BUREAU OF MINERAL RESOURCES, GEOLOGY AND GEOPHYSICS

RECORD

Record 1979/76

ESTIMATION OF PARAMETERS USING LEAST SQUARES

R.F. Moore

Record 1979/76

ESTIMATION OF PARAMETERS USING LEAST SQUARES

R.F. Moore

Contents

Summary

The method of estimation of parameters using least squares is being used more and more in BMR in techniques of automatic parameter estimation and interpretation, especially in the Geophysical Branch. The method of estimation of linear parameters is used extensively by the Airborne Reductions Group for fitting polynomials to (x,y) data, especially in the flight path recovery and magnetic data levelling processes.

The method of estimation of non-linear parameters, more commonly referred to as "inversion" in recent years, is, however, receiving the greatest amount of attention and is being extensively used in "automatic" fitting of analytical models to geophysical data, in particular gravity and magnetic data. Its latest application, in the Airborne and Metalliferous Geophysics Section, will be the estimation of parameters from measurements made while using the Complex Resistivity Method.

This report describes the methods of parameter estimation using least squares and provides prospective users with routines that perform the processes outlined. The computer routines were written with a view to minimising storage requirements and maximising efficiency and accuracy. It is hoped that a user will now have available to him a set of routines that can be readily applied to the user's problem without his having to develop his own software, an unnecessary duplication. The method of non-linear parameter estimation, the Gauss-Newton method with a one-dimensional search, is not as powerful as many of the more recently developed inversion packages, but it is far less complex and requires far less computer storage and time; it should prove sufficient for many of the applications that arise in BMR.

1. Introduction

1.1 The Estimation of parameters using least squares

One of the most extensively used techniques in numerical methods is the estimation of parameters by the principle of least squares. The technique is employed to derive information about the functional relationship between x and y, assuming such a relation exists, from a set of data pairs $(x_i, y_i), (i=1,n)$.

Data are commonly obtained by observing physical phenomena and quantitatively measuring variables of interest. These data so obtained may contain a variety of errors. Some of these errors may be errors of observation, errors of measurement, and errors of recording. Additional errors may also be introduced in transmission or in conversion.

The estimation of parameters by least squares causes a smoothing of a given set of data and eliminates, to some degree, errors in observation, measurement, recording, transmission, and conversion, as well as other types of random error which may have been introduced into the data. Bias errors (i.e., a fixed offset Δy in each y_i) will not be eliminated by this technique. This feature of smoothing the data is one of the most important features of the principle of least squares, and distinguishes it from interpolation (an interpolating polynomial exactly fits all of the data points used so that any errors in the data will be retained).

1.2 Categories of least-squares techniques

There are two distinct but related categories of techniques based on the principle of least squares:

(1) the estimation of linear parameters, and
(2) the estimation of non-linear parameters.

Category 1 is the well known problem of "fitting" a function of the form

$$F(x) = \beta_1 f_1(x) + \beta_2 f_2(x) + \ldots\ldots\ldots\ldots + \beta_m f_m(x) \qquad (1.2.1)$$

to a set of data pairs $(x_i, y_i), (i=1,n)$. In this problem, the unknown parameters enter into the functional relation linearly. That is, the approximating function f(x) is linear in the parameters $\beta_1, \beta_2, \ldots \beta_m$

Category 2 is the problem of "fitting" an explicit function again to a set of data pairs $(x_i, y_i), (i=1,n)$. In this problem the unknown parameters enter into the functional relation non-linearly. That is, F(x) is non-linear in the unknown parameters.

1.3 Derivation of the least squares result

Assuming that a functional relation exists between x and y for those data measured $(x_i, y_i), (i=1,n)$, then the "m" parameters that best approximate the given set of data in a least squares sense will be those which minimise the differences between the measured (y) and calculated (y^o) values at each point of measurement. The measure of these differences is the sum of the squared residuals, J, given by:

$$J = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i^o - y_i)^2$$

where y_i is the measured value at x_i,

y_i^o is the value of the function F(x) at $x = x_i$

From equation (1.2.1),

$$J = \sum_{i=1}^{n} (\beta_1 f_1(x_i) + \beta_2 f_2(x_i) + \ldots + \beta_m f_m(x_i) - y_i)^2, (i=1,n) \qquad (1.3.1)$$

It is apparent then that the sum J is a function of the independent parameters $\beta_1, \beta_2, \ldots \beta_m$. We can therefore consider the parameters β_k as independent variables and write $J(\beta_1, \beta_2, \ldots \beta_m)$ to indicate that J is a function of these variables. Now, as the minimum of a function of several variables occurs at a point at which all partial derivatives of the function are simultaneously zero, then a minimum is obtained when the "m" partials of $J(\beta_1, \beta_2, \ldots \beta_m)$ with respect to the parameters $\beta_1, \beta_2, \ldots \beta_m$ simultaneously vanish; i.e. when

$$\frac{\partial J}{\partial \beta_k} = 2 \sum_{i=1}^{n} (y_i^o - y_i) \frac{\partial y_i^o}{\partial \beta_k} = 0, \ (k=1,m) \qquad (1.3.2.)$$

If the model is correct, the resultant "m" equations (the Normal equations) will be independent. Using matrix equations, Equation (1.2.1) may be rewritten

$$\underset{\sim}{\chi}^{o} = A\underset{\sim}{\beta}$$

and

$$J = \sum_{i=1}^{n} e_i^2 = \underset{\sim}{e}^T \underset{\sim}{e}$$
$$= (\underset{\sim}{\chi}^{o} - \underset{\sim}{\chi})^T . (\underset{\sim}{\chi}^{o} - \underset{\sim}{\chi})$$
$$= (A\underset{\sim}{\beta} - \underset{\sim}{\chi})^T . (A\underset{\sim}{\beta} - \underset{\sim}{\chi}) \qquad (1.3.3)$$

hence,

$$\frac{\partial J}{\partial \beta} = 2A^T(A\underset{\sim}{\beta} - \underset{\sim}{\chi}) = \underset{\sim}{0}$$ for J to be a minimum

i.e.

$$A^T A\underset{\sim}{\beta} - A^T\underset{\sim}{\chi} = \underset{\sim}{0}$$

and therefore

$$\underset{\sim}{\beta} = (A^T A)^{-1} A^T \underset{\sim}{\chi} \qquad (1.3.4)$$

Now

$$A = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ \vdots & \vdots & & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{bmatrix}$$

and, hence,

$$A^T = \begin{bmatrix} f_1(x_1) & f_1(x_2) & \cdots & f_1(x_n) \\ f_2(x_1) & f_2(x_2) & \cdots & f_2(x_n) \\ \vdots & \vdots & & \vdots \\ f_m(x_1) & f_m(x_2) & \cdots & f_m(x_n) \end{bmatrix}$$

Therefore, equation (1.3.4) becomes

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_m \end{bmatrix} = \left[\begin{bmatrix} f_1(x_1) & f_1(x_2) & & f_1(x_n) \\ \vdots & \vdots & & \vdots \\ f_m(x_1) & f_m(x_2) & \cdots & f_m(x_n) \end{bmatrix} \begin{bmatrix} f_1(x_1) & f_2(x_1) & & f_m(x_1) \\ \vdots & \vdots & & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{bmatrix} \right]^{-1}$$

$$\begin{bmatrix} f_1(x_1) & f_1(x_2) & \cdots & f_1(x_n) \\ \vdots & \vdots & & \vdots \\ f_m(x_1) & f_m(x_2) & \cdots & f_m(x_n) \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_n \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{n} (f_1(x_i))^2 & \sum_{i=1}^{n} f_1(x_i) f_2(x_i) & \cdots & \sum_{i=1}^{n} f_1(x_i) f_m(x_i) \\ \sum_{i=1}^{n} f_2(x_i) f_1(x_i) & \sum_{i=1}^{n} (f_2(x_i))^2 & \cdots & \sum_{i=1}^{n} f_2(x_i) f_m(x_i) \\ \vdots & \vdots & & \vdots \\ \sum_{i=1}^{n} f_m(x_i) f_1(x_i) & \sum_{i=1}^{n} f_m(x_i) f_2(x_i) & \cdots & \sum_{i=1}^{n} (f_m(x_i))^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} \gamma_i f_1(x_i) \\ \sum_{i=1}^{n} \gamma_i f_2(x_i) \\ \vdots \\ \sum_{i=1}^{n} \gamma_i f_m(x_i) \end{bmatrix} \tag{1.3.5}$$

i.e., $\underset{\sim}{\beta} = C^{-1} \underset{\sim}{b}$

2. Estimation of linear parameters using least squares.

2.1 Computer algorithm.

Using equation (1.3.2) the following algorithm was compiled:

1. Form the matrix A where
   $a_{ij} = f_j(x_i), (i=1,n), (j=1,m)$
2. Form the new matrix $C = A^T A$ where
   $$c_{ij} = \sum_{k=1}^{n} a_{ki} \cdot a_{kj}, (j=1,m), (i=1,m)$$
3. Form the vector $\underset{\sim}{b} = A^T \underset{\sim}{\chi}$ where
   $$b_i = \sum_{i=1}^{n} a_{ki} \cdot \gamma_k \ , (i=1,m)$$
4. Solve the Normal equations
   $$C\underset{\sim}{\beta} = \underset{\sim}{b}$$
   by inverting C and forming $C^{-1}\underset{\sim}{b}$

On inspection, however, several of these steps can be combined.

2.2. Modified algorithm

As $C = A^T A$ is symmetrical (because $(AA^T)^T = A^T A$ only one of the triangular matrices (upper or lower) of C need be evaluated, the other being given by

$$C_{ij} = C_{ji}$$

To reduce storage requirements, two steps may be taken also:

i) C may be evaluated directly from (x,y) without forming A, and

ii) C need be stored only as a vector by compressing one of th triangular matrices of C by columns, usually the upper matrix, as shown below:

$$\begin{array}{|ccc|} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{array}$$

$V_1 = C_{11}$ $V_2 = C_{12}$ $V_4 = C_{13}$
$V_3 = C_{22}$ $V_5 = C_{23}$
$V_6 = C_{33}$

The saving in storage requirements is given by

$$m*n + (m*n - \frac{m*(m+1)}{2}) = \frac{1}{2} m (m-1+2n)$$

The new algorithm is:

1. Form $C = A^T A$ directly where

$$c_{ij} = \sum_{k=1}^{n} f_i(x_k).f(x_k), (i=1,m), (j=i,m)$$

2. Form the vector $\underset{\sim}{b}$ directly where

$$b_i = \sum_{k=1}^{n} f_i(x_k).y_k, (i=1,m)$$

3. Store C in the vector (one dimensional array) CMTRX where the subscript for each element is given by

$$\text{CMTRX (INDEX)} = C_{ij}$$

where $\text{INDEX} = i + (j^2 - j)/2$

4. Solve $C\underset{\sim}{\beta} = \underset{\sim}{b}$ where C is a symmetric matrix stored columnwise compressed in an array (CMTRX). The IBM SSP routine GELS will solve a system of linear equations where the matrix is columnwise compressed, using Gaussian elimination with pivoting in the main diagonal. Routine GELS outputs an error flag "IER" and the values it can have are:

IER = -1 - C is either singular or roundoff has induced singularity
= 0 - no loss of significance
= K - possible loss of significance

For a well-scaled matrix and appropriate tolerance, ε, IER=K may be interpreted as C having rank K.

A flowchart for the modified algorithm is shown in Fig. 1.

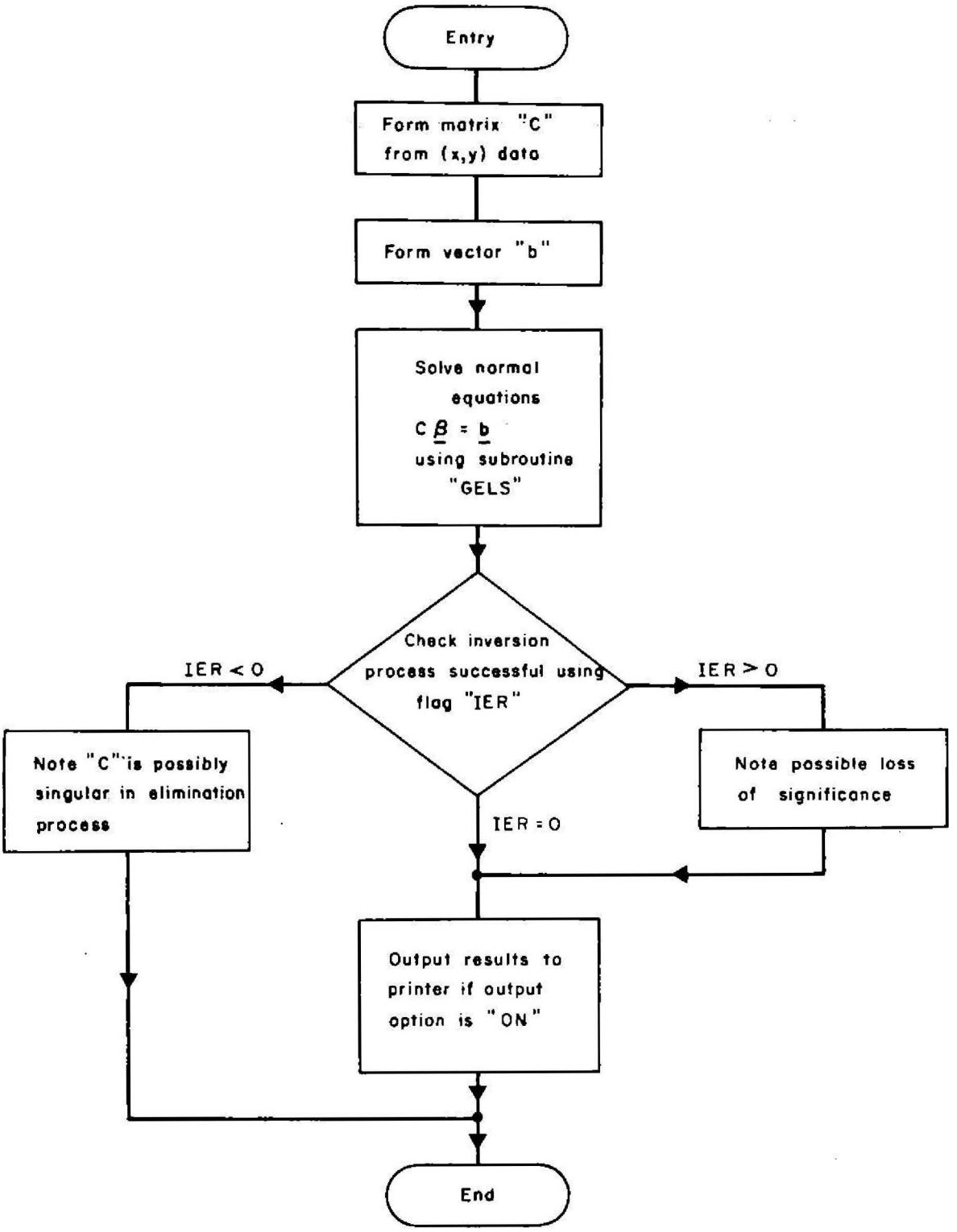

Fig.1 Flowchart of subroutine LLSQF

ESTIMATION OF LINEAR PARAMETERS BY LEAST SQUARES

NUMBER OF PARAMETERS : 2

NUMBER OF DATA PAIRS : 5

I	X(I)	Y(I)
1	0.0000	1.0000
2	1.0000	3.8500
3	2.0000	6.5000
4	3.0000	9.3500
5	4.0000	12.0500

THE VECTOR B AND THE UPPER TRIANGULAR MATRIX OF C TRANSPOSED :

32.7500	5.0000	
93.1000	10.0000	30.0000

FOR THE SOLUTIONS LISTED BELOW :

THE SUM OF THE RESIDUALS SQUARED WAS: 0.0090

THE R.M.S. ERROR WAS : 0.0424

I	BETA (I)
1	1.0300
2	2.7600

TABLE 1: Example output from LLSQF (LULP $\neq$ 0)

2.3 Subroutine LLSQF

A subroutine has been written in FORTRAN IV to perform the operations outlined in the algorithm given in section (2.2) of this report. A listing of the subroutine is given in Appendix 1. A listing of the IBM Scientific Subroutine Package Subroutine "GELS" is given in Appendix 2.

Subroutine LLSQF can be called by a user's program, and LLSQF will return estimates of the linear parameters for a user specified set of (x,y) data. The calling request should be:

```
CALL LLSQF (X,Y,N,B,M,IER,LULP)
```

where X,Y are the (x,y) data pairs,
N is the number of data pairs,
B is the estimated parameters returned,
M is the number of parameters required,
IER is the error code (see section 2.2), and
LULP is the logical unit of the line printer.
(if LULP = 0, there will be no printer output)

2.4 Example

The following set of data were input to LLSQF (N=5):

X	0	1	2	3	4
y = F(x)	1.00	3.85	6.50	9.35	12.05

Two parameters (M = 2) were to be estimated. The results are shown in Table 1.

3. Estimation of non-linear parameters using least squares - the Gause-Newton Method

3.1 Introduction

In many applied problems, the form of a function representing a phenomenon is known but the values of the parameters are to be determined. For example, a particular phenomenon might be represented by an exponential, logarithmic, trigonometric, or other function. If a set of observations provides empirical values y_i of the function

$$y_i^o = f(x_i, \beta_1, \beta_2, \ldots \beta_m), (i=1,n), (n > m) \qquad (3.1.1)$$

then the parameters $\beta_k, (k=1,m)$ can often be determined from a set of estimates β_k^o by differential-correction techniques based on least squares.

3.2 Application of the least squares result to the non-linear problem

Assume for a particular phenomenon there exists an explicit function (see equation (3.1.1)) relating variables x and y, where $\beta_1, \beta_2, \ldots \beta_m$ are unknown independent parameters that enter non-linearly into the functional expression. Observation of this phenomenon has provided a set of data pairs $(x_i, y_i), (i=1,n)$ at distinct, discrete values (x_i) of the independent variable x. The recorded values y_i may contain errors of observation, measurement, recording, transmission, conversion, and so forth.

If the values of the parameters β_k were known it would be possible to evaluate $f(x, \beta_1, \beta_2, \ldots \beta_m)$ for each x_i to obtain a set of "true" residuals :

$$e_i = f(x_i, \beta_1, \beta_2, \ldots \beta_m) - y_i, (i=1,n)$$

A "true" residual then would represent the difference between the actual function value at x_i and the empirical (recorded) value y_i . These "true" residuals cannot be calculated because the actual values of the parameter β_k are unknown.

However, if estimates β_k^o of the parameters can be obtained by some means (the "initial" estimates), then "computed" residuals

$$e_i^o = f(x_i, \beta_1, \beta_2 \ldots \beta_m^o) - y_i \ , \ (i=1,n) \tag{3.1.3}$$

can be calculated. The problem then, is to obtain improved values of the parameters β_k using the data pairs (x_i, y_i), , the estimates β_k^o, and the "computed" residuals e_i^o . This can be accomplished by a differential-correction technique based on least squares, provided that the estimates are sufficiently close to the actual values of the parameters β_k to lead to convergence of the method to the correct solution.

This differential-correction technique can be derived by first expanding the function about $(\beta_1, \beta_2, \ldots \beta_m)$ using a linear Taylor-series expansion of the form

$$f(x, \beta_1, \beta_2, \ldots \beta_m) \approx f(x, \beta_1^o, \beta_2^o, \ldots \beta_m^o) + \frac{\partial f}{\partial \beta_1}(\beta_1 - \beta_1^o) + \ldots + \frac{\partial f}{\partial \beta_m}(\beta_m - \beta_m^o) \tag{3.1.4}$$

so that a relation between e_i and e_i^o can be obtained. This relation can be found by evaluating (3.1.4) at each value of x_i and subtracting y_i from both sides of the equation. Using the definitions

$$\delta\beta_k = \beta_k - \beta_k^o \ , \ (k=1,m)$$

and

$$\frac{\partial f_i}{\partial \beta_k} = \frac{\partial f}{\partial \beta_k}\Big|_{x=x_i, \beta_k=\beta_k^o},$$

we can write the result given in equation (3.1.4) as

$$f(x_i, \beta_1, \beta_2 \ldots \beta_m) - y_i \approx f(x_i, \beta_1^o, \beta_2^o \ldots \beta_m^o) - y_i + \frac{\partial f_i}{\partial \beta_1}\delta\beta_1 + \ldots + \frac{\partial f_i}{\partial \beta_m}\delta\beta_m, \ (i=1,n) \tag{3.1.5}$$

or, in matrix form,

$$\underset{\sim}{f}(\underset{\sim}{x}, \underset{\sim}{\beta}) = \underset{\sim}{f}(\underset{\sim}{x}, \underset{\sim}{\beta}^o) + \frac{\partial \underset{\sim}{f}}{\partial \underset{\sim}{\beta}}\Big|_{\underset{\sim}{\beta}=\underset{\sim}{\beta}^o}$$

$$= \underset{\sim}{f}(\underset{\sim}{x}, \underset{\sim}{\beta}^o) + A\delta\underset{\sim}{\beta}$$

and $\frac{\partial \underset{\sim}{f}}{\partial \underset{\sim}{\beta}}\Big|_{\underset{\sim}{\beta}=\underset{\sim}{\beta}^o}$ is an n x m matric (A)

where $a_{ij} = \frac{\partial f(x_i, \beta_1; \beta_2, \ldots \beta_m)}{\partial \beta_j}\Big|_{\underset{\sim}{\beta}=\underset{\sim}{\beta}^o}$

Now, as with the linear case, to minimise the errors, we minimise the measure J defined:

$$
\begin{aligned}
J &= \sum_{i=1}^{n} e_i^2 \\
&= \underset{\sim}{e}^T \underset{\sim}{e} \\
&= \left[\underset{\sim}{y}^o - \underset{\sim}{y}\right]^T \left[\underset{\sim}{y}^o - \underset{\sim}{y}\right]
\end{aligned}
\tag{3.1.7}
$$

In this case, using equation (3.1.6),

$$
\begin{aligned}
J &= \left[\underset{\sim}{f}(\underset{\sim}{x},\underset{\sim}{\beta}) - \underset{\sim}{y}\right]^T \left[\underset{\sim}{f}(\underset{\sim}{x},\underset{\sim}{\beta}) - \underset{\sim}{y}\right] \\
&= \left[A\delta\underset{\sim}{\beta} + \underset{\sim}{f}(\underset{\sim}{x},\underset{\sim}{\beta}^o) - \underset{\sim}{y}\right]^T \left[A\delta\underset{\sim}{\beta} + \underset{\sim}{f}(\underset{\sim}{x},\underset{\sim}{\beta}^o) - \underset{\sim}{y}\right]
\end{aligned}
\tag{3.1.8}
$$

and

$$\underset{\sim}{e}^o = \underset{\sim}{f}(\underset{\sim}{x},\underset{\sim}{\beta}^o) - \underset{\sim}{y}$$

then

$$J = \left[A\delta\underset{\sim}{\beta} + \underset{\sim}{e}^o\right]^T \left[A\delta\underset{\sim}{\beta} + \underset{\sim}{e}^o\right] \tag{3.1.9}$$

This can now be differentiated with respect to $\delta\underset{\sim}{\beta}$ and a minimum found,

$$
\begin{aligned}
\frac{\partial J}{\partial \delta\underset{\sim}{\beta}} &= 2A^T(A\delta\underset{\sim}{\beta} + \underset{\sim}{e}^o) \\
&= \underset{\sim}{0} \qquad \text{for a minimum.} \\
\therefore\ \delta\underset{\sim}{\beta} &= -(A^TA)^{-1}A^T\underset{\sim}{e}^o
\end{aligned}
\tag{3.1.10}
$$

Hence, we have an iterative method for improving $\underset{\sim}{\beta}$ by:

$$\underset{\sim}{\beta} = \underset{\sim}{\beta}^o + \delta\underset{\sim}{\beta}$$

As $\delta\underset{\sim}{\beta}$ is effectively a first-order approximation for the change in $\underset{\sim}{\beta}$ (since the Taylor series expansion was truncated), it is normally necessary to repeat the procedure of refinement several times until J reaches an acceptable level. This process is iterative:

$$\underset{\sim}{\beta}^{k+1} = \underset{\sim}{\beta}^k + \delta\underset{\sim}{\beta}^k \tag{3.1.11}$$

where

$$\delta\underset{\sim}{\beta}^k = -(A^TA)^{-1}A^T\underset{\sim}{e}_k^o$$

and k is the iteration number (k = 0,1,2...) until for all i . Convergence may be slow, particularly if the initial estimates for $\underset{\sim}{\beta}$ were poor. The rate of convergence may be improved by employing a linear search, namely

$$\underset{\sim}{\beta}^{k+1} = \underset{\sim}{\beta}^{k} + d \, . \, \delta \underset{\sim}{\beta}$$

where "d" is a scalar constant given by performing a one-dimensional search:

$$J(\underset{\sim}{\beta}^{k+1}) = \min_{d} J(\underset{\sim}{\beta}^{k} + d \, . \, \delta \underset{\sim}{\beta}^{k}) \qquad (3.1.12)$$

This will be included in the computer algorithm described next.

3.3 Computer algorithm

1. Form the matrix A where

$$a_{ij} = \frac{\partial f(x_i, \beta_1^{l}, \beta_2^{l}, \ldots \beta_m^{l})}{\partial \beta_j} \quad (i=1,m), (j=1,m) \quad (l^{th} \text{ iteration})$$

2. Form the vector $\underset{\sim}{e}^{l}$ where

$$e_i^{l} = f(x_i, \beta_1^{l}, \beta_2^{l}, \ldots \beta_m^{l}) - y_i \; , \; (i=1,n)$$

3. Form the new matrix $C = A^T A$ where

$$c_{ij} = \sum_{k=1}^{n} a_{ki} \cdot a_{kj} \, , \quad (j=1,m), (i=1,m)$$

4. Form the vector $\underset{\sim}{b} = A^T \underset{\sim}{e}^{l}$ where

$$b_i = \sum_{i=1}^{n} a_{ki} \, e_k^{l}$$

5. Solve the Normal Equations

$$C \delta \underset{\sim}{\beta} = - \underset{\sim}{b}$$

6. Perform the one-dimensional search

$$J(\beta^{l+1}) = \min_{d} J(\underset{\sim}{\beta}^{l} + d \, . \, \delta \underset{\sim}{\beta}^{l})$$

7. Compute $\underset{\sim}{\beta}^{l+1}$, J and RMSERR

8. Test for convergence; if not converged, go back to step 1.

3.4 Modified algorithm

As with the linear result, the matrix C and the vector $\underset{\sim}{b}$ can be formed directly without the matrix A as can the vector e_k^l. Also, C is stored columnwise compressed in an array.

Also, as the matrix C is often poorly scaled, resulting in loss of significance, it is scaled before the inversion process and $\delta\underset{\sim}{b}$ is rescaled afterwards.

The new algorithm steps are:

1. Form the matrix C (without A) where

$$c_{ij} = \sum_{k=1}^{n} \frac{\partial f(x_k,\underset{\sim}{\beta})}{\partial \beta_i} \cdot \frac{\partial f(x_k,\underset{\sim}{\beta})}{\partial \beta_j}, \quad (i=1,m),(j=i,m)$$

and store in CMTRX (INDEX) = c_{ij}

INDEX = $i + (j^2 - j)/2$

2. Form the vector $\underset{\sim}{b} = A^T\underset{\sim}{e}^l$ where

$$b_i = \sum_{i=1}^{n} \frac{\partial f(x_k,\underset{\sim}{\beta})}{\partial \beta_i} \cdot e_k^l \qquad (l^{th} \text{ iteration})$$

and

$$e_k^l = f(x_k,\underset{\sim}{\beta}^l) - y_k$$

3. Scale C and $\underset{\sim}{b}$
4. Solve the Normal Equations $C\delta\underset{\sim}{\beta} = -\underset{\sim}{b}$
5. Rescale the resultant solution $\delta\underset{\sim}{\beta}$
6. Perform the one-dimensional search where

$$J(\underset{\sim}{\beta}^{l+1}) = \min_d (\underset{\sim}{\beta}^l + d \cdot \delta\underset{\sim}{\beta}^l)$$

7. Compute $\underset{\sim}{\beta}^{l+1}$, J and RMSERR
8. Test for convergence; if not converted go back to step 1.

Flowcharts for the algorithm and step 6 are given over. (Figures 2,3).

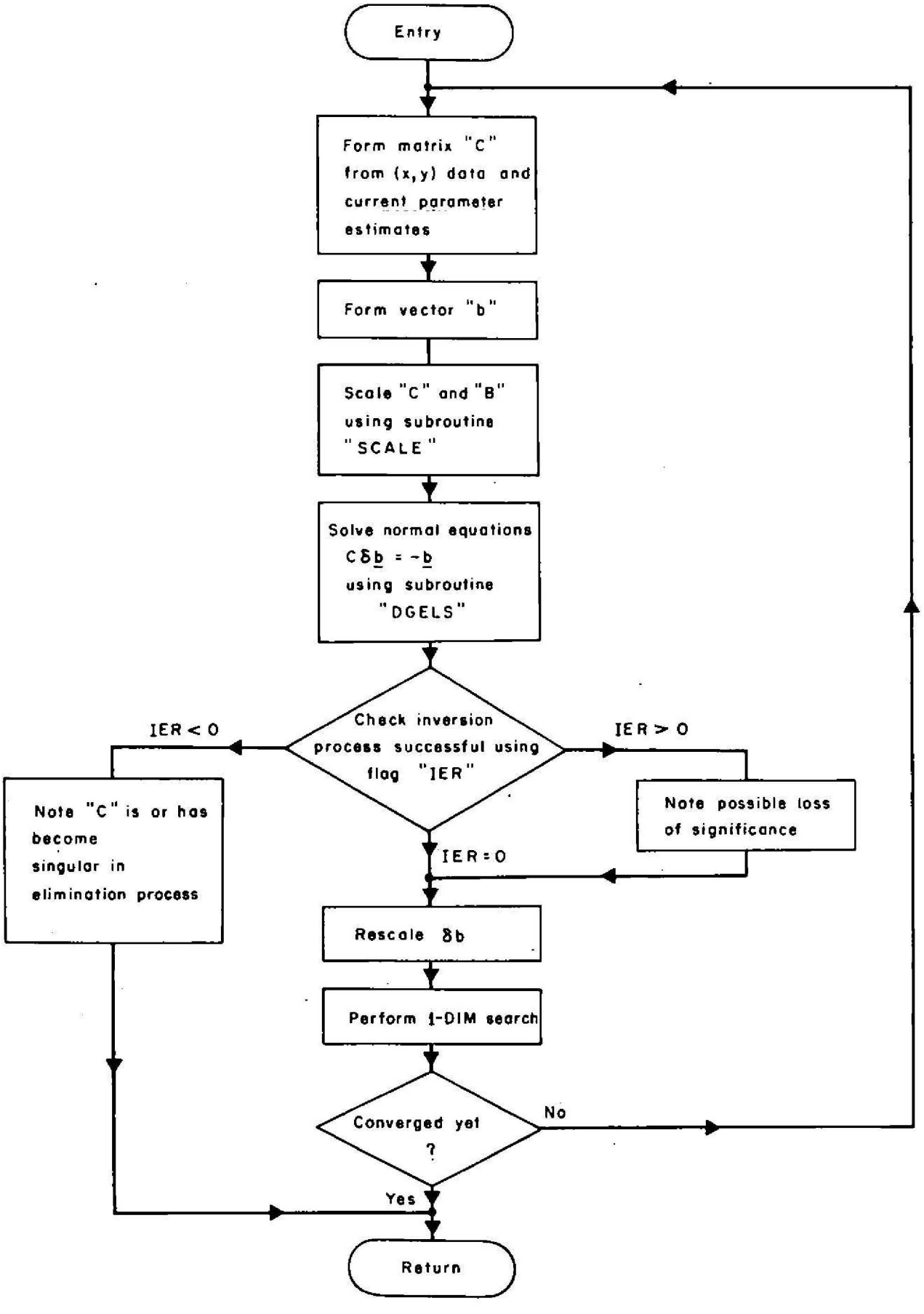

Fig.2 Flowchart of subroutine NLSQF

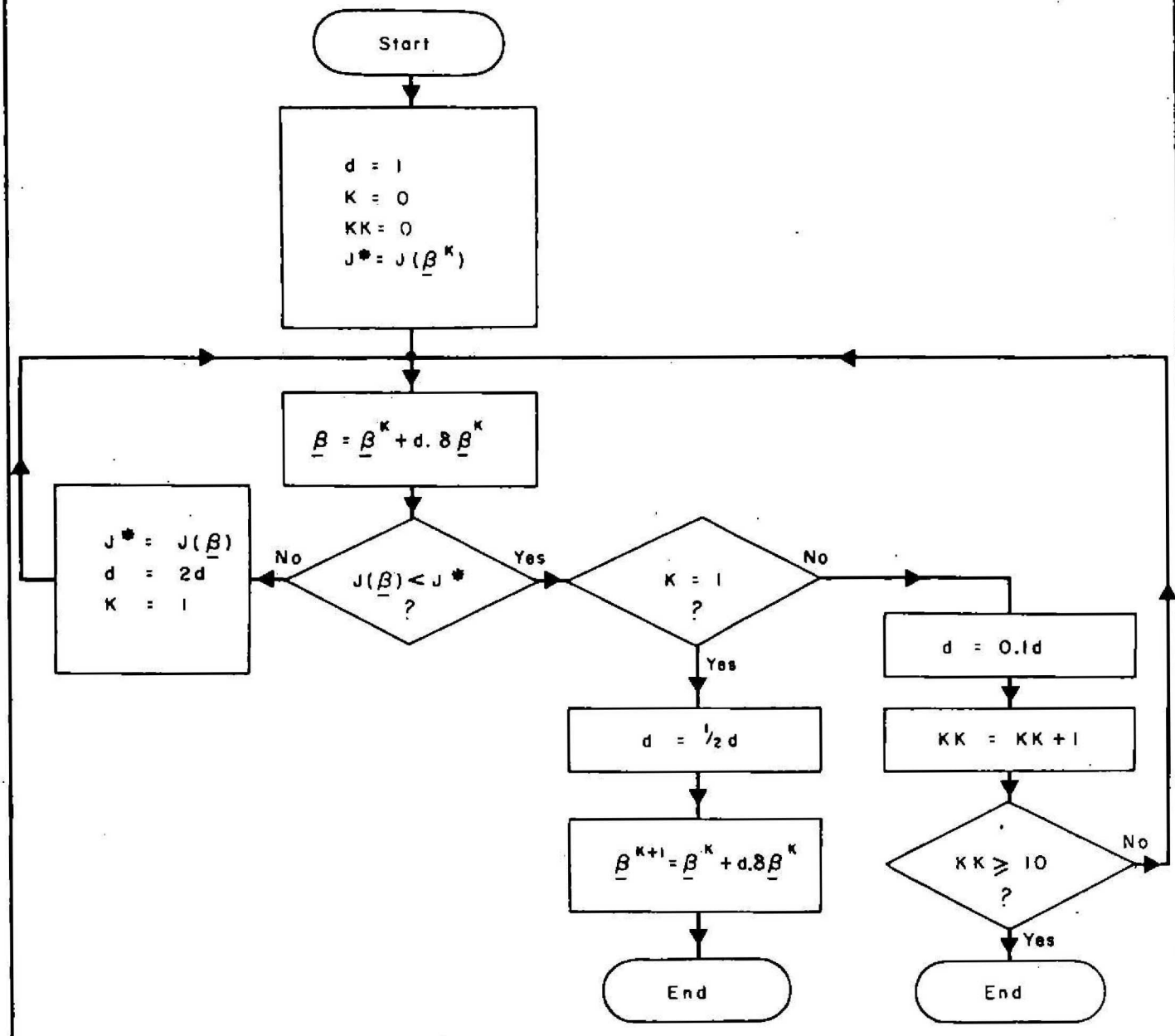

Fig. 3 Flowchart of One – dimensional search

3.5 Subroutine NLSQF

A subroutine has been written in Fortran IV to perform the operations outlined in the algorithm given in section (3.4). A listing of the subroutine is given in Appendix 3. A listing of the double-precision version of GELS, namely DGELS, is given in Appendix 4.

Subroutine NLSQF can be called by a user's program and will return estimates of the non-linear parameters for the user-specified set of (x,y) data and starting solution. The calling request should be

```
CALL NLSQF (X,Y,N,B,M,CGNCE,MAXIT,IER,LULP)
```

where	X,Y	are the (x,y) data pairs (double precision),
	N	is the number of data pairs;
	B	is the starting estimates on entry, the final estimates on return (double precision),
	M	is the number of parameters,
	*CGNCE	is the convergence factor on entry, the RMS error on return,
	MAXIT	is the maximum number of iterations to be performed on entry, the actual number of iterations on return,
	IER	is the error code (see section 2.2), and
	**LULP	is the logical unit number of the line printer.

* The process of estimation of parameters is said to have converged to the solution when the RMS error (= SQRT (J/N)), the average residual, is less than some user-specified value (CGNCE); if the maximum number of iterations is equalled and the RMS error is still greater than CGNCE, then the process stops. The user must consider this result as the solution may not have converged.

**If LULP is not equal to zero, the normal equations, error vectors etc will be output for each iteration on logical unit = LULP.

***X, Y and B are double precision variables.

3.6 Example

The following set of data was input to NLSQF (N=40)

I	X(I)	Y(I)	I	X(I)	Y(I)
1	0.0	0.00000	21	20.0	0.26773
2	1.0	0.19466	22	21.0	0.27716
3	2.0	0.37928	23	22.0	0.28640
4	3.0	0.55472	24	22.0	0.29544
5	4.0	0.72178	25	24.0	0.30430
6	5.0	0.88117	26	25.0	0.31299
7	6.0	0.10335	27	26.0	0.32152
8	1.0	0.11794	28	27.0	0.32990
9	8.0	0.13195	29	28.0	0.33813
10	9.0	0.14541	30	29.0	0.34623
11	10.0	0.15837	31	30.0	0.35420
12	11.0	0.17087	32	31.0	0.36204
13	12.0	0.18296	33	22.0	0.26977
14	13.0	0.19465	34	33.0	0.37738
15	14.0	0.20598	35	34.0	0.38489
16	15.0	0.21697	36	35.0	0.39229
17	16.0	0.22766	37	36.0	0.39959
18	17.0	0.23806	38	77.0	0.40679
19	18.0	0.24819	39	38.0	0.41390
20	19.0	0.25808	40	39.0	0.42091

The explicit function to be fitted was

$$F(\underset{\sim}{x},\underset{\sim}{\beta}) = \beta_1(1-e^{\beta_2 x}) + \beta_3(1-e^{\beta_4 x})$$

and the partial derivatives were

$$\frac{\delta F}{\delta \beta_1} = 1 - e^{\beta_2 x} \qquad \frac{\delta F}{\delta \beta_2} = -\beta_1 x e^{\beta_2 x}$$

$$\frac{\delta F}{\delta \beta_3} = 1 - e^{\beta_4 x} \qquad \frac{\delta F}{\delta \beta_4} = -\beta_3 x e^{\beta_4 x}$$

The convergence factor was 0.00001 and the maximum number of iteration was set to 10. The process converged in 4 iterations with RMS error = 0.3614E-06 and

$$\underset{\sim}{\beta} = (1{,}000, -0.0100, 0.1000, -0.1000)$$

These parameters were actually used to generate the (x,y) data pairs used in input. The initial estimates for $\underset{\sim}{\beta}$ were (1.1, -0.015, 0.08, -0.09)

3.7 Functions F and DFDB

In the example discussed in section (3.6), the functional relation being used was

$$F(\underset{\sim}{x},\underset{\sim}{\beta}) = \beta_1(1-e^{\beta_2 x}) + \beta_3(1-e^{\beta_4 x})$$

This function was coded and included as a function subprogram (F). It is unlikely that this functional relation is the one the user will want to use. Therefore, the user must provide his own function "F". The following statement must be included in the function subprogram:

```
DOUBLE   PRECISION   FUNCTION   F (B, X )
```

where B is a dummy variable array containing the current parameter estimates, and
X is the value of x at which $F(\underset{\sim}{x},\underset{\sim}{\beta})$ is to be evaluated

The remaining code must result in the variable "F" being defined prior to the RETURN statement.

Similarly, there must also be a function subprogram called DFDB which allows the partial derivatives of F with respect to each of the parameters to be evaluated at $x = x_i, (i = 1,n)$. The partial derivatives for the example (section 3.6), shown previously, were coded in function subprogram called DFDB; the user must provide a similar function with the following statement as the first statement in the routine:

```
DOUBLE   PRECISION   FUNCTION   DFDB   (I,B,X)
```

where I indicates that the partial derivative to be calculated is with respect to the "i"th parameter (i = 1,m),
B is a dummy variable array containing the current parameter estimates, and
X is the value of x at which the partial derivative is to be evaluated.

Listings of F and DFDB for the example problem described in section (3.6) are given in Appendix 5.

3.8 Function DFDB - general case

Often it is difficult to derive analytical expressions for the partial derivatives of the explicit function F with respect to the parameters being estimated. In such cases, a generalised function that evaluates the partial derivatives using a numerical differentiation technique might be used. The user must, however, be careful if he elects to use such a method, as those numerical techniques available almost always have limitations that the user must consider.

The author has provided a generalised routine (DFDB) that will evaluate the partial derivatives of the function $F(x,\underset{\sim}{\beta})$ with respect to the parameters being estimated. The computation is based on Richardson & Romberg's extrapolation method as applied to a sequence of central divided differences. This routine is a modified version of subroutine DDCAR, a routine included in the IBM Scientific Subroutine Package. A listing of DFDB can be found in Appendix 6.

The limitation of this particular method is that it assumes that F is eleven-times differentiable in a domain containing a closed, 2-sided symmetric interval of radius H about the current value of the independent variable that F is being differentiated with respect to. The magnitude of H is given by:

$$H = 0{\cdot}01 * \left[\beta_i\right]$$

where $\left[\beta_i\right]$ is the current value of the parameter β_i.

It is also worth noting that although this routine may save some time and effort in the short term (as it is unnecessary to derive the analytical expressions for the partial derivatives), it may prove to be fairly expensive in the long term in actual computer time as compared with evaluating the value of the derivatives using analytical formulae.

With this in mind, another generalised function DFDB was written. This routine simply calculates the slope of the function $F(\underset{\sim}{x},\underset{\sim}{\beta})$ in the vicinity of the current value of the parameter β_i that F is being differentiated with respect to. The partial derivative of $F(\underset{\sim}{x},\underset{\sim}{\beta})$ with respect to β_i $(i=1,m)$ is approximated by

$$\frac{\partial F(\underset{\sim}{x},\underset{\sim}{\beta})}{\partial \beta_i} = \frac{\left[(F(\underset{\sim}{x},\underset{\sim}{\beta}),\beta_i = \beta_i - \Delta\beta_i) - (F(\underset{\sim}{x},\underset{\sim}{\beta}),\beta_i = \beta_i + \Delta\beta_i)\right]}{2\Delta\beta_i}$$

where $\Delta\beta_i = c * \beta_i$

and c is a constant.

The listing of the function DFDB is given in Appendix 7. The value of "c" was set to 0.01 (i.e., 1%). The advantage of this method is that it is very fast and it should suit most applications. This method has, however, one severe limitation: it requires that the second partial derivative of F with respect to each of the parameters in the region $\{\beta_i - \Delta\beta_i, \beta_i + \Delta\beta_i\}$ be close to zero. That is, the first partial derivative is almost constant in this region.

If this condition is not met the value of the partial derivative will most probably be a poor estimate and will hence affect the convergence of the overall method.

Appendix 1

Listing of Subroutine

LLSQF

```
FTN4,L,B
      SUBROUTINE LLSQF(X,Y,N,BBETA,M,IERRA,LULP)
C
C ***************************************************
C *                                                 *
C * ESTIMATION OF LINEAR PARAMETERS BY LEAST SQUARES *
C *                                                 *
C ***************************************************
C
C     PURPOSE
C
C     ESTIMATE A SET OF M INDEPENDENT LINEAR PARAMETERS
C     THAT RESULT IN FITTING A FUNCTION OF THE FORM
C
C         F(x)=b f (x)+b f (x)+......+b f (x)
C               1 1     2 2            m m
C
C     BEST IN A LEAST SQUARES SENSE TO THE N EMPIRICAL
C     (X,Y) DATA PAIRS.
C
C     REFERENCE     MOORE, R.F. (1979)
C                   "ESTIMATION OF PARAMETERS USING LEAST SQUARES"
C                   BMR RECORD (IN PREP.).
C
C     USAGE
C
C          CALL LLSQF(X,Y,N,BBETA,M,IERRA,LULP)
C
C     DESCRIPTION OF PARAMETERS
C
C          X,Y        : (X,Y) DATA PAIRS
C          N          : NUMBER OF (X,Y) DATA PAIRS
C          BBETA      : UNDEFINED ON ENTRY,
C                       CONTAINS SOLUTION ON RETURN (IF IERRA.GE.0)
C          M          : NUMBER OF PARAMETERS TO BE ESTIMATED
C          IERRA      : ERROR CODE FOR SOLUTION OF NORMAL EQUATIONS (SEE GE
C                       (N.B. IERRA SET TO -2 IF M(1 OR )10)
C          LULP       : LOGICAL UNIT OF PRINTER FOR OUTPUT;
C                       IF LULP=0, THERE WILL BE NO PRINT OUTPUT.
C
C
C     DIMENSION CMTRX(L!),...,AUX(L) WHERE L=MAX(M) (CURRENTLY = 10)
C                AND L!=L+(L-1)+(L-2)+...+1
      DIMENSION CMTRX(55),BBETA(1),X(1),Y(1),AUX(10)
      LOCN(II,JJ)=II+(JJ*JJ-JJ)/2
      IERRA=-2
      IF(M.LE.0.OR.M.GT.10) GO TO 999
    1 IF(LULP.NE.0)WRITE(LULP,301) M,N,(I,X(I),Y(I),I=1,N)
  301 FORMAT(1H1////6X"ESTIMATION OF LINEAR PARAMETERS BY LEAST SQUARES"
     *       ///11X"NUMBER OF PARAMETERS :",I5//11X"NUMBER OF DATA PAIRS
     * :",I5///17X"I",12X"X(I)",13X"Y(I)"//(13XI5,2(5XF12.4)))
C     EVALUATE 'C' AND 'B' DIRECTLY FROM DATA (X(I),Y(I),I=1,N).
      BI=0.
      CIJ=0.
      DO 120 I=1,M
```

```
      DO 115 J=I,M
      DO 110 K=1,N
      IF(I-2) 2,3,4
    2 AKI=1.
      GO TO 5
    3 AKI=X(K)
      GO TO 5
    4 AKI=X(K)**(I-1)
    5 IF(J-2) 6,7,8
    6 AKJ=1.
      GO TO 9
    7 AKJ=X(K)
      GO TO 9
    8 AKJ=X(K)**(J-1)
    9 CIJ=CIJ+AKI*AKJ
      IF(J.NE.M) GO TO 110
      BI=BI+AKI*Y(K)
  110 CONTINUE
      INDEX=LOCN(I,J)
      CMTRX(INDEX)=CIJ
      CIJ=0.
  115 CONTINUE
      BBETA(I)=BI
      BI=0.
  120 CONTINUE
      IF(LULP.NE.0)WRITE(LULP,302)
  302 FORMAT(//////6X"THE VECTOR B AND THE UPPER TRIANGULAR MATRIC OF C
     *TRANSPOSED :"///)
      N1=0
      N2=0
      DO 122 I=1,M
      N1=N2+1
      N2=N1+I-1
      IF(LULP.NE.0)WRITE(LULP,303)(BBETA(I),(CMTRX(J),J=N1,N2))
  303 FORMAT(//5XF11.4,5X10F11.4)
  122 CONTINUE
C     SOLVE NORMAL EQUATIONS
      CALL GELS(BBETA,CMTRX,M,1,.1E-05,IERRA,AUX)
      IF(IERRA.NE.0.AND.LULP.NE.0) WRITE(LULP,304) IERRA
  304 FORMAT(///5X"***** NOTE : ",6H'GELS'," ERROR CODE SET TO  ",I3," -
     * BEWARE *****"//)
      IF(IERRA.NE.-1) GO TO 11
      DO 123 I=1,M
  123 BBETA(I)=0.
      GO TO 999
C     PRINT RESULTS
   11 SUMLSQ=0.
      DO 130 I=1,N
      YP=BBETA(1)+BBETA(2)*X(I)
      IF(M-2) 12,12,10
   10 DO 125 J=3,M
      YP=YP+BBETA(J)*X(I)**(J-1)
  125 CONTINUE
   12 RESID=Y(I)-YP
      SUMLSQ=SUMLSQ+RESID*RESID
```

```
0111     130 CONTINUE
0112         RMSERR=SQRT(SUMLSQ/FLOAT(N))
0113         IF(LULP.NE.0)WRITE(LULP,305) SUMLSQ,RMSERR,(I,BBETA(I),I=1,M)
0114     305 FORMAT(//////15X"FOR THE SOLUTIONS LISTED BELOW ,"///
0115        *5X"THE SUM OF THE RESIDUALS SQUARED WAS :",1XE15.8//
0116        *5X"THE R.M.S. ERROR WAS :",17XE15.8///7X"I",9X"BETA(I)"//
0117        *(2XI5,5XE15.8))
0118  C      RETURN
0119     999 RETURN
0120         END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00941 COMMON = 00000

Appendix 2

Listing of Subroutine

GELS

```
      SUBROUTINE GELS(R,A,M,N,EPS,IER,AUX)
C
C     ..................................................
C
C     SOURCE
C       GELS WAS TAKEN FROM THE IBM SYSTEM/360 SCIENTIFIC SUBROUTINE
C       PACKAGE (VSN III) PROGRAMMER'S MANUAL (PROG.NO. 360A-CM-03X).
C
C     PURPOSE
C       TO SOLVE A SYSTEM OF SIMULTANEOUS LINEAR EQUATIONS WITH
C       SYMMETRIC COEFFICIENT MATRIX, UPPER TRIANGULAR PART OF
C       WHICH IS ASSUMED TO BE STORED COLUMNWISE.
C
C     USAGE
C       CALL GELS(R,A,M,N,EPS,IER,AUX)
C
C     DESCRIPTION OF PARAMETERS
C       R      - M BY N RIGHT HAND SIDE MATRIX (DESTROYED)
C                ON RETURN R CONTAINS THE SOLUTION
C       A      - UPPER TRIANGULAR PART OF THE SYMMETRIC
C                M BY M COEFFICIENT MATRIX (DESTROYED).
C       M      - THE NUMBER OF EQUATIONS IN THE SYSTEM
C       N      - THE NUMBER OF RIGHT HAND SIDE VECTORS.
C       EPS    - AN INPUT CONSTANT WHICH IS USED AS RELATIVE
C                TOLERANCE FOR LOSS OF SIGNIFICANCE
C                (SUGGESTED : FROM 10**-6 TO 10**-7)
C       IER    - RESULTING ERROR PARAMETER CODED AS FOLLOWS
C                IER=0  - NO ERROR
C                IER=-1 - NO RESULT BECAUSE OF M LESS THAN 1 OR
C                         PIVOT ELEMENT AT ANY ELIMINATION
C                         STEP EQUALS 0.
C                IER=K  - WARNING DUE TO POSSIBLE LOSS OF
C                         SIGNIFICANCE INDICATED AT ELIMINATION
C                         STEP K+1, WHERE PIVOT ELEMENT WAS LESS
C                         THAN OR EQUAL TO THE INTERNAL TOLERANCE
C                         EPS TIMES ABS MAXIMIUM MAIN DIAGONAL
C                         ELEMENT OF MATRIX A.
C       AUX    - AN AUXILIARY STORAGE ARRAY WITH DIMENSION M-1.
C
C     REMARKS
C       UPPER TRIANGULAR PART OF MATRIX A IS ASSUMED TO BE STORED
C       COLUMNWISE IN M*(M+1)/2 SUCCESSIVE STORAGE LOCATIONS, RIGHT
C       HAND SIDE MATRIX R COLUMNWISE IN N*M SUCCESSIVE STORAGE
C       LOCATIONS. ON RETURN SOLUTON MATRIX R IS STORED COLUMNWISE.
C       THE PROCEDURE GIVES RESULTS IF THE NUMBER OF EQUATIONS M IS
C       GREATER THAN 0 AND PIVOT ELEMENTS AT ALL ELIMINATION STEPS
C       ARE DIFFERENT FROM 0. HOWEVER WARNING IER=K - IF GIVEN -
C       INDICATES POSSIBLE LOSS OF SIGNIFICANCE. IN THE CASE OF A WELL
C       SCALED MATRIX A AND APPROPRIATE TOLERANCE EPS, IER=K MAY BE
C       INTERPRETED THAT MATRIX A HAS THE RANK K. NO WARNING IS
C       GIVEN IN CASE M=1.
C       ERROR IER=-1 DOES NOT NECCESSARILY MEAN THAT
C       MATRIX A IS SINGULAR, AS ONLY MAIN DIAGONAL ELEMENTS
C       ARE USED AS PIVOT ELEMENTS. POSSIBLY SUBROUTINE GELG
C       (WHICH WORKS WITH TOTAL PIVOTING) WOULD FIND A
```

```
C          SOLUTION.
C
C        EXTERNALS
C          ABS
C
C        METHOD
C          SOLUTION IS DONE BY MEANS OF GAUSS-ELIMINATION WITH
C          PIVOTING IN MAIN DIAGONAL, IN ORDER TO PRESERVE
C          SYMMETRY IN REMAINING COEFFICIENT MATRICES.
C
C        ..............................................................
C
         DIMENSION A(1),R(1),AUX(1)
         IF(M)24,24,1
       1 IER=0
         PIV=0.
         L=0
         DO 3 K=1,M
         L=L+K
         TB=ABS(A(L))
         IF(TB-PIV)3,3,2
       2 PIV=TB
         I=L
         J=K
       3 CONTINUE
         TOL=EPS*PIV
         LST=0
         NM=N*M
         LEND=M-1
         DO 18 K=1,M
         IF(PIV)24,24,4
       4 IF(IER)7,5,7
       5 IF(PIV-TOL) 6,6,7
       6 IER=K-1
       7 LT=J-K
         LST=LST+K
         PIVI=1./A(I)
         DO 8 L=K,NM,M
         LL=L+LT
         TB=PIVI*R(LL)
         R(LL)=R(L)
       8 R(L)=TB
         IF(K-M)9,19,19
       9 LR=LST+(LT*(K+J-1))/2
         LL=LR
         L=LST
         DO 14 II=K,LEND
         L=L+II
         LL=LL+1
         IF(L-LR)12,10,11
      10 A(LL)=A(LST)
         TB=A(L)
         GO TO 13
      11 LL=L+LT
      12 TB=A(LL)
```

```
      A(LL)=A(L)
   13 AUX(II)=TB
   14 A(L)=PIVI*TB
      A(LST)=LT
      PIV=0.
      LLST=LST
      LT=0
      DO 18 II=K,LEND
      PIVI=-AUX(II)
      LL=LLST
      LT=LT+1
      DO 15 LLD=II,LEND
      LL=LL+LLD
      L=LL+LT
   15 A(L)=A(L)+PIVI*A(LL)
      LLST=LLST+II
      LR=LLST+LT
      TB=ABS(A(LR))
      IF(TB-PIV)17,17,16
   16 PIV=TB
      I=LR
      J=II+1
   17 DO 18 LR=K,NM,M
      LL=LR+LT
   18 R(LL)=R(LL)+PIVI*R(LR)
   19 IF(LEND)24,23,20
   20 II=M
      DO 22 I=2,M
      LST=LST-II
      II=II-1
      L=A(LST)+.5
      DO 22 J=II,NM,M
      TB=R(J)
      LL=J
      K=LST
      DO 21 LT=II,LEND
      LL=LL+1
      K=K+LT
   21 TB=TB-A(K)*R(LL)
      K=J+L
      R(J)=R(K)
   22 R(K)=TB
   23 RETURN
   24 IER=-1
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00588 COMMON = 00000

Appendix 3

Listing of Subroutine

NLSQF

with JAY and SCALE

```
FTN4,L,B
      SUBROUTINE NLSQF(X,Y,N,BETA,M,CGNCE,MAXIT,IERRA,LULP)
C
C ***********************************************************
C *                                                         *
C *    ESTIMATION OF NON-LINEAR PARAMETERS BY LEAST SQUARES *
C *                                                         *
C *                  THE GAUSS-NEWTON METHOD                *
C *                     WITH 1-DIM SEARCH                   *
C *                                                         *
C ***********************************************************
C
C     PURPOSE
C
C     ESTIMATE BY MEANS OF A DIFFERENTIAL-CORRECTION TECHNIQUE
C     A SET OF M INDEPENDENT PARAMETERS THAT RESULT IN FITTING
C     THE EXPLICIT FUNCTION F (DEFINED IN FUNCTION SUBPROGRAM
C     "F") BEST IN A LEAST SQUARES SENSE TO THE N EMPIRICAL (X,Y)
C     DATA PAIRS. THE PROCESS WILL CONTINUE UNTIL THE RMS ERROR
C     OF THE FIT IS LESS THAN THE CONVERGENCE CRITERIUM "CGNCE" OR
C     THE NUMBER OF ITERATIONS REACHES THE UPPER LIMIT (MAXIT)
C     SET ON ENTRY.
C
C     REFERENCE.     MOORE, R.F.  (1979)
C                    "ESTIMATION OF PARAMETERS USING LEAST SQUARES"
C                    BMR RECORD.(IN PREP.)
C
C     USAGE
C
C          CALL NLSQF(X,Y,N,BETA,M,CGNCE,MAXIT,IERRA,LULP)
C
C     DESCRIPTION OF PARAMETERS
C
C         X,Y          : (X,Y) DATA PAIRS (DOUBLE PRECISION)
C         N            : NUMBER OF (X,Y) DATA PAIRS
C         BETA         : STARTING ESTIMATE OF PARAMETERS ON ENTRY,
C                        FINAL ESTIMATE ON RETURN (DOUBLE PRECISION);
C         M            : NUMBER OF PARAMETERS TO BE ESTIMATED
C         CGNCE        : CONVERGENCE FACTOR (IF RMS ERROR E
C                        (=SQRT(J/N)) < CGNCE, THE PROCESS OF
C                        ESTIMATION IS SAID TO HAVE CONVERGED;
C                        ON RETURN, SET = THE RMS ERROR;
C         MAXIT        : MAXIMUM NUMBER OF ITERATIONS ALLOWABLE
C                        BEFORE TERMINATING;
C                        ON RETURN, SET = THE NUMBER OF ITERATIONS
C                        COMPLETED.
C         IERRA        : ERROR CODE FOR SOLUTION OF NORMAL EQUATIONS
C                        (SEE 'DGELS');
C                        (N.B. IERRA SET TO -2 IF M<1 OR >10)
C         LULP         : LOGICAL UNIT OF PRINTER FOR OUTPUT;
C                        IF LULP=0, THERE WILL BE NO PRINT OUTPUT.
C
C     EXTERNALS REQUIRED (OTHER THAN INTRINSIC FUNCTIONS)
C
C                JAY
```

```
C                    SCALE
C                    DGELS
C                    F    (USER DEFINED)
C                    DFDB(USER DEFINED OR GENERAL VERSION)
C
C      METHOD
C
C      THE METHOD USED IS THE GAUSS-NEWTON METHOD OF ESTIMATION
C      OF NON-LINEAR PARAMETERS BY LEAST SQUARES WITH MATRIX
C      SCALING, MATRIX COMPRESSION (SAVING SPACE) AND A ONE-
C      DIMENSIONAL SEARCH (TO SPEED UP CONVERGENCE). FOR MORE
C      INFORMATION, SEE THE ABOVE REFERENCE.
C
       DOUBLE PRECISION JAY,JAY1,JAY2,D,X,Y,BETA,SCALF,BDUMMY,
      *CMTRX,BDB,AUX,AKI,AKJ,BI,CIJ,DFDB,F
C      DIMENSION CMTRX(L!),BDB(L),AUX(L),SCALF(L),BDUMMY(L) WHERE
C      L=MAX(M) (CURRENTLY = 10) AND L! = L+(L-1)+(L-2)+...+1.
       DIMENSION CMTRX(55),BDB(10),BETA(1),AUX(10),SCALF(10),
      *X(1),Y(1),BDUMMY(10)
       LOCN(I,J)=I+(J*J-J)/2
       EPS=0.1E-14
C *** PARAMETERS INPUT OUTPUT
     8 IF(LULP.NE.0)WRITE(LULP,300)
      *M,N,EPS,CGNCE,MAXIT,(I,X(I),Y(I),I=1,N)
   300 FORMAT(///6X"ESTIMATION ON NON-LINEAR PARAMETERS BY LEAST-SQUARES"
      *//9X"THE GAUSS-NEWTON METHOD WITH MATRIX SCALING AND"
      *" A LINEAR SEARCH"///11X"INPUT :"///11X"NO. OF PARAMETERS : ",I5/
      *//11X"NO. OF DATA PAIRS :",I5//11X"EPSILON :",
      *1XE15.8//11X"CONVERGENCE FACTOR : ",1XE15.8
      *//11X"MAX. NO. OF ITERATIONS FOR CONVERGENCE : ",I5/////
      *10X"I",12X"X(I)",16X"Y(I)"//(6XI5,2(5XD15.8)))
C *** WRITE STARTING SOLUTION
       IF(LULP.NE.0) WRITE(LULP,301) (I,BETA(I),I=1,M)
   301 FORMAT(1H1///11X"STARTING SOLUTION :"///
      *16X"BETA(I),I=1,M : "//(6XI5,2XD15.8))
C *** COMMENCE ITERATIONS
       IERRA=-2
       IF(M.LT.1.OR.M.GT.10) GO TO 12345
       ITER=0
    10 ITER=ITER+1
C *** SET-UP "A TRANSPOSE A" DIRECTLY FROM X,Y DATA AND STORE COLUMNWISE
C      THE SYMMETRIC , POSITIVE DEFINITE MATRIX C IN "CMTRX"
       DO 120 I=1,M
       BI=0.
       DO 115 J=I,M
       CIJ=0.
       DO 110 K=1,N
       AKI=DFDB(I,BETA,X(K))
       AKJ=DFDB(J,BETA,X(K))
       CIJ=CIJ+AKI*AKJ
       IF(J.NE.M) GO TO 110
       BI=BI+AKI*(F(BETA,X(K))-Y(K))
   110 CONTINUE
       INDEX=LOCN(I,J)
       CMTRX(INDEX)=CIJ
```

```
  115 CONTINUE
C *** RIGHTHAND SIDE
      BDB(I)=-BI
  120 CONTINUE
C *** OUTPUT BDB AND CMTRX
      IF(LULP.EQ.0) GO TO 122
      WRITE(LULP,302) ITER
  302 FORMAT(//////6X"THE VECTOR DB AND THE UPPER TRIANGULAR MATRIX OF C
     * TRANSPOSED FOR ITERATION : ",I5/// )
      WRITE(LULP,303)(BDB(I),I=1,M)
  303 FORMAT(//(2X8D14.6))
      N1=0
      N2=0
      DO 121 I=1,M
      N1=N2+1
      N2=N1+I-1
      WRITE(LULP,303)(CMTRX(J),J=N1,N2)
  121 CONTINUE
C *** SCALE C,B
  122 CALL SCALE(CMTRX,M,SCALF,BDB,1,LULP)
C *** SOLVE THE NORMAL EQUATIONS (SOLUTION IN BDB)
      CALL DGELS(BDB,CMTRX,M,1,EPS,IERRA,AUX)
C *** CMTRX - POSSIBLE SINGULARITY OR LOSS OF SIGNIFICANCE IN G.E. ?
      IF(IERRA.NE.0.AND.LULP.NE.0) WRITE(LULP,304) IERRA
  304 FORMAT(///5X"***** NOTE : ",7H"DGELS"," ERROR CODE SET TO ",I3,
     *" - BEWARE *****"//)
      IF(IERRA.NE.-1) GO TO 15
      IF(LULP.NE.0) WRITE(LULP,305)
  305 FORMAT(////5X"ERROR TERMINATION - NORMAL EQNS. ILL-CONDITIONED"//)
      DO 124 I=1,M
      BETA(I)=0.
  124 CONTINUE
      MAXIT=ITER
      CGNCE=0.0
      GO TO 12345
   15 IF(LULP.NE.0) WRITE(LULP,306) ITER,(I,BDB(I),I=1,M)
  306 FORMAT(/////11X"DB(I),I=1,M FOR ITERATION :",I5//(11XI5,2XD16.8/))
C *** UNSCALE BDB
      CALL SCALE(CMTRX,M,SCALF,BDB,2,LULP)
C *** LINEAR SEARCH
      JSTOP=MAXIT/2
      IF(JSTOP.LT.4) JSTOP=4
      IF(JSTOP.GT.10)JSTOP=10
   20 D=1.
      J=0
      K=0
      JAY1=JAY(BETA,X,Y,N)
   25 DO 125 I=1,M
      BDUMMY(I)=BETA(I)+D*BDB(I)
  125 CONTINUE
      JAY2=JAY(BDUMMY,X,Y,N)
      IF(JAY2.GT.JAY1) GO TO 30
      JAY1=JAY2
      D=2.*D
      K=1
```

```
0166          GO TO 25
0167       30 IF(K.NE.1) GO TO 35
0168          D=0.5*D
0169          DO 130 I=1,M
0170          BETA(I)=BETA(I)+D*BDB(I)
0171      130 CONTINUE
0172          GO TO 38
0173       35 D=0.1*D
0174          J=J+1
0175          IF(J.LT.JSTOP) GO TO 25
0176   C      CALCULATE RMS ERROR
0177       38 RMSERR=SQRT(SNGL(JAY1)/FLOAT(N))
0178          IF(LULP.NE.0) WRITE(LULP,3061) D,(I,BETA(I),I=1,M)
0179     3061 FORMAT(//6X"LINEAR SEARCH COMPLETE - D =",D11.4//
0180         *//6X"BETA(I),I=1,M :"///(11XI5,1XD15.8/))
0181   C      CONVERGED YET ?
0182          IF(RMSERR.LT.CGNCE) GO TO 45
0183   C      MAXIMUM ITERATIONS YET ?
0184       40 IF(ITER.GE.MAXIT) GO TO 50
0185          GO TO 10
0186   C *** SOLUTION - CONVERGED
0187       45 MAXIT=ITER
0188          CGNCE=RMSERR
0189          IF(LULP.NE.0) WRITE(LULP,307) JAY1,RMSERR,D,(I,BETA(I),I=1,M)
0190      307 FORMAT(/////11X"AFTER CONVERGENCE ,"///
0191         *6X"J - THE SUM OF THE RESIDUALS SQUARED WAS : ",1XD15.8//
0192         *6X"THE RMS ERROR WAS : ",1XE15.8//
0193         *6X"THE LINEAR FACTOR D WAS : ",1XD15.8//
0194         *6X"BETA(I),I=1,M : "///(15XI5,1XD15.8/))
0195          RETURN
0196   C *** DID NOT CONVERGE IN "MAXIT" ITERATIONS
0197       50 CGNCE=RMSERR
0198          IF(LULP.NE.0) WRITE(LULP,308) JAY1,RMSERR,D,(I,BETA(I),I=1,M)
0199      308 FORMAT(/////11X"AFTER MAXIMUM ITERATIONS ,"///
0200         *6X"J - THE SUM OF THE RESIDUALS SQUARED WAS : ",1XD15.8//
0201         *6X"THE RMS ERROR WAS : ",1XE15.8//
0202         *6X"THE LINEAR FACTOR D WAS : ",1XD15.8//
0203         *6X"BETA(I),I=1,M : "///(15XI5,1XD15.8/))
0204    12345 RETURN
0205          END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 01770 COMMON = 00000

```
0206          DOUBLE PRECISION FUNCTION JAY(B,X,Y,N)
0207   C      FUNCTION JAY CALCULATES THE ERROR MEASURE J (=SUM((F(X)-Y)**2))
0208          DOUBLE PRECISION B,X,Y,EK,F
0209          DIMENSION B(1),X(1),Y(1)
0210          JAY=0.0
0211          DO 100 K=1,N
0212          EK=F(B,X(K))-Y(K)
0213          JAY=JAY+EK*EK
0214      100 CONTINUE
0215          RETURN
0216          END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00066 COMMON = 00000

```
      SUBROUTINE SCALE(C,M,S,B,IJ,LULP)
C
C     SCALE WILL SCALE THE SYMMETRIC MATRIX C STORED
C           COLUMNWISE COMPRESSED USING THE DIAGONAL
C           COEFFICIENTS IN AN ATTEMPT TO REDUCE ILL-
C           CONDITIONING, OVERFLOW AND UNDERFLOW IN
C           THE MATRIX INVERSION PROCEDURE IN 'DGELS'.
C           THE SOLUTION VECTOR 'B' IS RESCALED AFTER
C           'GELS' IS CALLED AND BEFORE 1-D SEARCH.
C
      DOUBLE PRECISION C(1),S(1),B(1)
      LOCN(I,J)=I+(J*J-J)/2
      IF(IJ.NE.1) GO TO 106
C *** STORE SQRT(C(I,I),I=1,M)
      DO 101 I=1,M
      K=LOCN(I,I)
      S(I)=DSQRT(C(K))
  101 CONTINUE
      IF(LULP.NE.0) WRITE(LULP,300) (S(I),I=1,M)
  300 FORMAT(////6X"SCALING : SQRT(C(I,I)),I=1,M :"///(2X8D14.6))
C *** SCALE C
      DO 103 I=1,M
      K=LOCN(I,I)
      C(K)=1.0
      IP1=I+1
      IF(IP1.GT.M) GO TO 103
      DO 102 J=IP1,M
      K=LOCN(I,J)
      C(K)=C(K)/(S(I)*S(J))
  102 CONTINUE
  103 CONTINUE
C *** SCALE B
      DO 104 I=1,M
      B(I)=B(I)/S(I)
  104 CONTINUE
C *** OUTPUT SCALING
      IF(LULP.EQ.0) RETURN
      WRITE(LULP,301)(B(I),I=1,M)
  301 FORMAT(///6X"DB AND C BECOME :"///(2X8D14.6))
      I=0
      J=0
      DO 105 K=1,M
      I=J+1
      J=I+K-1
      WRITE(LULP,302) (C(L),L=I,J)
  302 FORMAT(//(2X8D14.6))
  105 CONTINUE
      RETURN
C *** REVERSE THE SCALING PROCESS FOR B
  106 DO 107 I=1,M
      B(I)=B(I)/S(I)
  107 CONTINUE
      IF(LULP.NE.0) WRITE(LULP,303) (B(I),I=1,M)
  303 FORMAT(//6X"AFTER RESCALING , DB BECOMES :"//(2X8D14.6))
      RETURN
      END
```

Appendix 4

Listing of Subroutine

DGELS

```
      SUBROUTINE DGELS(R,A,M,N,EPS,IER,AUX)
C
C     ..................................................................
C
C     SOURCE
C       DGELS WAS TAKEN FROM THE IBM SYSTEM/360 SCIENTIFIC SUBROUTINE
C       PACKAGE (VSN III) PROGRAMMER'S MANUAL (PROG.NO. 360A-CM-03X).
C
C      PURPOSE
C        TO SOLVE A SYSTEM OF SIMULTANEOUS LINEAR EQUATIONS WITH
C        SYMMETRIC COEFFICIENT MATRIX, UPPER TRIANGULAR PART OF
C        WHICH IS ASSUMED TO BE STORED COLUMNWISE.
C
C      USAGE
C        CALL DGELS(R,A,M,N,EPS,IER,AUX)
C
C      DESCRIPTION OF PARAMETERS
C        R      - M BY N RIGHT HAND SIDE MATRIX (DESTROYED)
C                 ON RETURN R CONTAINS THE SOLUTION
C                 (R IS DOUBLE PRECISION)
C        A      - UPPER TRIANGULAR PART OF THE SYMMETRIC
C                 M BY M COEFFICIENT MATRIX (DESTROYED).
C                 (A IS DOUBLE PRECISION)
C        M      - THE NUMBER OF EQUATIONS IN THE SYSTEM
C        N      - THE NUMBER OF RIGHT HAND SIDE VECTORS.
C        EPS    - AN INPUT CONSTANT WHICH IS USED AS RELATIVE
C                 TOLERANCE FOR LOSS OF SIGNIFICANCE
C                 (SUGGESTED : FROM 10**-14 TO 10**-16)
C        IER    - RESULTING ERROR PARAMETER CODED AS FOLLOWS
C                 IER=0   - NO ERROR
C                 IER=-1  - NO RESULT BECAUSE OF M LESS THAN 1 OR
C                           PIVOT ELEMENT AT ANY ELIMINATION
C                           STEP EQUALS 0.
C                 IER=K   - WARNING DUE TO POSSIBLE LOSS OF
C                           SIGNIFICANCE INDICATED AT ELIMINATION
C                           STEP K+1, WHERE PIVOT ELEMENT WAS LESS
C                           THAN OR EQUAL TO THE INTERNAL TOLERANCE
C                           EPS TIMES ABS MAXIMIUM MAIN DIAGONAL
C                           ELEMENT OF MATRIX A.
C        AUX    - AN AUXILIARY STORAGE ARRAY WITH DIMENSION M-1.
C                 (AUX IS DOUBLE PRECISION)
C
C      REMARKS
C        UPPER TRIANGULAR PART OF MATRIX A IS ASSUMED TO BE STORED
C        COLUMNWISE IN M*(M+1)/2 SUCCESSIVE STORAGE LOCATIONS, RIGHT
C        HAND SIDE MATRIX R COLUMNWISE IN N*M SUCCESSIVE STORAGE
C        LOCATIONS. ON RETURN SOLUTON MATRIX R IS STORED COLUMNWISE.
C        THE PROCEDURE GIVES RESULTS IF THE NUMBER OF EQUATIONS M IS
C        GREATER THAN 0 AND PIVOT ELEMENTS AT ALL ELIMINATION STEPS
C        ARE DIFFERENT FROM 0. HOWEVER WARNING IER=K - IF GIVEN -
C        INDICATES POSSIBLE LOSS OF SIGNIFICANCE. IN THE CASE OF A WELL
C        SCALED MATRIX A AND APPROPRIATE TOLERANCE EPS, IER=K MAY BE
C        INTERPRETED THAT MATRIX A HAS THE RANK K. NO WARNING IS
C        GIVEN IN CASE M=1.
C        ERROR IER=-1 DOES NOT NECCESSARILY MEAN THAT
```

```
C          MATRIX A IS SINGULAR, AS ONLY MAIN DIAGONAL ELEMENTS
C          ARE USED AS PIVOT ELEMENTS. POSSIBLY SUBROUTINE GELG
C          (WHICH WORKS WITH TOTAL PIVOTING) WOULD FIND A
C          SOLUTION.
C
C        EXTERNALS
C          DABS.
C
C        METHOD
C          SOLUTION IS DONE BY MEANS OF GAUSS-ELIMINATION WITH
C          PIVOTING IN MAIN DIAGONAL, IN ORDER TO PRESERVE
C          SYMMETRY IN REMAINING COEFFICIENT MATRICES.
C
C        ..................................................................
C
      DOUBLE PRECISION R,A,AUX,PIV,TB,TOL,PIVI
      DIMENSION A(1),R(1),AUX(1)
      IF(M)24,24,1
    1 IER=0
      PIV=0.
      L=0
      DO 3 K=1,M
      L=L+K
      TB=DABS(A(L))
      IF(TB-PIV)3,3,2
    2 PIV=TB
      I=L
      J=K
    3 CONTINUE
      TOL=EPS*PIV
      LST=0
      NM=N*M
      LEND=M-1
      DO 18 K=1,M
      IF(PIV)24,24,4
    4 IF(IER)7,5,7
    5 IF(PIV-TOL) 6,6,7
    6 IER=K-1
    7 LT=J-K
      LST=LST+K
      PIVI=1./A(I)
      DO 8 L=K,NM,M
      LL=L+LT
      TB=PIVI*R(LL)
      R(LL)=R(L)
    8 R(L)=TB
      IF(K-M)9,19,19
    9 LR=LST+(LT*(K+J-1))/2
      LL=LR
      L=LST
      DO 14 II=K,LEND
      L=L+II
      LL=LL+1
      IF(L-LR)12,10,11
   10 A(LL)=A(LST)
```

```
      TB=A(L)
      GO TO 13
   11 LL=L+LT
   12 TB=A(LL)
      A(LL)=A(L)
   13 AUX(II)=TB
   14 A(L)=PIVI*TB
      A(LST)=LT
      PIV=0.
      LLST=LST
      LT=0
      DO 18 II=K,LEND
      PIVI=-AUX(II)
      LL=LLST
      LT=LT+1
      DO 15 LLD=II,LEND
      LL=LL+LLD
      L=LL+LT
   15 A(L)=A(L)+PIVI*A(LL)
      LLST=LLST+II
      LR=LLST+LT
      TB=DABS(A(LR))
      IF(TB-PIV)17,17,16
   16 PIV=TB
      I=LR
      J=II+1
   17 DO 18 LR=K,NM,M
      LL=LR+LT
   18 R(LL)=R(LL)+PIVI*R(LR)
   19 IF(LEND)24,23,20
   20 II=M
      DO 22 I=2,M
      LST=LST-II
      II=II-1
      L=A(LST)+.5
      DO 22 J=II,NM,M
      TB=R(J)
      LL=J
      K=LST
      DO 21 LT=II,LEND
      LL=LL+1
      K=K+LT
   21 TB=TB-A(K)*R(LL)
      K=J+L
      R(J)=R(K)
   22 R(K)=TB
   23 RETURN
   24 IER=-1
      RETURN
      END
```

Appendix 5

Listing of Function Subprograms

F and DFDB

for example given in text

```
FTN4,L,B
      DOUBLE PRECISION FUNCTION F(B,X)
C
C     FUNCTION F WILL EVALUATE THE EXPLICIT FUNCTION F AT X FOR
C     THE CURRENT PARAMETER ESTIMATES STORED IN B. THE FUNCTION USED
C     IS ONLY AN EXAMPLE. THE USER, DEPENDING ON THE APPLICATION,
C     MUST SPECIFY HIS OWN FORMULATION FOR F.
C
C     THE FUNCTION USED IN THE EXAMPLE WAS :
C
C                        b2*x            b4*x
C     F(x)=b1*(1.-e     )+b3*(1.-e     )
C
      DOUBLE PRECISION B,X
      DIMENSION B(1)
      F=B(1)*(1.-DEXP(B(2)*X))+B(3)*(1.-DEXP(B(4)*X))
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00114 COMMON = 00000

```
      DOUBLE PRECISION FUNCTION DFDB(I,BETA,X)
C
C     FUNCTION DFDB WILL EVALUATE THE VALUE OF THE PARTIAL DERIVATIVE
C     OF THE FUNCTION F WITH RESPECT TO THE Ith PARAMETER (INDICATED
C     BY I) FOR THE CURRENT PARAMETER ESTIMATES (EACH PARAMETER DOES
C     NOT NECCESSARILY COME INTO THE CALCULATION). NOTE, AS THERE ARE
C     ONLY 4 PARAMETERS, THEN I<1 OR I>4 WILL BE FLAGGED AS AN ERROR
C     AND THE PROGRAM WILL BE TERMINATED.
C
      DOUBLE PRECISION BETA(1),X
      IF(I.GT.4.OR.I.LT.1) STOP 1
      GO TO (1,2,3,4),I
    1 DFDB=1.D0-DEXP(BETA(2)*X)
      RETURN
    2 DFDB=-BETA(1)*X*DEXP(BETA(2)*X)
      RETURN
    3 DFDB=1.D0-DEXP(BETA(4)*X)
      RETURN
    4 DFDB=-BETA(3)*X*DEXP(BETA(4)*X)
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00157 COMMON = 00000

Appendix 6

Listing of Function Subprogram

DFDB

(uses a method of numerical differentiation)

```
      DOUBLE PRECISION FUNCTION DFDB(I,BB,X)
C
C     FUNCTION DFDB WILL EVALUATE THE VALUE OF THE PARTIAL DERIVATIVE
C     OF THE FUNCTION F WITH RESPECT TO THE Ith PARAMETER (INDICATED
C     BY I) FOR THE CURRENT PARAMETER ESTIMATES AT X. THE METHOD OF
C     NUMERICAL DIFFERENTIATION IS BASED ON RICHARDSON'S AND ROMBERG'S
C     EXTRAPOLATION METHOD AS APPLIED TO THE SEQUENCE OF CENTRAL
C     DIFFERENCES ASSOCIATED WITH THE POINT PAIRS
C              (BB(I)-(K*HH)/5,BB(I)+(K*HH)/5),K=1,5
C
C     REFERENCES
C         FILLIPI,S. AND ENGELS,H.(1966),
C         "ALTES UND NEVES ZUR NUMERISCHEN DIFFERENTIATION",
C         ELECTRONISCHE DATENVERARBEITUNG, ISS. 2(1966), PP. 57-65.
C
C         MOORE,R.F.(1979),
C         "ESTIMATION OF PARAMETERS USING LEAST SQUARES",
C         BMR RECORD (IN PREP.)
C
C
      DOUBLE PRECISION Z,AUX,A,B,C,DH,F,BB(1),HH,X,F
      DIMENSION AUX(5),BBB(1)
      H=1.D-2*BB(I)
C
C     GENERATE STEPSIZE HH FOR DIVIDED DIFFERENCES
    1 C=ABS(H)
    2 HH=.5D-2
      IF(C-HH)3,4,4
    3 HH=C
    4 BB(I)=BB(I)+HH
      A=F(BB,X)
      BB(I)=BB(I)-2.D0*HH
      B=F(BB,X)
      BB(I)=BB(I)+HH
      Z=DABS((A-B)/(HH+HH))
      A=.5D0*DABS(A+B)
      HH=.5D-2
      IF(A-1.D0)6,6,5
    5 HH=HH*A
    6 IF(Z-1.D0)8,8,7
    7 HH=HH/Z
    8 IF(HH-C)10,10,9
    9 HH=C
C
C     INITIALISE DIFFERENTIATION LOOP
   10 BB(I)=BB(I)+HH
      Z=F(BB,X)
      BB(I)=BB(I)-2.D0*HH
      Z=(Z-F(BB,X))/(HH+HH)
      BB(I)=BB(I)+HH
      J=5
      JJ=J-1
      AUX(J)=Z
      DH=HH/DBLE(FLOAT(J))
      DZ=1.E38
```

```
C
C     START DIFFERENTIATION LOOP
   11 J=J-1
      C=J
      HH=C*DH
      BB(I)=BB(I)+HH
      AUX(J)=F(BB,X)
      BB(I)=BB(I)-2.D0*HH
      AUX(J)=(AUX(J)-F(BB,X))/(HH+HH)
      BB(I)=BB(I)+HH
C
C     INITIALISE EXTRAPOLATION LOOP
      D2=1.E38
      B=0.D0
      A=1.D0/C
C
C     START EXTRAPOLATION LOOP
      DO 12 II=J,JJ
      D1=D2
      B=B+A
      HH=(AUX(II)-AUX(II+1))/(B*(2.D0+B))
      AUX(II+1)=AUX(II)+HH
C
C     TEST ON OSCILLATING INCREMENTS
      D2=DABS(HH)
      IF(D2-D1)12,13,13
   12 CONTINUE
C     END EXTRAPOLATION LOOP
C
C     UPDATE RESULT VALUE Z
      II=JJ+1
      GO TO 14
   13 D2=D1
      JJ=II
   14 IF(D2-DZ)15,16,16
   15 DZ=D2
      Z=AUX(II)
   16 IF(J-1)17,17,11
C     END OF DIFFERENTIATION LOOP
C
   17 DFDB=Z
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00598 COMMON = 00000

Appendix 7

Listing of Function DFDB (Simplified Version)

```
      DOUBLE PRECISION FUNCTION DFDB(I,BB,X)
C
C     FUNCTION DFDB WILL EVALUATE THE VALUE OF THE PARTIAL DERIVATIVE
C     OF THE FUNCTION F WITH RESPECT TO THE Ith PARAMETER (INDICATED
C     BY I) FOR THE CURRENT PARAMETER ESTIMATES AT X. THE METHOD
C     EVALUATES THE PARTIAL DERIVATIVE USING THE FORWARD FUNCTION F.
C     DFDB SIMPLY CALCULATES THE GRADIENT OF F W.R.T. BB(I) IN THE
C     REGION (BB(I)-DB;BB(I)+DB) WHERE DB=C*BB(I); C IS A CONSTANT.
C
      DOUBLE PRECISION  BB(1),X,C,Y2,DB,DB2
      DATA C/0.1D-1/
      DB=C*BB(I)
      DB2=DB+DB
      BB(I)=BB(I)-DB
      Y2=F(BB,X)
      BB(I)=BB(I)+DB2
      DFDB=(Y2-F(BB,X))/DB2
      BB(I)=BB(I)-DB
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00123 COMMON = 00000