$, i.e. the process converges in one step (although most programs written for the procedure will calculate $\delta^1$ anyway).

From [4], referring to the general process described above,

"The linearization procedure has possible draw-backs for some problems in that
1. It may converge very slowly; that is, a very large number of iterations may be required before the solution stabilizes even though the sum of squares...may decrease consistently as $j$ increases. This sort of behavior is not common but can occur.
2. It may oscillate widely, continuously reversing direction, and often increasing, as well as decreasing the sum of squares. Nevertheless the solution may stabilize eventually.
3. It may not converge at all, and even diverge, so that the sum of squares increases iteration after iteration without bound."

Despite these serious drawbacks, the technique, commonly referred to as the Gauss-Newton procedure, is useful, and will work successfully on a wide variety of nonlinear problems ([4]).

To circumvent the problem of wide oscillation and divergence, we discuss three proposed modifications to the general technique. But first the method of steepest descent is discussed as two of the modifications partially incorporate it into their modifications. As in the above discussion, the development of the method of steepest descent is from [4]. A more general and theoretical development is found in [1].

2.1.2 Method of steepest descent.

From some initial guess, $\theta^0$, it is desired to seek an iterative value $\theta^{j+1}$, such that $\phi(\theta^{j+1}) < \phi(\theta^j)$ in a sequential manner that leads to con-
vergence. From elementary calculus we know that the direction in which 
\( \phi(\theta) \) decreases lies along the vector

\[
\hat{\delta}_* = \left[ \begin{align*}
- \frac{\partial \phi(\theta)}{\partial \theta_1} , & \quad - \frac{\partial \phi(\theta)}{\partial \theta_2} , & \quad \cdots , & \quad - \frac{\partial \phi(\theta)}{\partial \theta_p}
\end{align*} \right]_{\theta = \theta^j} .
\]

Thus if \( \hat{\delta}^j = \rho \hat{\delta}_* \) for \( 0 < \rho \leq 1 \), then for some \( \rho \), \( \phi(\hat{\theta} + \hat{\delta}^j) \leq \phi(\theta^j) \). Hence at the \( j \)th step, \( \hat{\theta}_* \) is searched by varying \( \rho \) until we find a point at which \( \phi(\hat{\theta} + \hat{\delta}_*) \leq \phi(\theta^j) \).

By this process we avoid the possibility of divergence or wild oscillation mentioned in the discussion of the Taylor series technique. But often the method of steepest descent converges very slowly, so slowly, in fact, as to make the method unadvisable as a sole technique. The problem occurs for models whose sums of squares surface consists of a long, narrow trough in the vicinity of the minimum. An example of this is the model

\[
f(x_1) = \theta_1 e^{\theta_2 x_1} + \varepsilon_1 \quad \text{for } x_1 > 0,
\]

since \( \frac{\partial f}{\partial \theta_1} = 1 \) and \( \frac{\partial f}{\partial \theta_2} = x_1 e^{\theta_2 x_1} \) which can be very large. Thus a small change in \( \theta_2 \) will produce a much larger change in the sums of squares function than will a similar change in \( \theta_1 \).

To illustrate the differences between the Taylor series and steepest descent vectors produced, consider the function

\[
y_1 = 5 + e^{-0.5x_1} \quad \text{where } x_1 = 1, x_2 = 2, \text{ and } x_3 = 3.
\]

Figure 2.1 graphs \( \phi(\theta) \) for various values of \( \theta \) in the vicinity of the true values \( \theta_1 = 5, \theta_2 = 0.5 \). Suppose, not knowing \( \theta \), we supply as initial
ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE
<table>
<thead>
<tr>
<th>(0_2)</th>
<th>(0_1)</th>
<th>4.0</th>
<th>4.2</th>
<th>4.4</th>
<th>4.6</th>
<th>4.8</th>
<th>5.0</th>
<th>5.2</th>
<th>5.4</th>
<th>5.6</th>
<th>5.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1</td>
<td>2.569E1</td>
<td>2.294E1</td>
<td>1.963E1</td>
<td>1.699E1</td>
<td>1.453E1</td>
<td>1.234E1</td>
<td>1.040E1</td>
<td>8.687E0</td>
<td>7.218E0</td>
<td>5.989E0</td>
<td></td>
</tr>
<tr>
<td>.2</td>
<td>2.039E1</td>
<td>1.758E1</td>
<td>1.502E1</td>
<td>1.289E1</td>
<td>1.061E1</td>
<td>8.709E0</td>
<td>7.155E0</td>
<td>5.769E0</td>
<td>4.664E0</td>
<td>3.779E0</td>
<td></td>
</tr>
<tr>
<td>.3</td>
<td>1.442E1</td>
<td>1.205E1</td>
<td>9.922E0</td>
<td>8.035E0</td>
<td>6.388E0</td>
<td>4.91E0</td>
<td>3.814E0</td>
<td>2.888E0</td>
<td>2.201E0</td>
<td>1.754E0</td>
<td></td>
</tr>
<tr>
<td>.4</td>
<td>9.239E0</td>
<td>6.435E0</td>
<td>4.870E0</td>
<td>3.546E0</td>
<td>2.461E0</td>
<td>1.617E0</td>
<td>1.012E0</td>
<td>6.477E1</td>
<td>5.232E1</td>
<td>6.387E1</td>
<td></td>
</tr>
<tr>
<td>.5</td>
<td>3.009E0</td>
<td>1.320E0</td>
<td>1.080E0</td>
<td>4.800E-1</td>
<td>1.200E-1</td>
<td>4.930E-32</td>
<td>1.200E-1</td>
<td>4.800E-32</td>
<td>1.200E-1</td>
<td>1.200E0</td>
<td></td>
</tr>
<tr>
<td>.6</td>
<td>1.164E0</td>
<td>0.322E0</td>
<td>1.119E0</td>
<td>1.456E0</td>
<td>2.032E0</td>
<td>2.651E0</td>
<td>3.308E0</td>
<td>5.205E0</td>
<td>6.743E0</td>
<td>8.520E0</td>
<td></td>
</tr>
<tr>
<td>.7</td>
<td>7.723E0</td>
<td>8.798E0</td>
<td>1.011E1</td>
<td>1.167E1</td>
<td>1.346E1</td>
<td>1.550E1</td>
<td>1.777E1</td>
<td>2.029E1</td>
<td>2.304E1</td>
<td>2.603E1</td>
<td></td>
</tr>
<tr>
<td>.8</td>
<td>3.291E1</td>
<td>3.067E1</td>
<td>3.797E1</td>
<td>4.119E1</td>
<td>4.450E1</td>
<td>4.812E1</td>
<td>5.193E1</td>
<td>5.608E1</td>
<td>6.042E1</td>
<td>6.500E1</td>
<td></td>
</tr>
<tr>
<td>.9</td>
<td>9.379E1</td>
<td>9.853E1</td>
<td>1.035E2</td>
<td>1.087E2</td>
<td>1.142E2</td>
<td>1.199E2</td>
<td>1.258E2</td>
<td>1.320E2</td>
<td>1.384E2</td>
<td>1.451E2</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>2.298E2</td>
<td>2.342E2</td>
<td>2.419E2</td>
<td>2.498E2</td>
<td>2.580E2</td>
<td>2.664E2</td>
<td>2.751E2</td>
<td>2.840E2</td>
<td>2.931E2</td>
<td>3.025E2</td>
<td></td>
</tr>
</tbody>
</table>

Sums of Squares Surface for Model (2.12)

Figure 2.1
guesses $\theta_1=4.2$, $\theta_2=.8$. As can be seen in Figure 2.1, $\phi(\theta^0)=35.07$.
As presented in section 2.1.1, we form the matrix of partials,

$$
Z = \begin{bmatrix}
\frac{\partial f(x_1, \theta)}{\partial \theta_1} & \frac{\partial f(x_1, \theta)}{\partial \theta_2} \\
\frac{\partial f(x_2, \theta)}{\partial \theta_1} & \frac{\partial f(x_2, \theta)}{\partial \theta_2} \\
\frac{\partial f(x_3, \theta)}{\partial \theta_1} & \frac{\partial f(x_3, \theta)}{\partial \theta_2}
\end{bmatrix}
$$

or

$$
Z = \begin{bmatrix}
1 & x_1 e^{\theta_2 x_1} \\
1 & x_2 e^{\theta_2 x_2} \\
1 & x_3 e^{\theta_2 x_3}
\end{bmatrix},
$$

and the vector $f^0 = \begin{bmatrix} f(x_1, \theta^0) \\
f(x_2, \theta^0) \\
f(x_3, \theta^0)
\end{bmatrix}$.

From the given data, we compute $\chi = \begin{bmatrix} 6.6487 \\
7.7183 \\
9.4817 \end{bmatrix}$.

Thus $\theta^1 = \theta^0 + (Z'Z)^{-1}Z'(\chi - f^0) = (4.77, 0.608)'$. $\phi(\theta^1)=2.407$, and so by the Taylor series technique we have found a new point $\theta^1$ such that $\phi(\theta^1)<\phi(\theta^0)$, and hence we have a successful iteration.

Following the method of steepest descent,

$\delta^* = (-6.95, -203.47)'$.

This correction is too severe to allow graphing in Figure 2.1, so we use $\rho=.001$, and compute $\delta^1 = \rho \delta^* = (-.00695, -.20347)'$. Thus $\theta^1 = \theta^0 + \delta^1 = (4.19, .5965)'$, and $\phi(\theta^1)=0.9496$.

Our two updates serve to illustrate fairly universal properties of the two techniques: the update $\theta^1$ by steepest descent produces a
greater reduction in the objective function, but the $\delta^1$ of the Taylor series method is "closer" to $\theta$ in terms of units of $\theta_1$ and $\theta_2$. One should note how the steepest descent corrections will begin a hemstitch pattern across the sums of squares through depicted in Figure 2.1 as it continually moves across the valley in the direction of steepest slope.

When $\theta^0$, the initial guess, is "far enough" away from $\theta$, steepest descent updates are more satisfactory than corrections produced by the Taylor series method, but as $\theta$ approaches $\theta$, Newtonian steps are more satisfactory, as illustrated above.

2.2 Hartley's modified Gauss-Newton procedure.

Intuition tells us that, given "enough" time (i.e., iterations), the method of steepest descent will ultimately lead us to a minimum in $\phi(\theta)$, albeit a spurious, local minimum is always a possibility. However, this is not the case with the Gauss-Newton procedure. If the linear approximation of $\phi(\theta)$ in the vicinity of $\theta^1$ is a particularly poor one, the method can easily lead to divergence, due mainly to an inappropriate step size rather than an inappropriate direction. An obvious "solution" exists; go some portion of the distance of the Gauss-Newton correction vector, as, for some $\delta^j$ correction obtained by the Gauss-Newton method, there exists $\rho$, $0 < \rho < 1$, such that $\phi(\theta^1 + \delta^j) < \phi(\theta^1)$. The proof of existence and subsequent convergence of the procedure is given in [6]. Hartley shows that if $\theta^0$ is within a bounded convex set $\mathcal{S}$ of the parameter space spanned by $\theta$, and if, for

$$0 = \lim_{\mathcal{S}} \inf_{x, \theta} \phi(x, \theta),$$
where $\bar{S}$ is the complement of $S$, there exists a $\beta^*$ in the interior of $S$

such that

$$\phi(x, \beta^*) < \eta,$$

then the process as modified above will converge (to at least a local
minimum, if $\beta^0$ is not properly chosen).

In application Hartley suggests that $\phi(x^0 + \rho^j)$ be evaluated at
$\rho = 0$, $\rho = \frac{1}{2}$, and $\rho = 1$. The three values describe a parabola in $\rho$, from
which a minimum is obtained from

$$\rho_{\text{min}} = \frac{1}{2} + \frac{\phi(\rho = 0) - \phi(\rho = 1)}{(\phi(\rho = 1) - 2\phi(\rho = \frac{1}{2}) + \phi(\rho = 0))}$$

If

$$\phi(x^0 + \rho_{\text{min}} \beta^j) > \phi(x^0),$$

then the computations are repeated using $\rho_{\text{min}}^j$, and continued until an
appropriate $\rho_{\text{min}}$ is found.

2.3 Marquardt's procedure.

As previously discussed, Newtonian steps are to be preferred to those
in the direction of steepest descent. Marquardt and others ([8]) have ex-
amined a number of nonlinear problems and found that typically the cor-
rection vectors produced by a Gauss-Newton method and the direction of
steepest descent are 80-90° apart. This is seen also in our earlier ex-
ample (see Figure 2.1). But a small enough step in the direction of
steepest descent will always produce a reduction in the objective func-
tion, $\phi(x)$, whereas no suitable correction may be found along the vec-
tor given by the Taylor series approximation (at least not along the
part searched by most computer programs). Hence it seems desirable that
as we reach a point in our search for \( \hat{\sigma} \) where a Newtonian step does not produce a suitable step, we would use corrections given by steepest descent until Newtonian steps can be resumed. Marquardt ([8]) suggests an algorithm that interpolates between the correction vectors of the Gauss-Newton procedure and steepest descent. The theoretical basis for the algorithm is as follows: At the \( j \)th iteration, let
\[
A = (Z^j)'Z^j \quad \text{and} \quad \nu = (y - f^j).
\]
Then the Gauss-Newton correction vector, which we will now denote by \( \delta_t \), is given by the solution of
\[
(2.3.1) \quad A \delta_t = (Z^j)' \nu
\]
and the direction of steepest descent lies along the vector \( \delta_g \), where
\[
(2.3.2) \quad \delta_g = (Z^j)' \nu.
\]
The following three theorems are due to Marquardt (see [8] for proofs).

**Theorem 1.** Let \( \lambda \neq 0 \) be arbitrary and let \( \delta \) satisfy the equation
\[
(2.3.3) \quad (A + \lambda I) \delta = (Z^j)' \nu,
\]
where \( I \) is the identity matrix of size \( p \times p \), \( p \) being the number of parameters to be estimated. Then \( \delta \) minimizes \( \phi(\theta) \) on the sphere whose radius \( \| \delta_r \| \) satisfies \( \| \delta_r \|^2 = \| \delta \|^2 \).

**Theorem 2.** Let \( \delta(\lambda) \) be the solution of (2.3.3) for a given \( \lambda \). Then \( \| \delta(\lambda) \|^2 \) is a continuous decreasing function of \( \lambda \), such that as \( \lambda \to \infty \),
\[
\| \delta(\lambda) \|^2 \to 0.
\]

**Theorem 3.** Let \( \gamma \) be the angle between \( \delta \) and \( \delta_g \). Then \( \gamma \) is a continuous monotone decreasing function of \( \lambda \) such that as \( \lambda \to \infty \), \( \gamma \to 0 \). Since \( \delta_g \) is independent of \( \lambda \), it follows that \( \delta \) rotates toward \( \delta_g \) as \( \lambda \to \infty \).

Examining (2.3.3) we see that for \( \lambda = 0 \) (2.3.3) is equivalent to (2.3.1), that is, \( \delta = \delta_t \), and we take a Newtonian step. As \( \lambda \) becomes large (2.3.3)
can be approximated by (2.3.2) (that is, \( \lambda \) dominates the maximum characteristic root of \( A \)), so that for large \( \lambda \) we move approximately in the direction of steepest descent. For \( \lambda = 0 \) the step size is that produced by the Gauss-Newton procedure. Theorem 2 tells us that as \( \lambda \) increases, the step size decreases asymptotically to zero, so that we will always obtain a suitable step \( \delta_j^* \) such that

\[
\phi(\theta^j + \delta_j^*) \leq \phi(\theta^j)
\]

for an appropriate choice of \( \lambda \), and hence convergence is guaranteed (even if only to a local minimum).

One additional step is necessary before we have a viable procedure. Adding \( \lambda \) to each diagonal element of \( A \) (as in (2.3.3)) will not be useful if the elements are of widely varying orders of magnitude. To circumvent the problem of needing to add a different \( \lambda \) to each diagonal element, Marquardt suggests rescaling the \( A \) matrix in terms of the standard deviations of the first partials. This amounts to a rescaling of the parameter space. As an added effect, we succeed in stabilizing \( A \) for the purpose of inversion. Letting \( g = Z^T \lambda \), define \( A^* \) and \( g^* \) by

\[
A^* = (a_{ij}^*) = \left( \frac{a_{ij}}{\sqrt{a_{ii}a_{jj}}} \right) \quad \text{and} \quad g^* = (g_j^*) = \left( \frac{g_j}{\sqrt{a_{jj}}} \right).
\]

Solving for the Gauss-Newton correction \( \delta_t^* \), we use

\[
A^* \delta_t^* = g^*.
\]

and

\[
\delta_t = (\delta_j) = \left( \frac{\delta_j}{\sqrt{a_{jj}}} \right).
\]

The algorithm is now clear. At the \( j^{th} \) step (iteration) construct
(2.3.4) \[ (\Delta^* j + \lambda^j) \hat{\delta}^j = a^j \]

and solve for \[ \hat{\delta}^j = (\hat{\delta}^j_i) = \frac{\delta^j_i}{\sqrt{\delta^j_i}} \).

As before, \[ \hat{e}^j = \hat{e}^j + \delta^j \]

if we have selected \( \lambda^j \) so that

(2.3.5) \[ \phi(\hat{\theta}^{j+1}) \leq \phi(\hat{\theta}^j) \].

It is clear from the theoerems that a (large) \( \lambda \) always exists such that (2.3.5) is satisfied, unless \( \hat{e}^j \) is already at \( \hat{e} \), the minimum (global or local, as the case may be) of \( \phi(\hat{e}) \). But constant use of a large \( \lambda \) will produce results similar to the steepest descent technique. Thus, when the sums of squares surface, \( \phi(\hat{e}) \), is reasonably well approximated by linearization, we wish to use a small \( \lambda \) so that we take Newtonian steps, and use a large \( \lambda \) for gradient steps only as necessary for convergence. Accordingly, Marquardt gives the following strategy:

Let \( \nu > 1 \).

Let \( \lambda^{j-1} \) denote the value of \( \lambda \) from the previous iteration (where \( \lambda^0 = .01 \), say, initially). Then the trial values for \( \lambda^j \) are \( \lambda^{j-1}/\nu \) and \( \lambda^{j-1} \).

(i) Compute \( \phi(\hat{\theta}^{j+1}, \lambda^{j-1}/\nu) \) and \( \phi(\hat{\theta}^{j+1}, \lambda^{j-1}) \). If \( \phi(\hat{\theta}^{j+1}, \lambda^{j-1}/\nu) \leq \phi(\hat{\theta}^j) \), then \( \lambda^{j} = \lambda^{j-1}/\nu \).

(ii) If \( \phi(\hat{\theta}^{j+1}, \lambda^{j-1}/\nu) > \phi(\hat{\theta}^j) \) and \( \phi(\hat{\theta}^{j+1}, \lambda^{j-1}) \leq \phi(\hat{\theta}^j) \), then \( \lambda^{j} = \lambda^{j-1} \).

(iii) Otherwise, increase \( \lambda \) successively by multiplying by \( \nu \) until for the smallest integer \( \omega \), \( \phi(\hat{\theta}^{j+1}, \lambda^{j-1}/\nu^\omega) \leq \phi(\hat{\theta}^j) \); then use \( \lambda^{j} = \lambda^{j-1}/\nu^\omega \).

From [8]:

Typically, condition (iii) is met only rarely. Thus it is most often required that (2.3.4) be solved for two values of
\( \lambda^j \) at each iteration. One such solution is required for the standard Taylor series method. The extra linear equation solution is generally much less computational effort than the evaluation of the \( A^* \) matrix, so that the small proportional increase in computation per iteration is more than offset by the gain in the power of an iteration.

To which we add: if the number of parameters is fairly "small" in relation to the number of observations.

2.4 Spiral algorithm.

Jones, [7], agrees that Newtonian steps are preferable and that steepest descent corrections should only be used as necessary, but he sees a major disadvantage in Marquardt's procedure in that a matrix inversion is required to generate each search point. Thus, he proposes an algorithm which searches roughly the same area as Marquardt's (this deserves further comment later in the development), but which generates search points between Newtonian and steepest descent corrections by vector addition rather than by matrix inversion. Though not specifically mentioned as a goal, we will note that his proposal also returns to a true Newtonian step immediately, rather than waiting a few iterations until the operation \( \lambda^j / \nu \) reduces \( \lambda \) sufficiently. This feature probably accounts for much of the improved performance Jones claims over Marquardt's procedure (on problems for which Newtonian steps are successful for a large majority of iterations).

The essence of the Spiral algorithm Jones proposes is as follows:

In agreement with Marquardt's procedure, \( \theta^{j+1} \) always exists in the plane formed by the Taylor series correction vector and the line of steepest descent such that
\( \phi(\theta^{j+1}) \leq \phi(\theta^j) \).

Figure 2.2, drawn in this defined plane, shows the point 0, which is \( \theta^j \); the vector \( \overrightarrow{OT} \), which is the Taylor series correction vector, and the vector \( \overrightarrow{OS} \), where D is chosen along the path of steepest descent such that \( ||\overrightarrow{OT}|| = ||\overrightarrow{OS}|| \).

![Figure 2.2](image)

As we wish to progress as much as possible at each iteration and keep computations at a minimum, strategy demands that \( \theta^{j+1} \) be as far from \( \theta^j \) as possible and the number of evaluations of \( \phi(\theta) \) (and hence \( f(x, \theta) \)) be kept to a minimum. Hence \( \theta^{j+1} \) will be chosen as the first point such that

\[ \phi(\theta^{j+1}) \leq \phi(\theta^j). \]

It follows that the point T in Figure 2.2 is the first to be investigated. If this does not produce a reduction in \( \phi(\theta) \), then a linear approximation of \( \phi(\theta) \) does not extend well to T from 0. Hence \( \phi(\theta) \) has some minimum "trough" that curves in one of the two directions shown by the hatched curves in Figure 2.2. At this point Jones claims "Since the overall strategy tends to give base points [point 0, that is, \( \theta^j \)] on the outside shoulder of the valley, it is reasonable to assume that the valley is
moving away from the line $OT$. Thus he suggests the search be conducted next along the spiral $OTS$, as to be described shortly, in order to try to intercept the "valley" (trough). The example discussed in section 2.1.2 can be shown to be an exception to this idea. Hence we suggest that the line $OT$ be searched as in the modified Gauss-Newton procedure of section 2.2 before attempting a search elsewhere. This strategy is incorporated in the computer program described in the appendix.

When no suitable correction is found at $T$ (or along $OT$), we must search the area between $OT$ and $OD$, as in Marquardt's procedure. As with Marquardt's procedure, we need to approach $OD$ in such a fashion that the step size approaches zero, so that convergence can again be guaranteed.

Jones suggests a search along the spiral $OTS$ and has found the most suitable spiral to be (expressed in polar coordinates, see Figure 2.2) given by

$$r=r_0(1-\phi \cos \theta - (1-\gamma \cos \phi) (\phi / \gamma)^2)$$

where $r$ is the distance $OS$, and $r_0$ is the distance $OT$. The sequence of points $S$ to be investigated are defined from a sequence of points $L$ generated on $TD$ in the ratio $\mu : (1-\mu)$ (see Figure 2.2). Jones suggests that successive values of $\mu$ be computed from

$$\mu_{n+1} = 2\mu_n / (1+\mu_n)$$

so that the points $\{L_n\}$ become closer together as they approach $D$ on $TD$.

The coordinates of $L_n, (\xi, \phi)$, can be derived as

$$\tan \phi = \frac{\mu \sin \gamma}{1-\mu + \mu \cos \gamma}$$

and

$$\xi = r_0 \frac{\mu \sin \gamma}{\sin \phi}$$
If 0 is the origin, the coordinates $s$ of $S$ are given by the coordinates $t$ of $T$ and $d$ of $D$ by

$$s = r \left( \frac{1}{\xi} (\mu d + (1-\mu) t) \right).$$

As shown, each of the successive search points $S$ is generated by a weighted sum of two vectors, as opposed to the matrix inversion required in Marquardt's procedure. As previously suggested, we stop as soon as we find a point $S$ such that

$$\phi(e^j+s) \leq \phi(e^j).$$

One should see that since the spiral is searched at only a discrete set of points, it is possible to jump from one side of the trough to the other. Jones suggests that an "interpolation be performed whenever three consecutive sums of squares along the spiral are convex downwards, the sum of squares being regarded as a function of $\mu$." Work with this algorithm has led us to suggest two refinements, in addition to the one already mentioned. First, it is a distinct possibility that the trough of the surface $\phi(e)$ curves through OT in Figure 2.2, contrary to Jones' statement. If the angle $\gamma$ is fairly large, the search along the spiral OTS will often lead to points $\{S_n\}$ far away from the trough. Machine limitations on magnitudes of numbers enter into play, as $\phi(e^j+s)$ can become very large, overflowing many machines. Hence the computer program in the appendix suspends any overflow messages produced by the IBM 370 FORTRAN IV compiler, and sets $\phi(e^j+s)$ to a predetermined constant ($10^{74}$). Secondly, if for two consecutive search points $s_n$ and $s_{n+1}$ we have

$$\phi(e^j+s_{n+1}) \leq \phi(e^j+s_n) \leq \phi(e^j),$$
then the angle $\beta$ (see Figure 2.2) is cut in half and a new spiral is searched that now lies closer to the line $OT$ ($\beta$ is initially set to half of $\gamma$, as suggested by Jones). These two suggestions are not theoretical faults of the algorithm, but are necessary for implementation on computers with finite computational capabilities.

2.5 Distributional properties of parameter estimates.

For the model

$$y_i = f(x_i, \theta) + \epsilon_i,$$

we assume that the $\epsilon_i$'s are independent, identically, normally distributed with mean zero and variances $\sigma^2$, i.e.,

$$\epsilon \sim N(0, \sigma^2 I).$$

Hence

$$E(y_i) = f(x_i, \theta).$$

The density function for an observation $y_i$ is given by

$$g_i(y_i, \theta) = (2\pi \sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2} (y_i - f(x_i, \theta))^2\right).$$

Thus, the likelihood function for $\theta$, $LH(\theta)$, is (for a sample of size $n$)

$$LH(\theta) = (2\pi \sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2\right),$$

and the log of the likelihood function, written $L(\theta)$, is

$$L(\theta) = -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2$$

$$= -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \phi(\theta).$$

It is now apparent that to maximize the likelihood function by suit-
able choice of $\hat{\theta}$, we can minimize $\phi(\theta)$ by choice of $\theta$. Thus the least squares estimate of $\theta$ is also the maximum likelihood estimate for $\theta$.

Since $\hat{\theta}$ is a maximum likelihood estimate for $\theta$, a central limit theorem gives us that asymptotically,

$$\hat{\theta} \sim \text{MVN}(\theta, V),$$

where

$$V = \sigma^2 (Z'Z)^{-1}.$$

(Note that $(Z'Z)^{-1}$ is evaluated at $\theta$, the true, but unknown, parameter value.) It can be shown that $\hat{\theta}$ is a consistent estimator of $\theta$

and thus $Z'Z\mid_{\theta=\hat{\theta}}$ is a consistent estimator of $Z'Z\mid_{\theta}$. It follows that

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2$$

is a consistent estimator of $\sigma^2$, and that

$$\hat{V} = \hat{\sigma}^2 (Z'Z\mid_{\theta=\hat{\theta}})^{-1}$$

is a consistent estimator of $V$. A $(1-\alpha)100\%$ asymptotic approximate confidence interval about $a'\theta$ for a given vector $a$ is

$$a'\hat{\theta} \pm t_{\alpha/2(n-p)} \sqrt{a'\hat{V}a}.$$

The computer program in the appendix provides the residual sums of squares, from which $\hat{\sigma}^2$ can be obtained, and $(Z'Z)^{-1}$ evaluated at the last iterative estimate for $\hat{\theta}$, so that confidence intervals involving $\hat{\theta}$ can be formed.
3. Some examples.

Four examples of application of the three algorithms developed in the previous section are now presented. The algorithms are incorporated in a computer program, NONLIN, written by the author and documented in the appendix along with a user's guide. It is important to notice that none of the algorithms are completely successful in that performance is related to starting values and the type of nonlinear function.

Initial parameter estimates provide the most critical problem; all of the theory for the procedures is dependent on starting the iterative process within the sphere of convergence for the least squares solution. Hence a successful estimator is often not obtained until after much searching for satisfactory starting values, if at all. Reasonable values can often be selected from a knowledge of the process, previous work, or, in the case of a few parameters and a relatively simple function, by using a few data points and solving for the unknown parameters.

The first example comes from animal science and involves estimating the parameters for a simple exponential growth curve, modeling weight of cows versus time. The model is

\[(3.1) \quad y_i = \theta_1 - \theta_2 \exp(-\theta_3 x_i) + \epsilon_i\]

where \(y_i\) is the weight of the animal at time \(x_i\) (time measured to the nearest month after birth). Data for a particular animal is given in Table 3.1. After 18 months a cyclical pattern dominates the data due to yearly calving. The model makes no attempt to account for this, fitting instead, a curve through something akin to an average weight through the cycle.
Table 3.1
Data for Growth Curve Model, Eq. (3.1)

<table>
<thead>
<tr>
<th>Month*</th>
<th>Weight (lbs)</th>
<th>Month*</th>
<th>Weight (lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>64</td>
<td>36</td>
<td>640</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>37</td>
<td>630</td>
</tr>
<tr>
<td>3</td>
<td>130</td>
<td>38</td>
<td>540</td>
</tr>
<tr>
<td>4</td>
<td>160</td>
<td>39</td>
<td>610</td>
</tr>
<tr>
<td>5</td>
<td>205</td>
<td>40</td>
<td>670</td>
</tr>
<tr>
<td>6</td>
<td>270</td>
<td>41</td>
<td>720</td>
</tr>
<tr>
<td>7</td>
<td>305</td>
<td>42</td>
<td>745</td>
</tr>
<tr>
<td>8</td>
<td>310</td>
<td>42</td>
<td>775</td>
</tr>
<tr>
<td>9</td>
<td>310</td>
<td>44</td>
<td>775</td>
</tr>
<tr>
<td>10</td>
<td>346</td>
<td>44</td>
<td>780</td>
</tr>
<tr>
<td>11</td>
<td>315</td>
<td>46</td>
<td>745</td>
</tr>
<tr>
<td>12</td>
<td>375</td>
<td>47</td>
<td>680</td>
</tr>
<tr>
<td>13</td>
<td>380</td>
<td>48</td>
<td>690</td>
</tr>
<tr>
<td>14</td>
<td>415</td>
<td>48</td>
<td>665</td>
</tr>
<tr>
<td>15</td>
<td>450</td>
<td>50</td>
<td>645</td>
</tr>
<tr>
<td>16</td>
<td>535</td>
<td>51</td>
<td>635</td>
</tr>
<tr>
<td>17</td>
<td>550</td>
<td>52</td>
<td>690</td>
</tr>
<tr>
<td>19</td>
<td>540</td>
<td>53</td>
<td>690</td>
</tr>
<tr>
<td>20</td>
<td>660</td>
<td>54</td>
<td>750</td>
</tr>
<tr>
<td>21</td>
<td>730</td>
<td>55</td>
<td>770</td>
</tr>
<tr>
<td>22</td>
<td>700</td>
<td>56</td>
<td>820</td>
</tr>
<tr>
<td>23</td>
<td>650</td>
<td>56</td>
<td>825</td>
</tr>
<tr>
<td>24</td>
<td>670</td>
<td>58</td>
<td>780</td>
</tr>
<tr>
<td>25</td>
<td>610</td>
<td>59</td>
<td>855</td>
</tr>
<tr>
<td>26</td>
<td>470</td>
<td>60</td>
<td>830</td>
</tr>
<tr>
<td>27</td>
<td>610</td>
<td>61</td>
<td>800</td>
</tr>
<tr>
<td>28</td>
<td>615</td>
<td>62</td>
<td>640</td>
</tr>
<tr>
<td>29</td>
<td>675</td>
<td>63</td>
<td>770</td>
</tr>
<tr>
<td>29</td>
<td>700</td>
<td>64</td>
<td>810</td>
</tr>
<tr>
<td>31</td>
<td>715</td>
<td>65</td>
<td>875</td>
</tr>
<tr>
<td>32</td>
<td>710</td>
<td>66</td>
<td>905</td>
</tr>
<tr>
<td>33</td>
<td>690</td>
<td>66</td>
<td>935</td>
</tr>
<tr>
<td>34</td>
<td>650</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>670</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Month after birth.
Starting values for this example present no particular problem. First, the function is sufficiently well-behaved that any reasonable guess will lead to convergence for all three algorithms. Secondly, note that as $x$ (time) approaches infinity, $y$ approaches $\theta_1$, so that an initial guess for $\theta_1$ should be the maximum weight suspected for the animal. At time zero (birth), $y=\theta_1-\theta_2$, so $\theta_2$ should be estimated by subtracting birth weight from the guess for $\theta_1$. A little forethought leads to selecting $\theta_3$ in the range from .01 to .1.

For the data in Table 3.1, the initial guess used was:

$\theta_0=900, \quad \theta_2=836, \quad \theta_3=.05$

The results for the three algorithms are summarized below:

<table>
<thead>
<tr>
<th></th>
<th>Mod. Gauss-Newton</th>
<th>Marquardt</th>
<th>Spiral</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of iterations</td>
<td>9</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>SS Residual</td>
<td>307,763.8970870435</td>
<td>307,763.8969855355</td>
<td>307,763.8969043224</td>
</tr>
<tr>
<td>$\hat{\theta}_1$</td>
<td>800.11853</td>
<td>800.11771</td>
<td>800.12045</td>
</tr>
<tr>
<td>$\hat{\theta}_2$</td>
<td>768.57996</td>
<td>768.57663</td>
<td>768.57561</td>
</tr>
<tr>
<td>$\hat{\theta}_3$</td>
<td>.05594</td>
<td>.05594</td>
<td>.05594</td>
</tr>
</tbody>
</table>

There is not enough disparity among the three concerning the final residual sums of squares (SS Residual) or final parameter estimates, hence for this model the Marquardt procedure is preferred, having taken the fewest iterations. Note that $\hat{\theta}_1-\hat{\theta}_2$ does not estimate birth weight very well. The animal scientist may, then, reject these estimates and, thus, the model as unrealistic. It is possible to conduct the search under the constraint $\theta_2-\theta_3 = \text{birth weight}$. A procedure is discussed in Section 4.

Next we look at a textbook example which was selected so that computer results from the program NONLIN could be verified with published work. The model is $$1$$
(3.2) \[ y_i = \exp(-\theta_1 x_1) \exp(-\theta_2/x_2) + \epsilon_i \]

(a model of the fraction remaining at time \( x_1 \) of a chemical compound undergoing a first order reaction, where \( x_2 \) is the absolute temperature of the system. The equation is obtained as a solution to the differential equation \( dy/dx = -ky \).)

Data for the model is given in Table 3.2. Following [1], the initial guess is \( \theta_1^0 = 750, \theta_2^0 = 1200 \). Results for the three algorithms were:

<table>
<thead>
<tr>
<th>Method</th>
<th>No. of iterations</th>
<th>SS Residual</th>
<th>( \hat{\theta}_1 )</th>
<th>( \hat{\theta}_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mod. Gauss-Newton</td>
<td>9</td>
<td>0.039806054412401</td>
<td>813.87105</td>
<td>961.00245</td>
</tr>
<tr>
<td>Marquardt</td>
<td>10</td>
<td>0.039806054421774</td>
<td>813.85866</td>
<td>960.99876</td>
</tr>
<tr>
<td>Spiral</td>
<td>11</td>
<td>0.039806054415955</td>
<td>813.86754</td>
<td>961.00090</td>
</tr>
</tbody>
</table>

These results are in close agreement with [1]. It appears the objective function has a long narrow trough in the vicinity of \( \hat{\theta} \), so that Newtonian steps lead to quickest convergence, and any step toward either side of that direction is not as efficient.

The next example utilizes a more complicated function that is an intermediate result in work on segmenting two sigmoidal growth curves. More detail is found in Section 4, where this topic is discussed as an example of constrained estimation. The equation is

(3.3) \[ y_i = \begin{cases} \theta_1 (1-\theta_3 \exp(-\theta_2 x_1^2)) + \epsilon_i, & x_i \leq \gamma \\ \theta_1 (1-\theta_3 \exp(-\theta_2 \gamma^2) + \theta_2 \theta_3 \exp(-\gamma^2 \theta_2) (1-\exp(-\theta_4 (x_i - \gamma^2))) + \epsilon_i, & x_i > \gamma \end{cases} \]

Rather than trying to fit this model to real data (which was done successfully after this example), we generate, with zero error, data using the values \( \theta_1 = .2, \theta_2 = .004, \theta_3 = .4, \theta_4 = .009, \gamma = 11.18 \).
Table 3.2

Data for Example from Bard, Eq. 3.2

<table>
<thead>
<tr>
<th>Experiment Number, $i$</th>
<th>Time $x_{1i}$ (hr)</th>
<th>Temperature $x_{2i}$ (°K)</th>
<th>Fraction remaining, $y_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>100</td>
<td>0.980</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>100</td>
<td>0.983</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>100</td>
<td>0.955</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>100</td>
<td>0.979</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>100</td>
<td>0.993</td>
</tr>
<tr>
<td>6</td>
<td>0.05</td>
<td>200</td>
<td>0.626</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>200</td>
<td>0.544</td>
</tr>
<tr>
<td>8</td>
<td>0.15</td>
<td>200</td>
<td>0.455</td>
</tr>
<tr>
<td>9</td>
<td>0.2</td>
<td>200</td>
<td>0.225</td>
</tr>
<tr>
<td>10</td>
<td>0.25</td>
<td>200</td>
<td>0.167</td>
</tr>
<tr>
<td>11</td>
<td>0.02</td>
<td>300</td>
<td>0.566</td>
</tr>
<tr>
<td>12</td>
<td>0.04</td>
<td>300</td>
<td>0.317</td>
</tr>
<tr>
<td>13</td>
<td>0.06</td>
<td>300</td>
<td>0.034</td>
</tr>
<tr>
<td>14</td>
<td>0.08</td>
<td>300</td>
<td>0.016</td>
</tr>
<tr>
<td>15</td>
<td>0.1</td>
<td>300</td>
<td>0.066</td>
</tr>
</tbody>
</table>
The generated data is

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X</th>
<th>Y</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.12032</td>
<td>8</td>
<td>0.13807</td>
<td>15</td>
<td>0.16428</td>
</tr>
<tr>
<td>2</td>
<td>0.12127</td>
<td>9</td>
<td>0.14214</td>
<td>16</td>
<td>0.16641</td>
</tr>
<tr>
<td>3</td>
<td>0.12283</td>
<td>10</td>
<td>0.14367</td>
<td>17</td>
<td>0.16811</td>
</tr>
<tr>
<td>4</td>
<td>0.12496</td>
<td>11</td>
<td>0.15069</td>
<td>18</td>
<td>0.16945</td>
</tr>
<tr>
<td>5</td>
<td>0.12761</td>
<td>12</td>
<td>0.15487</td>
<td>19</td>
<td>0.17046</td>
</tr>
<tr>
<td>6</td>
<td>0.13073</td>
<td>13</td>
<td>0.15853</td>
<td>20</td>
<td>0.17123</td>
</tr>
<tr>
<td>7</td>
<td>0.13424</td>
<td>14</td>
<td>0.16166</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using as an initial guess the values
\[ \theta_0 = 1.0, \quad \theta_2 = 0.01, \quad \theta_3 = 1.0, \quad \theta_4 = 0.01, \]
all three algorithms converged to \( \hat{\theta} = (0.2, 0.004, 4.009) \) with zero residual sums of squares. The number of iterations necessary for convergence was 12, 13, and 7 for the modified Gauss-Newton, Marquardt, and Spiral algorithms, respectively. Thus the Spiral routine performed significantly better than the other two. An initial guess of
\[ \theta_0 = 5.0, \quad \theta_2 = 1.0, \quad \theta_3 = 2.0, \quad \theta_4 = 0.01 \]
did not lead to convergence for any of the three algorithms.

In our last example, from chemical engineering, we are trying to model reaction rate \( y_i \) as a function of temperature, \( \theta_0 \), \( x_{1i} \) nitrobenzene concentration \( x_{2i} \), hydrogen concentration \( x_{3i} \), and aniline concentration \( x_{4i} \). The equation is ([12])

\[
(3.4) \quad y_i = \theta_1 \exp\left(-\theta_2/Rx_{1i}\right)x_{2i}^{\theta_3}x_{3i}^{\theta_4}/\left(1+\theta_5x_{4i}^{\theta_6}\right).
\]

Data for a particular experiment is given by Table 3.3. The thesis from which this example is obtained erroneously estimated the parameters, but we use the estimates provided there as "reasonable" guesses for starting values.
<table>
<thead>
<tr>
<th>Reaction rate ($y_1$)</th>
<th>Temp. ($x_{11}$)</th>
<th>Nitrobenzene ($x_{21}(\times 10^8)$)</th>
<th>Hydrogen ($x_{31}(\times 10^7)$)</th>
<th>Aniline ($x_{41}(\times 10^8)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.15 $\times 10^4$</td>
<td>433</td>
<td>21.19</td>
<td>78.06</td>
<td>0</td>
</tr>
<tr>
<td>0.96 $\times 10^4$</td>
<td>423</td>
<td>20.93</td>
<td>81.98</td>
<td>0</td>
</tr>
<tr>
<td>3.64 $\times 10^4$</td>
<td>443</td>
<td>18.01</td>
<td>78.53</td>
<td>0</td>
</tr>
<tr>
<td>2.42 $\times 10^4$</td>
<td>436</td>
<td>20.26</td>
<td>75.04</td>
<td>0</td>
</tr>
<tr>
<td>1.76 $\times 10^4$</td>
<td>427.5</td>
<td>21.79</td>
<td>78.67</td>
<td>0</td>
</tr>
<tr>
<td>2.04 $\times 10^4$</td>
<td>431</td>
<td>21.61</td>
<td>78.03</td>
<td>0</td>
</tr>
<tr>
<td>2.44 $\times 10^4$</td>
<td>436</td>
<td>21.36</td>
<td>77.13</td>
<td>0</td>
</tr>
<tr>
<td>3.32 $\times 10^4$</td>
<td>433</td>
<td>41.18</td>
<td>117.48</td>
<td>0</td>
</tr>
<tr>
<td>13.98 $\times 10^4$</td>
<td>448</td>
<td>62.69</td>
<td>265.40</td>
<td>0</td>
</tr>
<tr>
<td>5.49 $\times 10^4$</td>
<td>423</td>
<td>65.61</td>
<td>265.40</td>
<td>0</td>
</tr>
<tr>
<td>1.67 $\times 10^4$</td>
<td>426</td>
<td>72.76</td>
<td>76.99</td>
<td>0</td>
</tr>
<tr>
<td>1.60 $\times 10^4$</td>
<td>426</td>
<td>37.40</td>
<td>78.37</td>
<td>0</td>
</tr>
<tr>
<td>1.78 $\times 10^4$</td>
<td>426</td>
<td>27.50</td>
<td>78.76</td>
<td>0</td>
</tr>
<tr>
<td>1.59 $\times 10^4$</td>
<td>426</td>
<td>20.95</td>
<td>78.69</td>
<td>0</td>
</tr>
<tr>
<td>0.83 $\times 10^4$</td>
<td>426</td>
<td>31.00</td>
<td>32.55</td>
<td>0</td>
</tr>
<tr>
<td>1.58 $\times 10^4$</td>
<td>426</td>
<td>30.75</td>
<td>61.51</td>
<td>0</td>
</tr>
<tr>
<td>2.00 $\times 10^4$</td>
<td>426</td>
<td>30.82</td>
<td>93.99</td>
<td>0</td>
</tr>
<tr>
<td>2.38 $\times 10^4$</td>
<td>426</td>
<td>30.65</td>
<td>129.14</td>
<td>0</td>
</tr>
<tr>
<td>1.98 $\times 10^4$</td>
<td>426</td>
<td>128.18</td>
<td>72.97</td>
<td>0</td>
</tr>
<tr>
<td>1.98 $\times 10^4$</td>
<td>426</td>
<td>59.12</td>
<td>78.09</td>
<td>0</td>
</tr>
<tr>
<td>7.45 $\times 10^4$</td>
<td>426</td>
<td>10.04</td>
<td>77.22</td>
<td>0</td>
</tr>
<tr>
<td>2.27 $\times 10^4$</td>
<td>426</td>
<td>38.88</td>
<td>78.70</td>
<td>0</td>
</tr>
<tr>
<td>1.03 $\times 10^4$</td>
<td>426</td>
<td>16.62</td>
<td>36.16</td>
<td>0</td>
</tr>
<tr>
<td>1.36 $\times 10^4$</td>
<td>426</td>
<td>77.70</td>
<td>80.37</td>
<td>0</td>
</tr>
<tr>
<td>1.85 $\times 10^4$</td>
<td>426</td>
<td>116.30</td>
<td>85.80</td>
<td>0</td>
</tr>
<tr>
<td>1.78 $\times 10^4$</td>
<td>426</td>
<td>38.29</td>
<td>77.35</td>
<td>0</td>
</tr>
<tr>
<td>2.18 $\times 10^4$</td>
<td>435</td>
<td>37.53</td>
<td>75.80</td>
<td>0</td>
</tr>
<tr>
<td>3.01 $\times 10^4$</td>
<td>445</td>
<td>36.69</td>
<td>74.10</td>
<td>0</td>
</tr>
<tr>
<td>0.53 $\times 10^4$</td>
<td>426</td>
<td>28.24</td>
<td>26.97</td>
<td>0</td>
</tr>
<tr>
<td>1.20 $\times 10^4$</td>
<td>426</td>
<td>32.18</td>
<td>48.09</td>
<td>0</td>
</tr>
<tr>
<td>1.40 $\times 10^4$</td>
<td>426</td>
<td>31.13</td>
<td>68.43</td>
<td>0</td>
</tr>
<tr>
<td>2.08 $\times 10^4$</td>
<td>426</td>
<td>29.27</td>
<td>111.74</td>
<td>0</td>
</tr>
<tr>
<td>1.92 $\times 10^4$</td>
<td>426</td>
<td>114.50</td>
<td>87.05</td>
<td>0</td>
</tr>
<tr>
<td>1.11 $\times 10^4$</td>
<td>426</td>
<td>26.66</td>
<td>79.03</td>
<td>3.06</td>
</tr>
<tr>
<td>0.95 $\times 10^4$</td>
<td>426</td>
<td>40.66</td>
<td>78.66</td>
<td>4.66</td>
</tr>
<tr>
<td>1.01 $\times 10^4$</td>
<td>426</td>
<td>35.62</td>
<td>78.83</td>
<td>4.09</td>
</tr>
<tr>
<td>1.35 $\times 10^4$</td>
<td>426</td>
<td>22.13</td>
<td>79.33</td>
<td>2.54</td>
</tr>
<tr>
<td>1.44 $\times 10^4$</td>
<td>426</td>
<td>21.88</td>
<td>79.38</td>
<td>2.51</td>
</tr>
<tr>
<td>0.23 $\times 10^4$</td>
<td>426</td>
<td>16.56</td>
<td>38.79</td>
<td>39.55</td>
</tr>
<tr>
<td>0.52 $\times 10^4$</td>
<td>426</td>
<td>12.58</td>
<td>38.90</td>
<td>30.01</td>
</tr>
<tr>
<td>0.36 $\times 10^4$</td>
<td>426</td>
<td>14.63</td>
<td>38.76</td>
<td>34.78</td>
</tr>
<tr>
<td>0.24 $\times 10^4$</td>
<td>426</td>
<td>15.34</td>
<td>36.26</td>
<td>36.64</td>
</tr>
</tbody>
</table>
Using,
\[ e_1 = 190.50742, \quad e_2 = 14719.64130, \quad e_3 = 0.05386, \quad e_4 = 1.01018, \quad e_5 = 0.00421, \quad e_6 = 0.25189, \]
and \( R \), the universal gas constant, \( 1.9869 \text{ cal} \cdot \text{C}^{-1} \cdot \text{mole} \), we obtain the following results:

<table>
<thead>
<tr>
<th></th>
<th>Mod. Gauss-Newton</th>
<th>Marquardt</th>
<th>Spiral</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of iterations</td>
<td>50*</td>
<td>50*</td>
<td>1*</td>
</tr>
<tr>
<td>SS Residual</td>
<td>209,844,885.6128651</td>
<td>212,585,078.6659825</td>
<td>218,669,377.140358</td>
</tr>
<tr>
<td>( \hat{\theta}_1 )</td>
<td>209.63174</td>
<td>191.3244</td>
<td>190.50742</td>
</tr>
<tr>
<td>( \hat{\theta}_2 )</td>
<td>14,714.2779</td>
<td>14,714.51367</td>
<td>14,719.64130</td>
</tr>
<tr>
<td>( \hat{\theta}_3 )</td>
<td>0.04487</td>
<td>0.05316</td>
<td>0.05386</td>
</tr>
<tr>
<td>( \hat{\theta}_4 )</td>
<td>1.01482</td>
<td>1.01043</td>
<td>1.01018</td>
</tr>
<tr>
<td>( \hat{\theta}_5 )</td>
<td>0.00001</td>
<td>0.00030</td>
<td>0.00471</td>
</tr>
<tr>
<td>( \hat{\theta}_6 )</td>
<td>0.56116</td>
<td>0.37345</td>
<td>0.23189</td>
</tr>
</tbody>
</table>

*No convergence after 50 iterations.
+No successful correction vector along path of steepest descent.

We cannot say that the Spiral algorithm led to convergence in view of the results of the other two procedures. Though neither the modified Gauss-Newton nor the Marquardt routines converged in 50 iterations, we shall see below that the modified Gauss-Newton was near convergence.

To avoid endless loops, all three routines require some arbitrary stopping point in searching for a suitable correction vector. (This idea is not to be confused with stopping the routine because the new, updated estimate for \( \theta \) does not significantly differ from the previous estimate. Under this condition we say the procedure has converged.) When such a condition is reached, as was the case for the above starting value using the Spiral routine, NONLIN terminates the procedure with an appropriate message. When one encounters this condition, additional starting values should be tried to insure that a potential convergence point has indeed been reached. To illustrate, the following starting values were tried,
\( \varepsilon_1 = 195, \varepsilon_2 = 14700, \varepsilon_3 = 0.0551, \varepsilon_4 = 1.011, \varepsilon_5 = 1.0, \varepsilon_6 = 0.05 \)

with results as follows:

<table>
<thead>
<tr>
<th></th>
<th>Mod. Gauss-Newton</th>
<th>Marquardt</th>
<th>Spiral</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of iterations</td>
<td>35</td>
<td>50*</td>
<td>Did not</td>
</tr>
<tr>
<td>( \hat{\theta}_1 )</td>
<td>209.57059</td>
<td>191.03260</td>
<td></td>
</tr>
<tr>
<td>( \hat{\theta}_2 )</td>
<td>14,714.27215</td>
<td>14,716.61624</td>
<td></td>
</tr>
<tr>
<td>( \hat{\theta}_3 )</td>
<td>0.04489</td>
<td>0.05345</td>
<td></td>
</tr>
<tr>
<td>( \hat{\theta}_4 )</td>
<td>1.01481</td>
<td>1.01031</td>
<td></td>
</tr>
<tr>
<td>( \hat{\theta}_5 )</td>
<td>0.00001</td>
<td>0.00202</td>
<td></td>
</tr>
<tr>
<td>( \hat{\theta}_6 )</td>
<td>0.56173</td>
<td>0.27874</td>
<td></td>
</tr>
</tbody>
</table>

*No convergence after 50 iterations.

Using starting values:

\( \varepsilon_1 = 100, \varepsilon_2 = 10,000, \varepsilon_3 = 0.01, \varepsilon_4 = 1.0, \varepsilon_5 = 0.001, \varepsilon_6 = 0.5 \),

the Marquardt and Spiral routines did not converge, but the modified Gauss-Newton converged to

\( \hat{\theta} = (209.59451, 14,714.15046, 0.04487, 1.01482, 0.00001, 0.56111)' \)

in 34 iterations with a residual sums of squares equal to 209,844,879.8751693. Changing \( \theta_3 \) and \( \theta_4 \) to .05 caused all three routines to not converge. Additional starting values tried, none of which led to convergence, were:

\[(100, 1000, 0, 1, .05, .5)'
\[(10, 10000, .5, 2, 1, .01)'
\[(1000, 100, .25, 3, 5, 2)'
\[(1, 100, 1, 5, 10, 5)'.

It is apparent, now, that the starting value problem is critical.

The problem is best handled by having prior information about the model and physical limitations of the parameters so that reasonable values can
be obtained. Here the statistician, or modeler, must be in close contact with the experimenter, particularly for the more complicated nonlinear functions such as (3.4).

Outside factors, such as physical or management constraints, or the desire to test some hypothesis about \( \theta \) (e.g., see [3]), lead to the need for least squares estimates, \( \hat{\theta} \), in
\[
y = f(x, \theta) + \varepsilon
\]
constrained by
\[
g_1(\theta) = g_2(\theta) = \ldots = g_r(\theta) = 0.
\]
From the theory of Lagrangian multipliers, we can achieve least squares estimators for \( \theta \) by minimizing
\[
\Phi(\theta, \lambda) = \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2 + \sum_{s=1}^{r} 2\lambda_s g_s(\theta)
\]
with respect to \( \theta \) and \( \lambda \), an \( r \times 1 \) vector of multipliers. Differentiating (4.1) with respect to \( \theta \) and \( \lambda \) and setting the results equal to zero yields the \( p + r \) normal equations
\[
(4.2) \quad \frac{\partial \Phi(\theta, \lambda)}{\partial \theta_j} = -2 \sum_{i=1}^{n} \left( y_i - f(x_i, \theta) \right) \frac{\partial f(x_i, \theta)}{\partial \theta_j} + 2 \sum_{s=1}^{r} \lambda_s \frac{\partial g_s(\theta)}{\partial \theta_j} = 0; \quad j = 1, \ldots, p
\]
\[
\frac{\partial \Phi(\theta, \lambda)}{\partial (2\lambda_s)} = g_s(\theta) = 0; \quad s = 1, \ldots, r.
\]

Given some initial estimate of \( \theta \), say \( \theta^0 \) expansion of \( f(x_i, \theta) \) in a Taylor series about \( \theta^0 \) yields the first order approximation
\[
f(x_i, \theta) \approx f(x_i, \theta^0) + \sum_{j=1}^{p} \frac{\partial f(x_i, \theta^0)}{\partial \theta_j} d_j,
\]
where
\[
d_j = (\theta_j - \tilde{\theta}_j) \text{ for some } \tilde{\theta}_j.
\]
Similarly,
\[
g_s(\theta) \approx g_s(\theta^0) + \sum_{j=1}^{p} \frac{\partial g_s(\theta^0)}{\partial \theta_j} d_j.
\]
Let \( \mathbf{d} = [(d_r)]_{q \times 1} \); \( \mathbf{G} = \left[ \left( \frac{\partial g_s(\theta^0)}{\partial \theta_j} \right) \right]_{r \times p} \); \( \mathbf{q}^0 = [(g_s(\theta^0))]_{p \times 1} \); and \( \mathbf{y}, \mathbf{z} \) be defined as before. Then upon substitution into (4.2), using our expanded approximations, we obtain

\[
-2\mathbf{Z}'\mathbf{y} + 2\mathbf{Z}'\mathbf{z} + 2\mathbf{G}'\lambda = 0
\]

\[
\mathbf{q}^0 + \mathbf{G}\mathbf{d} = 0,
\]

which can be rewritten as

\[
(4.3) \quad \begin{bmatrix} \mathbf{Z}' \mathbf{Z} & \mathbf{G}' \\ \mathbf{G} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Z}' \mathbf{y} \\ -\mathbf{q} \end{bmatrix},
\]

a linear system from which a solution for \( \mathbf{d} \) (our Gauss-Newton type correction vector) may be found. If we rewrite (4.3) as

\[
\mathbf{W} \mathbf{y} = \mathbf{h},
\]

with obvious notation, then \( \mathbf{d} \) consists of the first \( p \) elements of \( \mathbf{y} \), where

\[
\mathbf{y} = \mathbf{W}^{-1} \mathbf{h}.
\]

Once \( \mathbf{d} \) is obtained, we compute \( \mathbf{q}^1 = \mathbf{q}^0 + \mathbf{d} \), as before, and we begin the process again, using \( \mathbf{q}^1 \) as our "initial guess".

If \( g_s(\theta) \), for \( s = 1, \ldots, r \), is a linear function of \( \theta \), then \( g_s(\xi^1) = 0 \) will be satisfied at each iteration and thus we need only \( \phi(\xi^{j+1}) = \phi(\xi^j) \) as \( \sum_{s=1}^{r} g_s(\xi^j) = 0 \) in (4.1). If \( g_s(\theta) \), for some \( s \), is a nonlinear function of \( \theta \), then this is not the case, and we should verify that

\[
\phi(\xi^{j+1}, \lambda^{j+1}) = \phi(\xi^j, \lambda^j),
\]

noting that (4.3) allows for solutions of \( \lambda \), as well as \( \mathbf{d} \). The program NONLIN, described in the appendix, computes \( \phi(\xi^{j+1}) \) instead of \( \phi(\xi^{j+1}, \lambda^{j+1}) \), assuming \( g_s(\xi^{j+1}) \) to be sufficiently close to zero. This has presented no problems thus far, as demonstrated by the next example, which involves fairly nonlinear constraints. The program allows the choice of a modified
Gauss-Newton or a Spiral type search for \( \hat{\theta} \) in the constrained parameter space.

4.1 An example using growth curves.

Most growth curves are sigmoidal, i.e., symmetric, about some inflection point and thus are inappropriate to describe phenomena that are influenced by different factors at different times in their growth. Thus we may see a rapid growth in an animal until puberty, say, to be followed by a growth which is much less rapidly changing. A possible model for this type of behavior is to describe pre-puberty growth with one curve and maturity with another curve. Thus we are talking of the problem of segmenting two (or more) growth curves.

As an example, consider the dependent variable, \( y \), to be a segmented function of two sigmoidal growth curves as

\[
y_1 = \begin{cases} 
A_1(1-C_1\exp(-B_1x_1^2)) + \epsilon_1 & \text{for } x_1 < \gamma \\
F+A_2(1-C_2\exp(-B_2x_1^2)) + \epsilon_1 & \text{for } x_1 > \gamma.
\end{cases}
\]

(4.4)

Realistically, we require continuity and differentiability of \( y \) at \( x=\gamma \), giving the two constraints

\[
A_1(1-C_1\exp(-B_1\gamma^2)) = F+A_2(1-C_2\exp(-B_2\gamma^2))
\]

(4.5)

\[
2A_1B_1C_1\gamma\exp(-B_1\gamma^2) = 2A_2B_2C_2\gamma\exp(-B_2\gamma^2).
\]

Equation (4.5) becomes

\[
F = A_1(1-C_1\exp(-B_1\gamma^2)) - A_2(1-C_2\exp(-B_2\gamma^2))
\]

(4.6)

\[
A_2 = \frac{A_1B_1C_1}{B_2C_2}\exp(-\gamma(B_1-B_2)).
\]

Substitution of (4.6) into (4.4) yields
\[(4.7) \quad y_i = \begin{cases} \Lambda_i (1-C_1 \exp(-B_1 x_i^2)) + \epsilon_i & x_i \leq \gamma, \\ \Lambda_i (1-C_1 \exp(-B_1 x_i^2)) + B_1 C_1 \exp(-\gamma^2 B_1 (1-\exp(-B_2 (x_i^2 - \gamma^2))) + \epsilon_i & x_i > \gamma \end{cases} \]

which is (4.4) constrained by the continuity and differentiability restrictions of (4.5).

Suppose in (4.7) we let \(x_i = i, i = 1, \ldots, 20\), \(A_1 = 0.2\), \(B_1 = 0.004\), \(C_1 = 0.4\), and \(B_2 = 0.009\). From [5] it is seen that for the growth curve
\[y_i = A_1 (1-C_1 \exp(-B_1 x_i^2))\]
the inflection point is \(1/\sqrt{2B_1}\). We wish to generate data from (4.7), but, for realism, we do not want the data to "flatten out" past the first inflection before beginning the part of the data produced by the second curve. Thus we fix \(\gamma = 1/\sqrt{2B_1}\) for generation. This is not necessary in the fitting of (4.7). If data actually contained a relative "plateau", the procedure should handle it; and in fact we will not restrict \(\gamma = 1/\sqrt{2B_1}\) in our search for \(\gamma\), merely in the generation of the data. The data corresponding to (4.7) is that given after (3.3); in fact, (3.3) is (4.4) after reparameterizing to account for the constraints in (4.5). We should be able to estimate \(\theta\) with zero error, and this is the case using the program NONLIN (see discussion after (3.3)).

If our technique for constrained estimation is valid, we should be able to estimate all the parameters in (4.4) using the constraints of (4.5). Using the starting values
\[A_1 = .25, \quad B_1 = .01, \quad C_1 = .5, \quad F = .1, \quad A_2 = .2, \quad B_2 = .01, \quad C_2 = .5, \quad \gamma = 15,\]
the program NONLIN, using the modified Gauss-Newton option converged to
\[\hat{A}_1 = .2, \quad \hat{B}_1 = .004, \quad \hat{C}_1 = .4, \quad \hat{F} = -.02706, \quad \hat{A}_2 = .20011, \quad \hat{B}_2 = .009, \quad \hat{C}_2 = .33196, \quad \hat{\gamma} = 11.18034\]
in 18 iterations with zero residual error. Using the Spiral option, the
routine failed to converge. Fixing $\gamma=11.18034$ and estimating the remaining seven parameters allowed both options to converge in 10 iterations to the values listed above with zero residual error.
References


APPENDIX

A User's Guide for NONLIN

The following control cards are needed to use the program NONLIN, which is documented following this guide:

1. Title card. Anything punched on the first control card (80 characters) is printed at the beginning of the output (once) for identification.

2. Limits card. Enter problem limitations as follows:
   cc 1-3: Number of observations (a value of the dependent variable along with its associated values of the independent variables constitute one observation).
   cc 5-6: Number of parameters to be estimated.
   cc 8-9: Number of independent variables in model.
   cc 11: Number of method to be used,
      1-Modified Gauss-Newton
      2-Marquardt
      3-Spiral
   cc 13-14: Maximum number of iterations to be attempted.
   cc 16-17: Number of parameter constraints supplied.

3. Initial parameter estimates. Enter initial guesses sequentially as decimal numbers, using cc 1-10 for \( \theta_1 \), cc 11-20 for \( \theta_2 \), etc.; continue on next card if necessary. Use TRANS(B) subroutine to transform initial guesses if it is necessary to
rescale due to format limitations.

4. Format. FORTRAN format statement for data to follow, e.g.,
   \[(F3.0,2X,3F4.0,1X,F10.0)\]

5. Data. Read in one observation at a time, with the dependent variable read in first.

SUBROUTINE TRANS(B): A user supplied FORTRAN subroutine (optional) in which the user can change (e.g., rescale) any of the parameters or variables read in. TRANS(B) is called once upon completion of reading in the data.

SUBROUTINE FUNC(B,F): A user supplied subroutine (mandatory) which gives the form of the model to be used. Parameters are in array B, independent variables are in array X. When more than one independent variable is present, all of the first variable is stored sequentially in X, followed by all of the second variable, etc. To address the \(i^{th}\) value of the third variable, address the \(((2 \times \text{no. of obs.}) + i)^{th}\) member of X. The function is written as \(F(\ ) = \ldots\), \(F\) containing the "predicted" value of Y using the current parameter estimates in B. An example of the model in Eq. (3.2) is given in SUBROUTINE FUNC(B,F) in the documentation following this guide.

SUBROUTINE LGRANG(B,G): A user supplied subroutine (optional) used to enter any parameter constraints. The constraints should be written as \(0=(\text{constraint})\). The first constraint is
then entered as $G(1) = (\text{constraint})$, etc. As an example, to use the two constraints $\theta_1 = \theta_2$ and $\theta_1 = 1/\theta_2$, enter $G(1) = B(1) - B(2)$ and $G(2) = B(1) \cdot B(2) - 1$.

Output: The user supplied title, initial parameter estimates, limitations, and data, as read in, are printed on the first page of output. To print transformed estimates or data, include the appropriate WRITE statements in SUBROUTINE TRANS(B).

After each iteration, updated estimates of the parameters are printed along with the SS Residual using these estimates.

If the program terminates normally, the inverse of the sums of squares and cross products matrix $(Z'Z)^{-1}$ is printed as evaluated at the final parameter estimates.

Limitations: The following limitations (maximums) are employed in NONLIN, as listed. They may be expanded by changing appropriate array sizes and formats. If array sizes are altered, the user must be sure to change all affected arrays.

No. of observations: 100
No. of parameters: 15
No. of indep. variables: 15
No. of constraints: 15
ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE
C Y IS THE VECTOR OF OBSERVATIONS OF THE DEPENDENT VARIABLE
C X IS THE MATRIX OF OBSERVATIONS OF THE INDEPENDENT VARIABLES
C B IS THE VECTOR OF CURRENT PARAMETER ESTIMATES
C P IS THE VECTOR OF FUNCTION VALUES USING CURRENT PARAMETER ESTIMATES
C A IS THE MATRIX OF PARTIAL DERIVATIVES OF F EVALUATED AT B
C PT IS THE TRANSPOSE OF P

C IMPLICIT REAL*(4-A-H,O-$)
C REAL*8 LAMDA
C COMMON X(1500),Y(100),DELTA(15),SS,NBBS,NVAR,NPARN,N,NCNST
C DIMENSION TITLE(20),FMT(20),B(15),P(1500),PT(1500),
C IA(225),LI(15),NI(15)
C LAMDA=0.0100
C READ *(5,1000) TITLE
C 1000 FORMAT (20A4)
C WRITE *(6,1001) TITLE
C 1001 FORMAT (1H1,125,?0A4)
C READ *(5,1002) NBBS,NPARN,NVAR,NI,THE,MITER,NCONST
C 1002 FORMAT (13,1X,12,1X,12,1X,11,1X,12,1X,12)
C NBBS- NUMBER OF OBSERVATIONS
C NPARN- NUMBER OF PARAMETERS TO BE ESTIMATED
C NVAR- NUMBER OF INDEPENDENT VARIABLES
C NMETH- METHOD TO BE USED FOR NONLINEAR ESTIMATION
C 1- MODIFIED GAUSS-NEWTON (H.O. HARTLEY, TECHNOMETRICS, MAY 1961)
C 2- MARQUARDT (D.W. MARQUARDT, SIAM, 1963, P 461)
C 3- SPIRAL (A. JONES, COMPUTER JOURNAL, AUG, 1970)
C MITER- MAXIMUM NUMBER OF ITERATIONS TO BE ATTEMPTED
C NCONST- NUMBER OF PARAMETER CONSTRAINTS
C
C GO TO (101,102,103), NMETH
C 101 WRITE*(6,2000)
C 2000 FORMAT('ONLINEAR ESTIMATION USING MODIFIED GAUSS-NEWTON PROCEDUR
C 1E+',")
C GO TO 100
C 102 WRITE*(6,2001)
C 2001 FORMAT('ONLINEAR ESTIMATION USING MARQUARDT PROCEDURE.')
C GO TO 100
C 103 WRITE*(6,2002)
C 2002 FORMAT('ONLINEAR ESTIMATION USING SPIRAL PROCEDURE.')
C 100 WRITE*(6,2003)NBBS,NPARN,NVAR,NI,THE,NCNST
C 2003 FORMAT('NO. OF OBS= ',3J3,'NO. OF PARAMETERS= ',3J3,'NO. OF INDEP. VAR= ',1X,
C 1 'MAXIMUM NO. OF ITERATIONS= ',3J3,'NO. OF USER-SUPPLIED CONSTRANTS= ',3J3)
C READ *(5,1003)NBBS,NPARN
C 1003 FORMAT(8F10.5)
C WRITE*(6,2004)
C 2004 FORMAT('ORIGINAL PARAMETER ESTIMATES.')
C WRITE*(6,2005)I,J,NPARN
C 2005 FORMAT(12X,2I8,12)
C WRITE*(6,2006)I,J,NPARN
C 2006 FORMAT(1X,6F20.5)
C READ *(5,1004) FMT
C 1004 FORMAT(20A4)
C WRITE*(6,1005)
C 1005 FORMAT(32HDATA AS INPUT TO PROGRAM NONLIN)
C NU 2=1,NCBS
FORTRAN IV G LEVEL 21

0036 N=(NVAR-1)*NOBS+1
0037 C DEPENDENT VARIABLE IS READ IN FIRST
0038 C IF MORE THAN ONE INDEPENDENT VARIABLE, MATRIX X IS STORED AS A STRING
0039 C
0040 READ(5,FRMT)Y(I),I(X(J),J=1,N,NOBS)
0041 2 WRITE(6,1006)Y(I),I(X(J),J=1,N,NOBS)
0042 C USE TRANS(B) TO FORM ANY NEW VARIABLES, OR CHANGE VALUES OF VARIABLES WHICH
0043 C ARE BEYOND FORMATS. TRANS(B) IS CALLED ONLY ONCE.
0044 C
0045 CALL TRANS(B)
0046 CALL FUNC(B,F)
0047 SS=SS+SQS(F)
0048 WRITE(6,1006) SS
0049
0050 1006 FORMAT(10,23HINITIAL SUM OF SQUARES=,1X,F4.15)
0051 N=NPARAM*NPARM
0052 ITER=0
0053 5 ITER=ITER+1
0054 IF(ITER.GT.NITER) GO TO 70
0055 CALL DERIV(P,F,B)
0056 IERR3=1
0057 ITERM=0
0058 GO TO 110,20,30,NMETH
0059 10 CALL GAUSS(P,F,B,IERR3)
0060 GO TO 40
0061 20 CALL MODT(P,B,LAMBDA,F)
0062 GO TO 40
0063 30 CALL SPIRAL(P,F,B,IERR3)
0064 C CHECK FOR CONVERGENCE
0065 C
0066 40 GO TO 51,1,NPARAM
0067 IF((DABS(Delta(A(I))/.001D0+DABS(B(I))))).GT.1.D-6) GO TO 51
0068 50 CONTINUE
0069 GO TO 52
0070 51 ITERM=1
0071 52 WRITE(6,1007) ITER
0072 1007 FORMAT(1H0,T20,13HITERATION NO.,13)
0073 WRITE(6,1008)
0074 1008 FORMAT(1X,16HPARAMETER VALUES)
0075 WRITE(6,1009)(1,1=NPARAM)
0076 1009 FORMAT(1X,12,5(1X,12))
0077 WRITE(6,1010)(B(I),1=I,NPARAM)
0078 1010 FORMAT(1X,6(F20.5))
0079 WRITE(6,1011) SS
0080 1011 FORMAT(1H1 SUM OF SQUARES= ',F4.15)
0081 C IERR3 POINTS TO APPROPRIATE TERMINATION MESSAGE
0082 C
0083 072 GO TO 155,75,71,IERR3
0084 073 55 IF(ITERM.GT.0) GO TO 5
0085 074 60 WRITE(6,1012)
0086 075 1012 FORMAT(2H1PROCEDURE TERMINATED DUE TO APPARENT CONVERGENCE TO THE LISTED PARAMETER VALUES.)
0087 GO TO 80
0088 076 70 WRITE(6,1013) NITER
0089 077 1013 FORMAT(13HPROCEDURE DOESN'T CONVERGE AFTER,13,12H ITERATIONS.)
GO TO 80
71 WRITE(6,1015)
1015 FORMAT('PROCEDURE TERMINATED. NO SUCCESSFUL CORRECTION VECTOR FOUND ALONG PATH OF STEEPEST DESCENT. POSSIBLE CONVERGENCE POINT HAS BEEN REACHED.')
GO TO 80
75 WRITE(6,1014)
1014 FORMAT('PROGRAM TERMINATED. NO SUCCESSFUL CORRECTION VECTOR FOUND ALONG TAYLOR-SERIES PATH. POSSIBLE CONVERGENCE POINT HAS BEEN REACHED.')

C COMPUTE Z^2 INVERSE AND PRINT
C
80 CALL FUNC(B,F)
81 CALL DERIV(P,F,B)
82 J=0
83 DO 90 I=1,NPMS
84 IJ=I-NPMS
85 DO 90 J=1,NPARM
86 IJ=IJ+NPMS
87 JI=JI+1
88 PT(JI)=P(IJ)
89 CALL MCHMUL(A,PT,P,NPMS,NPMS,NPARM)
90 CALL DMINV(A,NPMS,D,L,M)
91 WRITE(6,1016)
1016 FORMAT('INVERSE OF SS & CROSS PRODUCTS MATRIX OF PARTIALS EVALUATED AT FINAL POINT')
92 K=(NPARM-1)*NPARM
93 DO 91 I=1,NPARM
94 K=K+1
95 WRITE(6,1017)(A(J),J=1,N,NPARM)
96 FORMAT(1X,10E12.5)
97 STOP
98 END
FUNCTION SSQS(F)
IMPLICIT REAL*(A-H,O-Z)
COMMON X(1500),Y(100),DELT(15),SS,NOBS,NVAR,NPARM,N,NCONST
DIMENSION F(NOBS)

C SUBRUTINE COMPUTES ERROR SUM OF SQUARES FOR A PARTICULAR VALUE OF D
C
CALL ERRSET(207,256,-1,1)
SSQS=0.0
DO 1 I=1,NOBS
SSQS=SSQS+(Y(I)-F(I))*(Y(I)-F(I))
1 CALL OVERFL(J)
IF(J.EQ.1) GO TO 2
CONTINUE
2 SSQS=1.074
3 CALL ERRSET(207,256,256,2)
RETURN
END
SUBROUTINE DERIV(P,F,B)
IMPLICIT REAL*8(A-H,O-Z)
COMMON X(1500),Y(100),DELTA(15),SS,NORS,NVAR,NPARM,N,NCONST
DIMENSION B(15),BO(15),F(100),FO(100),P(1500)

DO 1 I=1,NPARM
    B0(J)=B(J)
DO 2 J=1,NORS
    H=R(I)*.001DO
    H1=1.DO/H
    BO(I)=B(I)+H
    CALL FUNC(BO,FO)
    DO 2 J=1,NORS
    P((I-1)*NORS+J)=(FO(J)-F(J))*H1
1 CONTINUE
2 CONTINUE
RETURN
END
SUBROUTINE MMULT(AB, A, B, L, M, N)

IMPLICIT REAL*8(A-H,O-Z)

DIMENSION AB(L,N), A(L,M), B(M,N)

DO 3 I=1,L
  DO 2 J=1,N
    AB(I,J)=0,0
  2 CONTINUE

1 AB(I,J)=A(I,K)*B(K,J)+AB(I,J)

2 CONTINUE

3 CONTINUE

RETURN

END
SUBROUTINE GAUSS(P,F,B,IER3)

IMPLICIT REAL*8(A-H,O-Z)

COMMON X(1500),Y(100),DELTA(15),SS,NOBS,NVAR,NPARM,N,NCONST

DIMENSION P(1500),F(100),B(15),A(225),PT(1500),YF(100),TEMP(30),

L(30),M(30),B1(15),B2(15),W(900),G(15),GDERRVIC(225),DELTA(30)

C FOLLOWING ROUTINE CALCULATES PT FROM P

C

J1=0

DO 1 I=1,NOBS

1 J1=J1+1

DO 1 J=1,NPARM

PT(J1)=P(J)

YF IS Y-F

DO 5 I=1,NCBS

5 YF(I)=Y(I)-F(I)

CALL MMULT(TEMP,PT,YF,NPARM,NOBS,1)

A=PT*P

CALL MMULT(A,PT,P,NPARM,NOBS,NPARM)

IF(NCONST.EQ.0) GO TO 180

C IF NCONST NOT 0, NEED TO BUILD AUGMENTED MATRIX

N=NPARM+ACCAST

N2=N*N

DO 100 I=1,N2

100 W(I)=0.0

CALL LGRANG(B,G)

CALL LGDRV(B,G,GDERIV)

DO 130 I=1,NPARM

K=(I-1)*N

K1=(I-1)*NPARM

K2=(I-1)*NCONST

DO 110 J=1,NPARM

110 W(K+J)=A(K1+J)

K=K+NPARM

DO 120 J=1,NCONST

120 W(K+J)=GDERIV(K2+J)

130 CONTINUE

K=NPARM*N

DO 140 I=1,NCONST

K1=(I-1)*NPARM

K2=(I-1)*N

DO 137 J=1,NPARM

137 DO 140 J=1,NPARM

140 W(K+K2+J)=GDERRVIC(J-1)*NCONST+I

150 CONTINUE

WRITE(6,3000)

3000 DC 151 I=1,N

WRITE AUGMENTED MATRIX, VALUE OF CONSTRAINTS, AND LAGRANGE MULTIPLIERS SO

USER CAN CHECK IF CONSTRAINTS ARE NEAR ZERO AT CONVERGENCE
C
0043 151 WRITE(6,3001)(W(J),J=1,N1,N)
0044 3000 FORMAT(*/ AUGMENTED MATRIX = */)
0045 3001 FORMAT(1X,9(1X,E12.5))
0046 CALL DMINV(N,N,D,L,M)
0047 DO 160 I=1,NCONST
0048 WRITE(6,40001),GI(I)
0049 4000 FORMAT(*VALUE OF CONSTRAINT NO. ,12,F = ,1,E12.5)
0050 160 TEMP(NPARM+1)=G(I)
0051 CALL MMULT(DELTA,W,TEMP,N,N,1)
0052 DO 170 I=1,NPARM
0053 170 DELTA(I)=DELTA(I)
0054 WRITE(6,40011)(DELTA(K+NPARM), K=1,NCONST)
0055 4001 FORMAT(*OLAMRDA= ,1X,9E12.5)
0056 GO TO 190
0057 180 CALL DMINV(A,NPARN,D,L,M)
C
0058 C  DELTA={PT+P}INVERSE PT*(Y-F) (SEE DRAPER & SMITH, 1966, P 268)
C
0059 CALL MMULT(DELTA,A,TEMP,NPARN,NPARN,1)
0060 190 V=1,0DO
0061 6 VI=,500*V
0062 DO 10 I=1,NPARM
0063 B1(I)=B(I)+VI*DELTA(I)
0064 10 B2(I)=B(I)+V*DELTA(I)
0065 CALL FUNC(B1,F)
0066 SS1=SSQSF(F)
0067 CALL FUNC(B2,F)
0068 SS2=SSQSF(F)
0069 VI=4,500*2,500*(SS-SS2)/(SS2-SS1-SS1+SS1)*V
0070 DO 20 I=1,NPARM
0071 20 B1(I)=B(I)+VI*DELTA(I)
0072 CALL FUNC(B1,F)
0073 SS1=SSQSF(F)
0074 IF(SS1.LT.SS) GO TO 30
0076 IF(V.LT.0,0DO) GO TO 50
0075 V=500*V
0077 GO TO 6
0078 30 DO 40 I=1,NPARN
0079 40 B2(I)=B1(I)
0080 SS=SS1
0081 GO TO 60
0082 50 ERR=2
0083 60 RETURN
0084 END
SUBROUTINE MQDT(P,B,LAMBOA,F)
REAL*8 LAMBOA, LAM
COMMON X(1500), Y(100), DELTA(15), SS, NOBS, NVAR, NPARM, N, NCONST
DIMENSION P(1500), F(100), B1(15), A(225), PT(1500), TEMP(15), YF(100),
AOLD(225), B1(15), B2(15)
C FORM SSCP MATRIX OF PARTIALS AND STORE IN A
C
J1=0
DO 1 I=1, NCBS
IJ=I-NOBS
DO 1 J=1, NPARM
IJ=IJ+NOBS
1 CONTINUE
1 PT(IJ)=P(IJ)
CALL MULT(A,PT,P,NPARM,NCBS,NPARM)
DO 2 I=1, NOBS
2 YF(I)=Y(I)-F(I)
CALL MULT(TEMP,PT,YF,NPARM,NOBS,1)
DO 3 I=1, N
3 AOLD(I)=A(I)
C PERFORM SCALE TRANSFORMATION ON MATRIX OF PARTIALS, A
C
DO 4 I=1, NPARM
II=(I-1)*NPARM
RDOT=DSQRT(AOLD(I)*AOLD(I+I))
TEMPI=TEMPI/RDOT
DO 4 K=1, NPARM
IJ=II+K
4 A(IJ)=AOLD(IJ)/(ROOT*DSQRT(AOLD(I-II-NPARM)*K))
C BEGIN MARQUARDT ITERATIVE PROCEDURE
C
LAM1=LAMBOA*G.1DO
CALL MQDT2(A,B,B2,TEMP,LAM1,SS2,F,AOLD)
IF(FSS2.LE.SS) GO TO 10
CALL MQDT2(A,B,B1,TEMP,LAMBOA,SS1,F,AOLD)
IF(FSS1.LE.SS) GO TO 7
LAMDA=LAMBOA*10.D0
CALL MQDT2(A,B,B1,TEMP,LAMBOA,SS1,F,AOLD)
GO TO 5
GO TO 5
10 SS=SS2
LAMBOA=LAM1
DO 6 I=1, NPARM
6 PI(I)=B2(I)
GO TO 9
7 SS=SS1
DO 8 I=1, NPARM
8 BI(I)=BI(I)
GO TO 9
9 RETURN
END
SUBROUTINE MQDT2(A,B,D1,TEMP,LAGDBA,SSQ,F,AOLD)
IMPLICIT REAL*8(A-H,O-Z)
REAL*8 LAGDBA,LAM
COMMON XL1(100),Y(100),DELTA(15),SS,NOBS,NVAR,NPARA,N,NCNST
DIMENSION A(225),F(100),B1(15),B(15),TEMP(15),A(225),L(15),M(15),
1 AOLD(225)
DO 10 I=1,N
10 AI(I)=A(I)
DO 1 I=1,NPAR
II=(I-1)*NPAR+1
1 AI(II)=AI(II)+LAGDBA
CALL DMINV(AI,NPARA,D,L,M)
CALL MFULT(Delta,AI,TEMP,NPARA,NPARA,1)
C RESCALE CORRECTION VECTOR, DELTA
DO 4 I=1,NPAR
4 DELTA(I)=DELTA(AI)/DSQRT(AOLD(I-1)*NPARA+1))
DO 5 I=1,NPAR
5 BL(I)=B(I)+DELTA(I)
CALL FUA(C,B1,F)
SSQ=SSQ+F
RETURN
END
0001 SUBROUTINE SPIRAL(P,F,B,IERR)
0002 1*IMPLICIT REAL*8(A-H,O-Z)
0003 COMMON X(1500),Y(1001),DELTA(15),SS,NBBS,NVAR,NPARM,N,NCONST
0004 DIMENSION P(1500),PT(1500),A(225),F(1001),YF(1001),B(15),TEMP(130),
0005 1B1(15),S1(15),SSQ(101),W(1000),G(15),GDERIV(225),DELTA(130),L(30),
0006 2M(30)
0007 REAL*8 U(101),1DO,1,307692307922306,
0008 4,74705882594155,6400,794087804878049,876712328767122,
0009 2,93430656934064,966037735849055,962725527831093/

C INITIAL PART OF ROUTINE IS SIMILAR TO GAUSS ROUTINE

0010 JJ=0
0011 DO 1 I=1,NBBS
0012 IJ=I-NBBS
0013 DO 2 J=1,NPARM
0014 JI=IJ+NPARM
0015 PT(JI)=P(IJ)
0016 CALL MMULT(A,PT,P,NPARM,NBBS,NPARM)
0017 DO 2 I=1,NBBS
0018 N=NCONST*NPARM
0019 K2=2*N
0020 DO 1100 I=1,N2
0021 1100 W(I)=0.00
0022 CALL LGRANG(B,G)
0023 CALL LGDROV(B,G,GDERIV)
0024 DO 1130 I=1,NPARM
0025 K=(I-I+1)*N
0026 K2=(I-I+1)*NCONST
0027 K1=(I-I+1)*NPARM
0028 DO 1130 J=1,NPARM
0029 1130 CONTINUE
0030 DO 1120 J=1,NCONST
0031 1120 W(I+J)=GDERIV(K2+J)
0032 DO 1130 CONTINUE
0033 DO 1150 I=1,NCONST
0034 K=NPARM*W
0035 DO 1150 I=1,N2
0036 K1=(I-I+1)*NPARM
0037 K2=(I-I+1)*N
0038 DO 1150 J=1,NPARM
0039 1150 CONTINUE
0040 DO 1150 I=1,N2
0041 WRITE(6,3000)
0042 3000 FORMAT(/'AUGMENTED MATRIX = /)
0043 WRITE(6,3001)W(J,J),J=1,NL,N
0044 3001 FORMAT(1X,9(I12.5))
0045 DO 1160 I=1,NCONST
0046 WRITE(6,4000)I,G(I)
0047 4000 FORMAT('VALUE OF CONSTRAINT NO. I,I2,’ = ',12.5)
0048 DO 1160 I=1,NCONST
0049 WRITE(6,4000)I,G(I)
0050 4000 FORMAT('VALUE OF CONSTRAINT NO. I,I2,’ = ',12.5)
0051 1160 CONTINUE
CALL MMULT(DELTA1',H,TEMP,N,N,1)
0053 DO 1170 I=1,NPARAM
0054 1170 DELTA1(I)=DELTA1(I)
0055 WRITE(6,4001)(DELTA1(K+NPARAM),K=1,NCONST)
0056 4001 FORMAT(*,CLAMBDAT=\*/1X,9E12.5)
0057 GO TO 4
0058 CALL MMULT(DELTA,A,TEPP,NPARAM,NPARAM,1)
0059 CALL DMINVIA(NPARAM,D,L,M)
0060 4 IFLAG=0
0061 5 DO 10 I=1,NPARAM
0062 10 B(I)=B(I)+DELTA1(I)
0063 CALL FUNC(B1,F)
0064 SS2=SSQS(F)
0065 IF(SS2.LT.SS) GO TO 40
0066 DO 20 I=1,NPARAM
0067 20 B(I)=B(I)+.5O0*DELTA1(I)
0068 CALL FUNC(B1,F)
0069 SS1=SSCS(F)
0070 IF(SS1.LT.SS) GO TO 50
0071 V=5D0+.2500*(SS-SS2)/(SS2+SS-SS1-SS1)
0072 DO 30 I=1,NPARAM
0073 30 B(I)=B(I)+V*DELTA1(I)
0074 CALL FUNC(B1,F)
0075 SS1=SSQS(F)
0076 IF(SS1.LT.SS) GO TO 50
0077 GO TO 60
0078 40 SS=SS2
0079 GO TO 60
0080 50 SS=SS1
0081 60 DO 70 I=1,NPARAM
0082 70 DELTA1(I)=B(I)-B(I)
0083 GO TO 230

BEGIN SEARCH ALONG THE SPIRAL

80 T=0,DO
0086 GRAD=0,DO
0087 DO 85 I=1,NPARAM
0088 85 DELTA1(I)=DELTA1(I)*V

TEMP VECTOR LIES IN DIRECTION OF STEEPEST DESCENT. NEED TO GET LENGTH OF TEMP & RESCALED TO LENGTH OF DELTA WHICH IS GAUSS-NEWTON CORRECTION

89 DO 90 I=1,NPARAM
0090 T=T+DELTA1(I)*DELTA1(I)
0091 90 GRAD=GRAD+TEMP(I)*TEMP(I)
0092 TD=DSCR(T)/DSCR(Grad)
0093 DO 100 I=1,NPARAM
0094 100 TEMP(I)=TEPP(I)*TD
0095 TMIND=0,DO

THIS SECTION IS STRAIGHTFORWARD APPLICATION OF SPIRAL ALGORITHM AS PUBLISHED BY JONES

DO 110 I=1,NPARAM
0096 110 TMIND=TMIND+(TFMP(I)-DELTA1(I))*TFMP(I)-DELTA1(I))
0097 COSG=1,DO-(TMIND/(T+T))
0099  GAMMA=CARCOS(COSG)
0100  SING=DSINC(GAMMA)
0101  I=0
0102  120 I=I+1
0103  CALL SPRL2(TEMP,BS,Bl,V1,SSQ(I),GAMMA,SING,COSG,S,F)
0104  SS1=SSQ(I)
0105  IF(SSL.LT.SS) GO TO 200
0106  IF(I.LT.3) GO TO 120
0107  I1=I-1
0108  I2=I-2
0109  IF(SSQ(I1).GT.SSQ(I2).OR(SSQ(I1).GT.SS1)) GO TO 130
0110  U1=U(I2)
0111  U2=U(I1)
0112  U3=U(I)
0113  U4=U1-U2
0114  U5=U1*U1-U2*U2
0115  U6=U1-U3
0116  U7=U1-U3-U1*U3
0117  U8=(SSQ(I2)-SSQ(I1))*U7-(SSQ(I2)-SSQ(I1))*U5)/(U4*U7-U6*U5)
0118  U9=-U8*U5/(SSQ(I2)-SSQ(I1)-U5*U4)
0119  CALL SPRL2(TEMP,BS,Bl,V1,SSL,GAMMA,SING,COSG,S,F)
0120  IF(SSL.LT.SS) GO TO 200
0121  130 CONTINUE
0122  IF(U(I+1).LT.9700) GO TO 120
0123  IFLAG=IFLAG+1
0124  IF(FLAG.GT.3) GO TO 145

C IF NO SUCCESSFUL POINT IS FOUND, HALVE THE LENGTH OF DELTA & REPEAT
C (UP TO 3 TIMES)
C
0125  DO 140 I=1,APARM
0126  140 DELTA(I)=.5D0*DELTA(I)
0127  GO TO 89
0128  145 SLCN=0
0129  DO 150 I=1,NPARM
0130  150 SLCN=SLCN+SI(I)*SI(I)
0131  TD=1.DO/TD
0132  SG=TD*DSQRT(SLCN)/DSQRT(Grad)
0133  DO 160 I=1,APARM
0134  160 TEMP(I)=TEMP(I)*SG
0135  170 DO 180 I=1,NPARM
0136  180 TEMP(I)=.5D0*TEMP(I)
0137  CALL FUNC(B1,F)
0138  SS1=SSQ(F)
0139  IF(SSL.LT.SS) GO TO 200
0140  DO 190 I=1,NPARM
0141  190 TEMP(I).GT.1.D-6) GO TO 170
0142  190 CONTINUE
0143  GO TO 220
0145  200 DO 210 I=1,NPARM
0146  210 DELTA(I)=B1(I)-B1(I)
0147  210 B1(I)=B1(I)
0148  SS=SS1
0149  GO TO 230
0150  220 IERROR=3
0151  230 RETURN
0152  END
SUBROUTINE SPRL2(TEMP,B1,U,V1,SS1,GAMMA,SING,COSG,S,F)
IMPLICIT REAL*8(A-H,O-Z)
COMMON XL500,Y(100),DELTA(15),SS,NRBS,NVAR,NPARAM,N,NCNST
DIMENSION TEMPS(15),BL(15),BL(15),S(15),F(100)

C
C THIS SUBROUTINE COMPUTES COORDINATES OF POINT ON SPIRAL TO BE SEARCHED.
C SEE JONES FOR DETAIL
C
BINCR=.5D0
CD 1 I=1,NPARAM
BL(I)=B(I)+DELTA(I)
1 S(I)=B(I)+DELTA(I)
THETA=DATAN1(U*SING)/(1.0D0-U+U*COSG)
COSB=DCS1(GAMMA+B1CR)
TG=THETA/GAMMA
TL=(1.0D0-THETA*COSB-(1.0D0-GAMMA*COSB)*TG*TG)
XSI=U*SING/DSIN(THETA)
XSI=T1/XSI
DO 120 I=1,NPARAM
S(I)=XSI*(LTEMPS(I)+(1.0D0-U)*DELTA(I))
120 BL(I)=B(I)+S(I)
CALL FUNC(BL,F)
SS1=SSQS(F)
IF(SS1.LT.1.0D4) GO TO 2

C
C GET HERE WHEN WE HAVE OVERFLOWED SSQS FUNCTION NEED TO DECREASE
C INCIDENT ANGLE OF SPIRAL
C
BINCR=BINCR*.5D0
GO TO 3
2 RETURN
0C24 END
SUBROUTINE MINV DMINV

PURPOSE
INVERT A MATRIX

USAGE
CALL MINV(A,N,D,L,M)

DESCRIPTION OF PARAMETERS
A - INPUT MATRIX, DESTROYED IN COMPUTATION AND REPLACED BY
RESULTANT INVERSE.
D - RESULTANT DETERMINANT
L - WORK VECTOR OF LENGTH N
M - WORK VECTOR OF LENGTH N

REMARKS
MATRIX A MUST BE A GENERAL MATRIX

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE

METHOD
THE STANDARD GAUSS-JORDAN METHOD IS USED. THE DETERMINANT
IS ALSO CALCULATED. A DETERMINANT OF ZERO INDICATES THAT
THE MATRIX IS SINGULAR.

*****************************************************************************

0001 SUBROUTINE MINV(A,N,D,L,M)
0002 DIMENSION A(225),L(15),M(15)

*****************************************************************************

IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
STATEMENT WHICH FOLLOWS.

DOUBLE PRECISION A,D,BIGA,HOLD,DABS

APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
ROUTINE.

THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO
CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS. ABS IN STATEMENT
10 MUST BE CHANGED TO DABS.

*****************************************************************************

SEARCH FOR LARGEST ELEMENT

D=1.0
0005 NK=-N
0006 DO 80 K=1,N
0007 NK=NK+1
0008 L(K)=K
0009 M(K)=K
0010 KK=NK+K

0011 80 D(K)=DD(K)/D(K-1)

*****************************************************************************
FORTRAN IV G LEVEL 21

BIGA=A(KK)
0012  DO 20 J=K,N
0013  IZ=N*(J-1)
0014  DO 20 I=K,N
0015    IJ=I+1
0016  10 IF(DABS(BIGA)-DABS(A(IJ))) 15,20,20
0017  15 BIGA=A(IJ)
0018  20 CONTINUE

C INTERCHANGE ROWS

0021  J=L(K)
0022    IF(J-K) 35,35,25
0023  25 KI=K-N
0024  DO 30 I=1,N
0025    KI=KI+N
0026    HOLD=A(KI)
0027    JI=KI+K+J
0028    A(KI)=A(JI)
0029  30 A(JI)=HOLD

C INTERCHANGE COLUMNS

0030  35 I=M(K)
0031    IF(I-K) 45,45,38
0032    JP=N*(I-1)
0033  DO 40 J=1,N
0034    JK=JK+J
0035    JI=JP+J
0036    HOLD=A(JK)
0037    A(JK)=A(JI)
0038  40 A(JI)=HOLD

C DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS
CONTAINED IN BIGA)

0039  45 IF(BIGA) 48,46,48
0040  46 D=0.0
0041  RETURN
0042  48 DO 55 I=1,N
0043    IF(I-K) 50,55,50
0044  50 IK=JK+I
0045    A(IK)=A(IK)/(-BIGA)
0046  55 CONTINUE

C REDUCE MATRIX

0047  DO 65 I=1,N
0048    IK=IK+I
0049  DO 65 J=1,N
0050    IJ=I-N
0051  60 IF(I-J) 62,65,62
0052    IJ=IJ+N
0053    IF(I-K) 60,65,60
0054  62 KJ=IJ+K
0055  65 CONTINUE
0056   A(IJ)=HOLD=A(KJ)+A(IJ)
0057  65 CONTINUE
C
C   DIVIDE ROW BY PIVOT
C
0058   KJ=K-N
0059   DO 75 J=1,N
0060   KJ=KJ+N
0061   IF(J-K) 70,75,70
0062   70 A(KJ)=A(KJ)/BIGA
0063   75 CONTINUE
C
C   PRODUCT OF PIVOTS
C
0064   D=D*BIGA
C
C   REPLACE PIVOT BY RECIPROCAL
C
0065   A(KK)=1.0/BIGA
0066  80 CONTINUE
C
C   FINAL ROW AND COLUMN INTERCHANGE
C
0067   K=N
0068  100 K=(K-1)
0069   IF(K) 150,150,165
0070  105 I=L(K)
0071   IF(I-K) 120,120,1C8
0072  108 JQ=N*(I-1)
0073   JR=N*(I-1)
0074   DO 74 J=1,N
0075   JQ=JQ+J
0076   HOLD=A(JK)
0077   JI=JR+J
0078   A(JK)=-A(JI)
0079  110 A(JI)=HOLD
0080  120 J=H(K)
0081   IF(J-K) 100,100,125
0082  125 KI=K-N
0083   DO 130 I=1,N
0084   KI=KI+N
0085   HOLD=A(KI)
0086   JI=KI-K+J
0087   A(KI)=-A(JI)
0088  130 A(JI)=HOLD
0089   GO TO 100
0090  150 RETURN
0091 END
SUBROUTINE LGDRV(B,G,GERIV)
IMPLICIT REAL*8(A-H,O-Z)
COMMON X(1500), Y(100), DELTA(15), SS, NORS, NVAR, NPAM, N, NCONST
DIMENSION B(15), BO(15), G(15), GO(15), GERIV(225)
C
C SUBROUTINE COMPUTES DERIVATIVES OF CONSTRAINTS W.R.T. PARAMETERS
C
DO 3 I=1,NPAM
DO 1 J=1,NFARM
1 BO(J)=B(J)
H=R(1)*.00100
H1=1.D0/H
BO(I)=B(I)+H
CALL LGRANG(B0, G0)
M=(I-1)*NCONST
DO 2 J=1,NCONST
2 GERIV(I+J)=(G0(J)-G(J))*H1
3 CONTINUE
RETURN
END
SUBROUTINE TRANS(B)
IMPLICIT REAL*8 (A-H, O-Z)
COMMON X(1500), Y(100), DELTA(15), SS, NORS, NVAR, NPARM, N, NCONST
DIMENSION B(15)
RETURN
END
SUBROUTINE LGRANG(B,G)
IMPLICIT REAL*8(A-H,O-Z)
COMMON X(1500),Y(100),DELTA(15),SS,NORES,NVAR,NPARM,N,NCONST
DIMENSION B(15),G(15)
CC
CC CONSTRAINTS CV PARAMETERS ARE PLACED HERE
CC
CC WRITE CONSTRAINTS AS 0=CCONSTRAINT, G(1) IS FIRST CONSTRAINT, ETC...
CC
RETURN
END
SUBROUTINE FUNC(B,F)
IMPLICIT REAL*8(A-H,O-Z)
COMMON X(1500),Y(100),DELTA(15),SS,NOBS,NVAR,NPARM,N
DIMENSION B(15),F(100)

FUNCTION TO BE MODELED IS SUPPLIED HERE

DO 1 I=1,NOBS
IN1=I+NOBS
F(I)=DEXP(-B(1))*X(I)*DEXP((-B(2))/X(IN1))
1 CONTINUE
RETURN
END
EXAMPLE PROBLEM FROM BARD, P.124

NONLINEAR ESTIMATION USING MARQUARDT PROCEDURE.

NO. OF OBS. = 15 NO. OF PARAMETERS = 2 NO. OF INDEP. VAR. = 2
MAXIMUM NO. OF ITERATIONS = 30 NO. OF USER-SUPPLIED CONSTRAINTS = 0

ORIGINAL PARAMETER ESTIMATES:
1  750.00000  1200.00000
2

DATA AS INPUT TO PROGRAM NCALIN
0.98000  0.10000  100.00000
0.98300  0.20000  100.00000
0.99500  0.30000  100.00000
0.97900  0.40000  100.00000
0.99300  0.50000  100.00000
0.62600  0.05000  200.00000
0.54400  0.10000  200.00000
0.45500  0.15000  200.00000
0.22500  0.20000  200.00000
0.16700  0.25000  200.00000
0.31700  0.04000  300.00000
0.56600  0.02000  300.00000
0.03400  0.06000  300.00000
0.01600  0.08000  300.00000
0.06600  0.10000  300.00000

INITIAL SUM OF SQUARES = 1.090440905418776

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>PARAMETER VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>152.76696</td>
</tr>
<tr>
<td></td>
<td>703.14640</td>
</tr>
<tr>
<td>SUM OF SQUARES = 0.448321270868956</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>PARAMETER VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>386.29234</td>
</tr>
<tr>
<td></td>
<td>929.50826</td>
</tr>
<tr>
<td>SUM OF SQUARES = 0.409038973823016</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>PARAMETER VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>624.68082</td>
</tr>
<tr>
<td></td>
<td>928.70228</td>
</tr>
<tr>
<td>SUM OF SQUARES = 0.055969223488254</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>PARAMETER VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>804.18994</td>
</tr>
<tr>
<td></td>
<td>965.48472</td>
</tr>
<tr>
<td>SUM OF SQUARES = 0.040732717907191</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>PARAMETER VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>818.35389</td>
</tr>
<tr>
<td></td>
<td>562.69427</td>
</tr>
<tr>
<td>SUM OF SQUARES = 0.039807476329395</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>PARAMETER VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
ITERATION NO. 7

PARAMETER VALUES

1  812.66423  2  960.65815

SUM OF SQUARES = 0.0398061361370666

ITERATION NO. 8

PARAMETER VALUES

1  812.66032  2  960.65862

SUM OF SQUARES = 0.03980613503003

ITERATION NO. 9

PARAMETER VALUES

1  812.65999  2  960.65865

SUM OF SQUARES = 0.03980613502997

ESTIMATION PROCEDURE TERMINATED DUE TO APPARENT CONVERGENCE TO THE LISTED PARAMETER VALUES.

INVERSE OF SS & CROSS PRODUCTS MATRIX OF PARTIAL β EVALUATED AT FINAL β:

0.158500  0.542510  0.07
0.542510  0.153970  0.07
Acknowledgements

I wish to thank Dr. George Milliken for his technical assistance and moral support while preparing this report and throughout my graduate studies at Kansas State University; his open door has always been a welcome sight.

Gratitude is also extended to Drs. Dayton and Fryer, and the Department of Statistics for financial support during my studies.

Deepest thanks are reserved for my wife, Evelyn, who devotedly backed my every whimsical notion and decision. It has been her encouragement that has helped me strive for the self-development one hopes to gain by graduate study.
A STUDY OF THREE ALGORITHMS FOR NONLINEAR LEAST SQUARES PARAMETER ESTIMATION

by

MICKEY LINN STILSON

B.S., Kansas State University, 1971

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Statistics

KANSAS STATE UNIVERSITY

1976
Two analytic approaches toward minimizing the residual sum of squares function of the nonlinear model are presented. The first uses a first order Taylor series approximation of the nonlinear function; the second uses a gradient search. Both general techniques entail an iterative scheme for finding the least squares parameter estimate.

From these two general approaches three currently used algorithms are developed. Examples are presented which show that no one algorithm is best for all problems. The topic of parameter estimation under parameter constraints is then discussed. An appendix documenting a computer program written by the author incorporating the algorithms and constrained estimation is provided.