Numerical solution of nonlinear regressions under linear constraints
by Ning-Chia Yeh

A thesis submitted in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Applied Economics
Montana State University

Abstract:
An algorithm for nonlinear least squares estimation, involving nonorthogonal data is described in this paper. The estimation procedure is developed by imposing an ellipsoid constraint on the regular least squares parameter estimators based on the statistical precision of estimation. The theoretical basis is reviewed for obtaining such biased estimators with a view to minimizing the mean square error.

Numerical tests are presented which allow a comparison between the results obtained in this work and those by the Marquardt method. The new technique benefits in terms of iteration number when the data are badly ill-conditioned, but not when having well-designed or moderately ill-conditioned data.

A method for nonlinear estimation subject to linear constraints is also considered. Moreover, an application of this new algorithm to solving nonlinear simultaneous equations is discussed. A FORTRAN program is developed for the new scheme and is described by reference to several modifications due to R. Fletcher.

NUMERICAL SOLUTION OF NONLINEAR REGRESSIONS UNDER LINEAR CONSTRAINTS

by

NING-CHIA YEH

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Applied Economics

Approved:

Oscar R Burt
Chairman, Examining Committee

R. J. McConnen
Head, Major Department

D. L. Greening
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

April, 1975

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Montana State University, I agree that the Library shall make it freely available for inspection. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by my major professor, or, in his absence, by the Director of Libraries. It is understood that any copying or publication on this thesis for financial gain shall not be allowed without my written permission.

Signature Ning-Chia Yeh

Date Apr. 10, 1995

ACKNOWLEDGMENTS

The author wishes to express deep appreciation to Dr. O. R. Burt for his advice and guidance during the course of this work. The author would also like to thank Mr. S. B. Townsend for his programming assistance.

TABLE OF CONTENTS

	Page
VITA	ii
ACKNOWLEDGMENTS	iii
ABSTRACT	v
CHAPTER I: INTRODUCTION	1
CHAPTER II: THE GENERALIZED MARQUARDT METHOD	7
2.1. Formulation of the Problem	10
2.2. Calculation of the Cut-Off Value of λ	12
2.3. Computational Algorithm	15
2.4. Numerical Results and Discussion	18
CHAPTER III: LEAST SQUARES UNDER LINEAR CONSTRAINTS	22
3.1. The Basic Method	22
3.2. Numerical Procedure	24
3.3. Covariance Matrix	26
CHAPTER IV: SOLVING SYSTEMS OF NONLINEAR EQUATIONS	29
CHAPTER V: CONCLUSIONS	31
5.1. Computer Program	31
5.2. Discussion	31
APPENDICES	34
Appendix A: Instructions	35
Appendix B: Flow Diagram	63
Appendix C: Sample Problem	68
Appendix D: Computer Program	79
LITERATURE CITED	112

ABSTRACT

An algorithm for nonlinear least squares estimation, involving nonorthogonal data is described in this paper. The estimation procedure is developed by imposing an ellipsoid constraint on the regular least squares parameter estimators based on the statistical precision of estimation. The theoretical basis is reviewed for obtaining such biased estimators with a view to minimizing the mean square error.

Numerical tests are presented which allow a comparison between the results obtained in this work and those by the Marquardt method. The new technique benefits in terms of iteration number when the data are badly ill-conditioned, but not when having well-designed or moderately ill-conditioned data.

A method for nonlinear estimation subject to linear constraints is also considered. Moreover, an application of this new algorithm to solving nonlinear simultaneous equations is discussed. A FORTRAN program is developed for the new scheme and is described by reference to several modifications due to R. Fletcher.

CHAPTER I

INTRODUCTION

Regression analysis is a statistical tool which has been used by economists for years. It is a numerical technique of determining a functional relation between a response and sets of input data. The mathematical form of the relation is assumed to be known and is represented by the regression equation

$$Y_i = f(x_{i1} \ldots x_{im}; b_1 \ldots b_p) \quad , \quad i = 1, \ldots, n \tag{1.1}$$

where $(b_1 \ldots b_p)'$ is the p-vector of unknown parameters; $(x_{i1} \ldots x_{im})'$ is an m-vector of independent variables at the i-th observation; Y_i is the value of response y_i predicted by (1.1) for the i-th observation, and there are n observations available. The method often used for the estimation of parameters is the Method of Least Squares. The least squares estimates, $\hat{\underline{b}}$, are obtained by minimizing the error sum of squares

$$\phi(\underline{b}) = \sum_{i=1}^{n} (y_i - Y_i)^2. \tag{1.2}$$

When f in (1.1) is linear in the parameters, a statistical theory of linear regression has been well developed, while if f is nonlinear in the parameters, estimation procedure has been more difficult. This paper is concerned with the methodology of nonlinear regression problems.

In statistical applications, $n > p$, and the objective is to estimate the function in (1.1) as the mean of the $\{y_i\}$ which are specified to have certain statistical properties. Another common application is where $n = p$ and (1.1) represents a system of nonlinear equations. In the case where $n = p$, the solution value of the least squares criterion given in (1.2) is zero. The role of parameters and variables is interchanged in the interpretation of (1.1) as a system of equations. This topic is discussed in detail in Chapter IV.

An approach widely used to compute least squares estimates of nonlinear parameters is the Gauss-Newton iterative method which linearizes the parameters by a Taylor expansion of f and the corrections of parameters at each iteration are calculated by using a linear regression. The linear approximation at a point $\underline{b} + \underline{\delta}$ where $\underline{\delta}$ is assumed small relative to $\underline{b}$ is given by

$$Y_i = f_i(\underline{x}, \underline{b} + \underline{\delta}) = f_i(\underline{x},\underline{b}) + \sum_{j=1}^{p} \frac{\partial f_i}{\partial b_j} \delta_j.$$

The iterative procedure uses

$$\underline{Y} = \underline{f}^{o} + P\underline{\delta}, \tag{1.3}$$

where $\underline{f}^{o}$ is the predicted value of the dependent variable at the previous iteration and P is an $n \times p$ matrix of partial derivatives:

$$P = \{\frac{\partial f_i}{\partial b_j}\}, \qquad \begin{matrix} i = 1, \ldots, n; \\ j = 1, \ldots, p. \end{matrix} \tag{1.4}$$

whose elements depend on $\underline{b}$ in the nonlinear model. Then the parameter correction vector $\underline{\delta}$ is obtained by treating (1.3) as a problem in linear regression which yields the system of linear equations

$$A\underline{\delta} = \underline{g}, \tag{1.5}$$

where: $A = P'P,$ (1.6)

$$\underline{g} = P'(\underline{y} - \underline{f}^{o}). \tag{1.7}$$

Once (1.5) has been solved, the parameters will be updated by

$$\underline{b}^{(q+1)} = \underline{b}^{(q)} + \underline{\delta}^{(q)} \tag{1.8}$$

where q is the iteration count. The procedure will terminate when a given convergence criterion is passed, which is typically of the form $|\delta_j|/|b| < \varepsilon$, $j = 1, \ldots, p$.

In practice difficulties of numerical convergence often occur because of high intercorrelation among the column vectors of P, which is equivalent to one or more small eigenvalues for P'P. It is common that the over-estimates of some parameters make Newton's method fail to converge unless the initial starting position is close to the final solution.

At the other extreme, various steepest descent methods are used which correct the parameter vector along the gradient vector associated with the error sum of squares. Steepest descent methods are known to be convergent for a small enough step size, but are often awfully slow.

Various modifications to the Newton method have therefore been developed [5, 10, 11]. The algorithm of Marquardt [11] which has been shown to be closely related to ridge regression for linear models, appears to be the most promising algorithm. It involves the system of equations

$$(A + \lambda I)\underline{\delta} = \underline{g} \tag{1.9}$$

as the counterpart of (1.5), where λ is a small nonnegative constant, I is the identity matrix, and $A = P'P$. Three important features of (1.9) are:

1) length of the correction vector decreases as λ increases;
2) the correction vector moves towards the steepest descent vector as λ increases; and
3) the eigenvalues of $A + \lambda I$ are those of A each increased by λ.

The crux of the method is to increase λ at any trial iteration in which an improvement in the error sum of squares is not obtained. Marquardt has proven that there always exists a sufficiently large λ to get an improvement, so that in practice, an improvement is obtained by systematically increasing λ or else the convergence test is passed.

Since matrix $A + \lambda I$ is better conditioned than A, it removes practical problems of matrix singularity within the arithmetic limits on automatic computers. A successful algorithm must keep λ relatively small when no difficulties of singularity in A or monotone improvement in the criterion are experienced.

Marquardt's method for choosing λ is extremely simple. If an improved error sum of squares is experienced, λ is decreased by some factor (usually 10), otherwise λ is increased by a factor (again, usually 10). The initial value is chosen arbitrarily, $\lambda = 0.01$ is usually used.

A further improvement based on Marquardt's method was recently developed by Fletcher [4]. The modification of the procedure is determine values of λ in each step as follows:

1) Introduction of a factor R, defined as the ratio of the actual reduction in error sum of squares to the predicted one under a linear model, to determine how to change λ. If R is too small, λ is increased by a factor ν; if R is too big, λ is chopped by a factor, usually 2; and for some intermediate values, λ is kept at its old value.

2) A multiplier ν, used to increase λ when an increase is implied, is chosen according to a criterion giving the most rapid rate of improvement. This factor ν is calculated each time that it is used in the algorithm by a formula which approximates the optimality criterion derived when λ is relatively large.

3) A cut-off value λ_c, the smallest non-zero value of λ, is chosen at which the vector length of $\underline{\delta}$ is approximately one-half of that at $\lambda = 0$. Fletcher showed that $\lambda_c = 1/\mathrm{tr}(A^{-1})$ provides a value of λ_c which is a good approximation with an error of no real harm for convergence.

An economist's use of regression techniques generally applies to nonexperimental data. Data obtained without prior experimental design are typically characterized by independent variables that are related to one another in the statistical sense of correlation. This paper develops an algorithm for nonlinear estimation which was thought to have a good deal of promise in extremely multicollinear data. The present work is only a preliminary to further investigation of nonlinear estimation problems.

CHAPTER II

THE GENERALIZED MARQUARDT METHOD

A digression into linear regression with an ill-conditioned "design" matrix is used to introduce the generalized Marquardt algorithm. In discussing the linear model, X is used to denote the $n \times p$ matrix of observations on independent variables. In actuality, P = X if the equation in (1.1) is linear in the parameters, but a separate notation is used for clarification.

Consider the general linear model

$$\underline{y} = X\underline{\beta} + \underline{\varepsilon} \tag{2.1}$$

where $\underline{y}$ is an n-dimensional vector of observation, $\underline{\beta}$ is a p-dimensional vector of unknown parameters, and $\underline{\varepsilon}$ is an n-dimensional error vector with mean zero and covariance matrix $\sigma^2 I$.

The solution of (2.1) will be scale invariant if the model is chosen under a normalized system. It is done if X has been post-multiplied by a diagonal matrix, say D, with diagonal elements equal to the reciprocal of the vector length of corresponding columns of the matrix X and $\underline{\beta}$ has been pre-multiplied by D^{-1}. This choice of scale makes the matrix $X'X$ a correlation matrix with sample moments about the origin. It is assumed that X and $\underline{\beta}$ in (2.1) have been normalized as described.

It is common in linear regression analysis that the columns of the matrix X will be nearly linearly dependent which results in a very

small determinant of the matrix X'X, and consequently, in relatively large variances in the least squares estimators of regression coefficients. In other words, if the input matrix of the design is ill-conditioned, poor statistical precision may be expected.

An analysis of precision of estimation and identification of multicollinearity was presented by Silvey [15]. He concluded that relatively precise estimation is possible in the direction of eigenvectors of X'X corresponding to large eigenvalues, and relatively imprecise estimation in those directions corresponding to small eigenvalues.

In recent years, a class of biased estimators has been investigated for dealing with estimation problems involving poorly-conditioned data. In the first, Hoerl and Kennard [6, 7, 8] introduced ridge regression with a view to reducing the mean square error of parameter estimators. A family of estimates based on the normal equations

$$(X'X + \lambda I)\underline{\beta} = X'\underline{y} \quad , \quad \lambda \geq 0$$

are calculated, and then, the value of λ which gives a "stable solution" and hopefully small mean square errors is selected. Subsequently, Marquardt [12] showed the similarity in properties between the ridge estimator and a generalized inverse estimator. The latter algorithm restricts the estimated parameter vector in a vector space with reduced dimension.

In ridge regression, all components of the estimated parameter vector are constrained symmetrically in all directions of the normalized parameter space. Since imprecise estimation is often caused by movements in parameter space in the directions of eigenvectors associated with small eigenvalues in the linear model, an improved method formulated below is primarily a generalization of ridge regression into elliptically constrained estimators with axes of the ellipsoid aligned with eigenvectors of X'X.

For the convenience in analysis, an orthogonal version of (2.1) will be used. To do this, define

$$Z = XV \tag{2.2}$$

$$\underline{\gamma} = V'\underline{\beta} \tag{2.3}$$

where V is an orthogonal matrix with columns comprised of the normalized eigenvectors of the matrix X'X. V has the following properties:

$$V'V = VV' = I \tag{2.4}$$

$$V'(X'X)V = M \tag{2.5}$$

where M is a diagonal matrix of eigenvalues of X'X. Then the model in (2.1) can be written as

$$\underline{y} = Z\underline{\gamma} + \underline{\varepsilon} \tag{2.6}$$

with orthogonal independent variables.

2.1. Formulation of the Problem

The basic concept presented here is that instead of using a spherical constraint on the components of parameter vectors as implied by ridge regression or Marquardt's nonlinear least squares method, an elliptical constraint based on the statistical precision of estimates is imposed. The problem now becomes one of calculating $\hat{\underline{\gamma}}$, which yields a minimum value of the error sum of squares:

$$\phi(\underline{\gamma}) = (\underline{y} - Z\underline{\gamma})'(\underline{y} - Z\underline{\gamma}) \tag{2.7}$$

subject to a constraint of the form

$$\underline{\gamma}'M^{-1}\underline{\gamma} \leq r^2 \quad \text{or} \quad \sum_{j=1}^{p} \gamma_j^2/\mu_j \leq r^2. \tag{2.8}$$

The p-dimensional constraint ellipsoid has the lengths of its axes inversely related to the statistical precision of estimation in directions of the axes. More specifically, the j-th axis length is inversely proportional to the standard deviation of the estimator of γ_j.

The Lagrangian equation is

$$L = \underline{y}\underline{y} - 2\underline{y}'Z\underline{\gamma} + \underline{\gamma}'\underline{\gamma} + \lambda(\underline{M}^{-1}\underline{\gamma} - r^2),$$

where λ is the Lagrangian multiplier. Setting $\partial L/\partial\underline{\gamma} = 0$, the normal equation is then

$$(M + \lambda M^{-1})\underline{\gamma} = Z'\underline{y}. \tag{2.9}$$

By using (2.2) and (2.3), Eq. (2.9) in the nonorthogonal system is

$$[X'X + \lambda(X'X)^{-1}]\underline{\beta} = X'\underline{y}. \tag{2.10}$$

Equation (2.10) implies that the amounts added to the diagonal elements of the matrix X'X are selected such that they are inversely related to precision of estimation in the directions of corresponding eigenvectors instead of being a constant as used in ridge regression. The solution vector of this approach has essentially the same properties as the ridge estimator except the error sum of squares is minimized over an ellipsoid instead of a sphere with the ellipsoid oriented according to eigenvectors.

The above results are, then, applied to a modification of the Marquardt's algorithm for nonlinear least squares estimation. The counterpart of (2.10) at a given iteration in nonlinear least squares is

$$[P'P + \lambda(P'P)^{-1}]\underline{\delta} = \underline{g} \tag{2.11}$$

which is in contrast to (1.9) under Marquardt's algorithm.

The bias of the correction vector, $\underline{\delta}$, toward the gradient vector in the nonlinear problem as λ is increased in (2.11) would not be symmetric among its components. In the elliptically constrained system, the components which cannot be precisely estimated will be forced toward the gradient vector relatively more for a given λ, and those precisely estimated will be left free so that accelerated convergence can be achieved in these directions. The intuitive idea is that the error sum

of squares criterion should be relatively insensitive to the γ's in the directions of imprecise estimation as measured in the linear approximation model.

One feature of this generalized Marquardt method is the rapid terminating of the iterative procedure. The iterative procedure is usually said to have converged when the rate of corrections in all directions, measured as the ratio to the current parameter estimates, is no larger than a pre-assigned small constant ε. Since all parameters are not allowed to change symmetrically in parameter space, some directions in which estimation is imprecise are restricted. It tends to give a convergence criterion which is asymmetric compared to what it would be under the Marquardt algorithm.

2.2. Calculation of the Cut-Off Value of λ

The only difference in the results of the foregoing analysis from the Marquardt algorithm is substitution of $\lambda(P'P)^{-1}$ in place of λI in (1.9). However, the method for choosing λ under Fletcher's modification must be examined to see if it is still applicable. The cut-off value, λ_c, suggested by Fletcher has to be modified if (2.11) is employed for solving least squares problems.

As discussed in connection with Fletcher's work, λ_c is chosen such that the reduction in length of the correction vector is no more than one-half of that with $\lambda = 0$. The basic idea is to choose λ_c such that

an increase in λ from zero to a positive level is essentially equivalent to doubling λ when λ is already positive. For convenience, we use the linear model notation below, but just keep in mind that P replaces X in the nonlinear application. Let

$$X'X = VMV' = \sum_{i=1}^{p} \mu_i \underline{v}_i \underline{v}_i'$$

where μ's are eigenvalues of X'X. Eq. (2.5) has been utilized above. Then

$$(X'X)^{-1} = VM^{-1}V' = \sum_{i=1}^{p} \underline{v}_i \underline{v}_i'/\mu_i.$$

The solution of (2.10) can be written

$$\underline{\beta}(\lambda) = [X'X + \lambda(X'X)^{-1}]^{-1}X'y$$

$$= [\sum_{i=1}^{p} \mu_i \underline{v}_i \underline{v}_i' + \lambda \sum_{i=1}^{p} \underline{v}_i \underline{v}_i'/\mu_i]^{-1}X'\underline{y}$$

$$= [\sum_{i=1}^{p} (\mu_i^2 + \lambda)\underline{v}_i \underline{v}_i'/\mu_i]^{-1}X'\underline{y}.$$

Define matrix

$$Q = \{\delta_{ij}(\mu_i^2 + \lambda)/\mu_i\},$$

where δ_{ij} is the Kronecker delta;

$$\delta_{ij} = \begin{cases} 1 \ , & i = j \\ 0 \ , & i \neq j \end{cases}$$

$$\underline{\beta}(\lambda) = (VQV')^{-1}X'\underline{y} = (VQ^{-1}V')X'\underline{y}$$

$$= [\sum_{i=1}^{p} \underline{v}_i\underline{v}_j\mu_i/(\mu_i^2 + \lambda)]X'\underline{y}$$

whence

$$||\underline{\beta}(\lambda)|| = \underline{\beta}(\lambda)'\underline{\beta}(\lambda)$$

$$= (X'\underline{y})'(VQ^{-1}V')'(VQ^{-1}V')X'\underline{y}$$

$$= (X'\underline{y})'VQ^{-2}V'X'\underline{y}$$

$$= \sum_{i=1}^{p} (\frac{\mu_i}{\mu_i^2 + \lambda})^2 (X'\underline{y})'\underline{v}_i\underline{v}_i'(X'y)$$

$$= \sum_{i=1}^{p} [(\frac{\mu_i}{\mu_i^2 + \lambda})(X'\underline{y})'\underline{v}_i]^2.$$

Thus,

$$\frac{1}{2}||\underline{\beta}(0)|| = \sum_{i=1}^{p} [\mu_i(X'\underline{y})'\underline{v}_i/2\mu_i^2]^2$$

$$\leq \sum_{i=1}^{p} [\mu_i(X'\underline{y})'\underline{v}_i/(\mu_i^2 + \lambda)]^2$$

if $2\mu_i^2 \geq \mu_i^2 + \lambda$ or $\lambda \leq \mu_i^2$ for all i. Let λ_c be defined such that

$$\lambda_c \leq \min\{\mu_i^2\}$$

which implies the following inequality

$$\frac{1}{2}||\underline{\beta}(0)|| \leq ||\underline{\beta}(\lambda_c)||.$$

Consequently, λ_c can be conservatively given [14] as

$$\lambda_c = [1/\mathrm{tr}(X'X)^{-1}]^2$$

$$\leq 1/\max\{1/\mu_i^2\} = \min\{\mu_i^2\}$$

which makes the length of the correction vector less than one-half of that for $\lambda = 0$.

2.3. Computational Algorithm

A general outline of the modified algorithm is now clear. In nonlinear estimation problems, given a model $y = f(X,\underline{b})$, the normal equation at each iteration is given by (2.11) where $\underline{g} = P'(\underline{y} - \underline{f})$, P is the parameter partial derivative matrix calculated at the previous iteration, and $\underline{y} - \underline{f}$ is defined as the difference between the observed value of the dependent variable and predicted value calculated at the previous iteration. Eq. (2.11) is modified by introducing the scaling operation under the assumption that P is in the original units. To do this, define

$$W = PD \tag{2.12}$$

$$\underline{\alpha} = D^{-1}\underline{\delta} \tag{2.13}$$

where

$$P = \{\partial f_i/\partial b_j\} \quad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix} \tag{2.14}$$

$$D = \{(\sum_{k=1}^{n} p_{ki}^2)^{-1/2}\delta_{ij}\} \quad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix} \tag{2.15}$$

δ_{ij} = the Kronecker delta,

and $\underline{\delta}$ is the parameter correction vector in original units. Thus,

$$(W'W)^{-1} = D^{-1}(P'P)^{-1}D^{-1}. \tag{2.16}$$

Eq. (2.11) where W and $\underline{\alpha}$ given in (2.12) and (2.13) respectively replace P and $\underline{\delta}$ can be rewritten as

$$[DP'PD + \lambda D^{-1}(P'P)^{-1}D^{-1}]D^{-1}\underline{\delta} = DP'(\underline{y} - \underline{f}). \tag{2.17}$$

Premultiplied by D^{-1}, (2.17) becomes

$$[P'P + \lambda D^{-2}(P'P)^{-1}D^{-2}]\underline{\delta} = P'(\underline{y} - \underline{f}). \tag{2.18}$$

Since $\underline{g} = P'(\underline{y} - \underline{f})$, the generalized algorithm now involves the system of equations

$$[P'P + \lambda D^{-2}(P'P)^{-1}D^{-2}]\underline{\delta} = \underline{g}. \tag{2.19}$$

Equation (2.19) is used in the computer program at each iteration to obtain the correction term for the nonlinear parameters. The practical computation of $(P'P)^{-1}$ is carried out by computing

$$(P'P + \eta I)^{-1} \tag{2.20}$$

where η is a nonnegative constant arbitrarily given for the purpose of circumventing any numerical inversion difficulties caused by the positive definite and symmetric, but possibly ill-conditioned, matrix P'P. This successive approximation procedure used to get the inverse when singular problems arise is stopped when the inverse checks to a particular level of accuracy.

A new error sum of squares $\phi^{(q+1)}$ can be computed based on the updated

$$\underline{b}^{(q+1)} = \underline{b}^{(q)} + \underline{\delta}^{(q)} \tag{2.21}$$

where q is an iteration count. The choice of λ based on Fletcher's method is used to improve the error sum of squares.

Since the correction component is kept small in the direction of imprecise estimation, this iterative procedure could possibly pass the ordinary epsilon test before the minimum error sum of squares is achieved. Therefore, it is important to either decrease the value of ε or shift over to the Marquardt method at some proper value, say ε_1, before the final criterion value of ε is reached. The latter method was

chosen here to ensure that the minimum error sum of squares is obtained (technically a local minimum).

In summary, the iteration scheme is:

1) initialize $\underline{b}$;

2) construct Eq. (2.19);

3) calculated $\underline{\delta}$ using (2.19) incorporated into Fletcher's technique;

4) update $\underline{b}$ using (2.21);

5) check for convergence; and

6) repeat starting at step (2) if no convergence.

It is important to remember that the initial choice of $\underline{b}$ in step (1) determines the local minimum to which the algorithm converges, and the problem of multiple local minima must not be neglected in practical applications.

2.4. Numerical Results and Discussion

A few test problems are given for the purpose of comparing the new algorithm with the standard Marquardt algorithm. These numerical examples illustrate the likely performance of the method proposed above.

A set of weather data has been utilized in conjunction with wheat experiments both on continuous cropping and alternate crop-fallow systems. Two models were used:

1) exponential function primarily for continuous cropping data

$$y = b_1 \exp(-b_2 u^{-b_3}) \tag{2.22}$$

where

$$u = b_7 x_1 + b_8 x_2 + b_9 x_3 + b_{10} x_4 + b_{11} x_5 + b_{12} x_6 + b_{13} x_7 \tag{2.23}$$

2) polynomial function primarily for alternate crop-fallow data

$$y = b_1 + b_2 u + b_3 u^2 + b_4 u^3 + b_3 u^4 + b_6 u^5 \tag{2.24}$$

where u is defined in (2.23).

The tests to be discussed are:

1) Model I.

2) Model I with parameter b_{13} eliminated by the constraint

$$\sum_{i=7}^{13} b_i = 1 \tag{2.25}$$

and nine parameters left to be estimated.

3) Model I subject to constraint (2.25) using a method called constrained nonlinear estimation to be described in the next chapter. There are therefore ten parameters involved in this case.

4) Model II with b_{13} deleted by relationship (2.25), in addition, b_3 and b_5 are omitted.

5) Model II subject to imposed constraint (2.25) using constrained nonlinear estimation method; b_3 and b_5 are omitted.

6) Model II with b_{13} deleted by (2.25).

7) Model II with b_3 and b_5 omitted.

8) Same as (4) but continuous cropping data have been used.

9) Same as (6) but continuous cropping data have been used.

The results, indicated by the number of iterations to convergence, are given as follows:

Test	Marquardt Method	Generalized Marquardt Method
1	12	13
2	21	23
3	21	19
4	27	24
5	28	26
6	45	32
7	Convergence did not occur at iteration cut-off = 300	104
8	24	25
9	25	24

The modified Marquardt method due to Fletcher has been used. Both models fitted with the continuous cropping data show essentially the same number of iterations to convergence for both methods. Model II shows fewer iterations for the generalized method. The polynomial model employed for alternate crop-fallow data is an extremely multicollinearity problem. As the condition is worsened by leaving higher correlations among column vectors of P, the generalized method displays more advantage.

More data sets should be considered before any substantial conclusions are drawn regarding the relative merits of the generalized algorithm as presented. The computer time for each iteration of the generalized method compared to the Marquardt method is greater which is to be expected because of the extra computation of $(P'P)^{-1}$ required. The generalized procedure has been designed to cope with a model having ill-conditioned data. In addition to the observed data, the particular model chosen to be fitted governs the structure of the P'P matrix; that is, the size distribution of its eigenvalues.

CHAPTER III

LEAST SQUARES UNDER LINEAR CONSTRAINTS

The highly efficient methods which have been developed in recent years for nonlinear regression do not allow for constraints among the parameters except for a procedure without rigorous theoretical basis which has been provided by Marquardt [13]. A serious problem with the Marquardt procedure is that the constraint may only be approximately met, and asymptotic standard errors cannot be obtained for parameter estimators.

In this chapter a method of constrained nonlinear estimation is presented which is a generalization of the procedure given by Theil [16] for linear regression.

3.1. The Basic Method

Consider minimization of the error sum of squares

$$\phi(\underline{\beta}^*) = (\underline{y} - X\underline{\beta}^*)'(\underline{y} - X\underline{\beta}^*) \tag{3.1}$$

subject to

$$\underline{\beta}^{*\prime} H \underline{\beta}^* \leq r^2 \tag{3.2}$$

and linear constraints

$$R\underline{\beta}^* = \underline{c} \tag{3.3}$$

where R and $\underline{c}$ are given matrices of order $\ell \times p$ and an ℓ-vector, respectively. It is assumed that the constraint (3.3) is linearly independent. Let X and $\underline{\beta}^*$ be scaled as described in Section 2.1. An analysis of the above optimization problem is carried out for the linear model and then is applied to nonlinear least squares.

The Lagrangian equation is

$$L = \underline{y}'\underline{y} + \underline{\beta}^{*\prime}(X'X)\underline{\beta}^* - 2\underline{y}'X\underline{\beta}^* +$$

$$2\underline{\rho}[R\underline{\beta}^* - \underline{c}] + \lambda[\underline{\beta}^{*\prime}H\underline{\beta}^* - r^2] \qquad (3.4)$$

where $2\underline{\rho}$ is an ℓ-vector of Lagrange multipliers and λ is also a Lagrange multiplier. Then, $\partial L/\partial\underline{\beta}^* = 0$ gives

$$[X'X + \lambda H]\underline{\beta}^* - X'\underline{y} + R'\underline{\rho} = 0. \qquad (3.5)$$

Multiply (3.5) by $R[X'X + \lambda H]^{-1}$, then use constraints (3.3) and solve for $\underline{\rho}$, we get

$$\underline{\rho} = [RA^{-1}R']^{-1}[RA^{-1}X'\underline{y} - \underline{c}]$$

where $A = X'X + \lambda H$. Substitute $\underline{\rho}$ into (3.5) and solve for $\underline{\beta}^*$,

$$\underline{\beta}^* = A^{-1}X\underline{y} - A^{-1}R'[RA^{-1}R']^{-1}[RA^{-1}X'\underline{y} - \underline{c}]. \qquad (3.6)$$

Let $\underline{\beta} = A^{-1}X'\underline{y}$ for the solution without the linear constraints imposed, the constrained least squares solution will be:

$$\underline{\beta}^* = \underline{\beta} + A^{-1}R'[RA^{-1}R']^{-1}[\underline{c} - R\underline{\beta}]. \tag{3.7}$$

By contrast, the constrained least squares estimates $\underline{\delta}^*$ for a nonlinear model is

$$\underline{\delta}^* = \underline{\delta} + A^{-1}R'[RA^{-1}R']^{-1}[\underline{c} - R\underline{\delta}] \tag{3.8}$$

where $A = P'P + \lambda H$.

In unscaled variables, with $W = PD$ and $\underline{\alpha} = D^{-1}\underline{\delta}$ substituted into the variables in place of P and $\underline{\delta}$, respectively in (3.8), and let

$$S = \begin{cases} P'P + \lambda D^{-2} & \text{if } H = I \text{ in (3.2)} \\ P'P + \lambda D^{-2}(P'P)^{-1}D^{-2} & \text{if } H = (P'P)^{-1} \text{ in (3.2)} \end{cases} \quad \text{for nonlinear parameters.} \tag{3.10}$$

$\underline{\delta}^*$ can be easily simplified to be

$$\underline{\delta}^* = \underline{\delta} + S^{-1}C'(CS^{-1}C')^{-1}(\underline{c} - C\underline{\delta}) \tag{3.11}$$

where

$$\underline{\delta} = S^{-1}P'(\underline{y} - \underline{f}) \tag{3.12}$$

$$C = RD^{-1}.$$

3.2. Numerical Procedure

It is noted that $\underline{\delta}^*$ derived above is the parameter correction vector in the nonlinear least squares model where P is no longer independent of the parameters as in the linear model. Suppose the con-

straints imposed on the parameters are

$$C\underline{b}^{*} = \underline{h}. \tag{3.13}$$

It is required, then, $\underline{b}^{*}$ obtained at each iteration has to satisfy (3.13). Thus,

$$C\underline{b}^{*(q+1)} - C\underline{b}^{*(q)} = 0,$$

which implies that

$$C\underline{\delta}^{*(q)} = 0 \tag{3.14}$$

at each iteration q. Eq. (3.14) is correspondingly the constraints imposed on the correction vector at each iteration under the assumption that the initial vector $\underline{b}$ meets the constraints of (3.13). As a consequence, the formula of $\underline{\delta}^{*}$, by setting $\underline{c} = 0$, is

$$\underline{\delta}^{*} = \underline{\delta} - S^{-1}C'(CS^{-1}C')^{-1}C\underline{\delta}, \tag{3.15}$$

where S is given by (3.10).

The numerical procedure then involves the following steps:

1) initialize $\underline{b}$, denoted by $\underline{b}^{o}$ which satisfies the constraints (3.13);
2) construct S from (3.10);
3) get S^{-1} and $(CS^{-1}C')^{-1}$;
4) calculate $\underline{\delta}$ from (3.12);

5) calculate $\underline{\delta}^*$ from (3.15);

6) update $\underline{b}^*$; and

7) check convergence.

3.3. Covariance Matrix

Under the assumption that the linearized form of a nonlinear model is a good approximation, calculation of the covariance matrix for nonlinear parameters can be approximately derived on the basis of the theory of linear regression.

At the least squares point, the covariance matrix of $\underline{\delta}^*$ is the asymptotic covariance matrix for $\underline{b}^*$ with $\lambda = 0$ and P evaluated with parameters at these final estimates. With $\lambda = 0$, $S = P'P$, and (3.15) becomes

$$\underline{\delta}^* = \{I - (P'P)^{-1}C'[C(P'P)^{-1}C']^{-1}C\}\underline{\delta} \tag{3.16}$$

where $\underline{\delta} = (P'P)^{-1}P'(\underline{y} - \underline{f})$, $\underline{f}$ evaluated at the least squares estimates.

In the vicinity of the least squares estimates, from the statistical theory of the linear model, we have

$$\mathrm{Cov}(\underline{\delta}) = \sigma^2(P'P)^{-1} \tag{3.17}$$

since it is assumed that $\mathrm{Cov}(\underline{y}) = \sigma^2 I$. It is clear, on the basis of linearity assumption, that

$$\mathrm{Cov}(\underline{\delta}^*) = T\mathrm{Cov}(\underline{\delta})T', \tag{3.18}$$

where

$$T = I - (P'P)^{-1}C'[C(P'P)^{-1}C']^{-1}C. \tag{3.19}$$

On substituting (3.19) and (3.17) into (3.18), we get

$$Cov(\underline{\delta}^*) = \sigma^2(P'P)^{-1}\{I - C'[C(P'P)^{-1}C']^{-1}C(P'P)^{-1}\}. \tag{3.20}$$

The algorithm has been tested on several problems with considerable success. Situations can arise where accurate results may not be obtained when using the proposed algorithm.

1) Numerical accuracy in computing the correction vector $\underline{\delta}^*$ deteriorates for an ill-conditioned matrix of λ is zero or very near zero. Either the Marquardt or new algorithm starts with $\lambda = 0$ and if the singularity criterion on $P'P$ is set too small, errors in the computations can be a source of discrepancy from the condition $C\underline{\delta}^* = 0$ and in turn the final estimates. There is no problem with large values of λ because the smallest eigenvalue of the matrix associated with the linear equations is sufficiently large. This problem of $C\underline{\delta}^* \neq 0$ at some point in the algorithm is easily detected by checking the final estimate of $\underline{b}^*$ against the constraints.

2) There can also be a problem in calculating the covariance matrix since $(P'P)^{-1}$ may "explode" the covariance matrix of $\underline{\delta}^*$. The statistical results, such as standard errors and confidence limits,

may not be reliable when P'P is nearly singular. The computations in (3.20) use $(P'P)^{-1}$ and any errors in this matrix are compounded much further by the matrix multiplications and in getting the inverse of $[C(P'P)^{-1}C']$. This problem could occur quite easily without the analyst's knowledge.

CHAPTER IV

SOLVING SYSTEMS OF NONLINEAR EQUATIONS

Another application of the nonlinear estimation technique is solving systems of nonlinear equations. Consider the problem of solving n simultaneous nonlinear equations in n unknowns; that is, of determining the values of vector $\underline{b}$ such that

$$f_i(b_1, \ldots, b_n) = 0 \quad , \quad i = 1, \ldots, n. \tag{4.1}$$

This problem has been pursued in broad directions. Most techniques are modifications of Newton's method which involves the first-order Taylor expansion of f_i and solves the resulting set of linear equations [1] to obtain "corrections" to the variables at each iteration. Most variations rely on using some approximation to the inverse Jacobian and modifying this iteration matrix at each stage of the process.

Let the $n \times n$ Jacobian matrix P be defined by $P_{ij} = \partial f_i / \partial b_j$, then at each iteration we have the set of linear equations

$$(P'P)\underline{\delta} = -P'\underline{f}, \tag{4.2}$$

where the solution $\underline{\delta}$ gives the required step to the next iteration, with P and $\underline{f}$ evaluated at current values of $\underline{b}$. Equation (4.2) has been premultiplied by P' on both sides of the equation for the purpose of having a positive definite and symmetric matrix for a system of linear equations to be solved. In many applications, a region of values of $\underline{b}$

is likely to be encountered where P'P is ill-conditioned, with the unpleasant result that the iterative procedure does not converge.

The algorithm described in Chapter II is readily applied to give the perturbed system:

$$[P'P + \lambda D^{-2}(P'P)^{-1}D^{-2}]\underline{\delta} = \underline{g} \tag{4.3}$$

where $\underline{g} = -P'\underline{f}$. The computational procedure for obtaining the solution is also identical. The solution may not be unique if fewer equations than n are involved or if the system is not independent. Moreover, the minimum value of the error sum of squares in the problem of solving systems of nonlinear equations is known to be zero if a solution for the equations exists.

CHAPTER V

CONCLUSIONS

5.1. Computer Program

A computer program based on the preceding algorithm has been developed and is given in Appendix D. It consists of a number of subroutines which are employed in conjunction with user-supplied subroutine specifying the function f_i and partial derivatives p_{ij}.

The program provides options to estimate parameters by using a Marquardt or Generalized Marquardt algorithm, each based on modifications of R. Fletcher, or to solve systems of simultaneous nonlinear equations. The program allows the imposition of linear constraints on the parameters. In addition, linear relations in the parameters may be specified to obtain standard errors and covariances among these linear relations.

Operational instructions giving all the details of using the program are presented in Appendix A. They include the input instructions, output interpretation, and a list of variables used in the program. The flow diagram of the main computational part for the method is shown in Appendix B. An illustrative sample problem using the program is also shown in Appendix C.

5.2. Discussion

The algorithm proposed in this paper provides, on the whole, an efficient procedure for nonorthogonal data. Only a few points of a

general nature can be made at the present time. Its advantages and disadvantages will become substantiated as further investigation is made.

There are a few further remarks on the present method to ensure that it gives a good performance.

1) The choice of ε_1, the epsilon convergence criterion used in the generalized algorithm before switching over to the Marquardt algorithm depends to a large extent on the user's practical experience with the subject matter. If the value of ε_1 is too large, the computational method is essentially the same as the Marquardt method. A value of 10^{-2} has been found in practice to be a good choice.

2) There should be a more objective way for choosing η, which is used to improve the condition of a nearly singular matrix, than the observation that for most of the problems discussed here a value of 10^{-6} appeared to be optimal. The theoretical justification of such a choice is somewhat flimsy. In general, η may be chosen at any arbitrary value, and large values transform the algorithm toward the Marquardt algorithm by relaxing the elliptical constraint somewhat in the directions of the smallest eigenvalues.

3) The proposed method is one for which a premium in increased computer time must be paid when a system is well-conditioned.

4) It is not known how the proposed method performs in sensitivity to a change in mathematical model or in initial guess of parameters.

Our efforts in investigating the generalized method have not provided any definitive results thus far. It is felt that in order for the estimators based on the generalized method to be thoroughly practical, more study has to be done before any conclusions are drawn in regard to the advantages of the method as presented.

APPENDICES

APPENDIX A: INSTRUCTIONS

I. Introduction

The nonlinear least-squares data fitting problem involves the function

$$f_i = f(x_{i1}, x_{i2}, \ldots, x_{im}; b_1, b_2, \ldots, b_p), \quad i = 1, 2, \ldots, n,$$

where $(b_1, b_2, \ldots, b_p)'$ is the $p \times 1$ vector of parameters $(x_{i1}, x_{i2}, \ldots, x_{im})'$ is an $m \times 1$ vector of independent variables at the i^{th} observation and f_i is the predicted value of the dependent variable y_i at the i^{th} observation. There are n observations available of the form

$$(y_i, x_{i1}, x_{i2}, \ldots, x_{im}), \quad i = 1, 2, \ldots, n.$$

The computational part of the estimation problem is to find a vector $\underline{b}$ for which the error sum of squares

$$\phi(\underline{b}) = \sum_{i=1}^{n} [y_i - f_i]^2$$

is a minimum.

This program provides options to estimate parameters by using a Marquardt or Generalized Marquardt algorithm, each based on modifications of R. Fletcher, to solve simultaneously a system of nonlinear equations or to solve the nonlinear regression problem. The program allows the user to impose linear constraints on the parameters, and in addition, linear relations in the parameters may be specified to obtain standard errors and covariances among these linear relations.

The user must supply three subroutines:

1) SUBROUTINE FCODE (B,Q, SUMQ, PHI) to calculate the regression function and residuals for all the observation data each time it is executed.

2) SUBROUTINE PCODE (I) to calculate the partial derivatives for one data point each time executed.

3) SUBROUTINE SUBZ to perform any data transformations other than those provided by the transformation subroutine. This subroutine is optional, but at least the following statements must be present:

```
SUBROUTINE SUBZ
RETURN
END
```

II. Control Cards

Card 1: Problem Description Card

Label	Format	Card Col.	Contents
SEQ	L1	1	= T System of equations solving
			= F Nonlinear regression
ND	I3	2-4	No. of equations if SEQ = T
			No. of data point if SEQ = F (ND $\leq$ 150)

Label	Format	Card Col.	Contents
NV	I3	5-7	No. of variables if SEQ = T No. of input variables including dependent variable if SEQ = F (NV $\leq$ 25)
NSUBZ	I3	8-10	= 0 No subroutine SUBZ used = 1 SUBZ used
NTRAN	I3	11-13	= 0 No data transformation = 1 Transformations are to be made
NPRNT	I3	14-16	No. of observations to be printed, zero prints one observation
NPT	I3	17-19	No. of auxiliary variables PRNT, to be printed with the residuals
MAXWIDE	I3	20-22	No. of matrix columns across one page of output. Zero gives 10 columns
HEAD	11A4	23-66	Main heading of the problem

Card 2: Format Card

Independent variables are read in first, followed by the dependent variable unless transformations will be made, in which case variables are rearranged to get the above order by using the variable NLOC described below on the transformation control card.

Label	Format	Card Col.	Contents
FMT	20A4	1-80	FORTRAN format of data to be read with left parenthesis in column 1

Card 3: Any data read in from subroutine SUBZ should go here.

Card 4: Transformation Control Card.

Use only if NTRAN is 1 on the first control card. Variable locations (subscripts) to be used in NLOC are ordered as they occur after the transformations have been made. Think of NLOC as optional reordering of variables after the transformations have been made.

Label	Format	Card Col.	Contents
NTR	I2	1-2	No. of transformations to be performed (NTR < 20)
NADD	I2	3-4	No. of variables added (signed integer)
NLOC	I2	5-6	Location of 1st independent variable
	I2	7-8	Location of 2nd independent variable
	.	.	.
	.	.	.
	.	.	.
	(last entry in NLOC)		Location of dependent variable

The total number of locations in NLOC is NV + NADD ($\leq$ 25). If entries for NLOC are left blank, NLOC contains the location of variables created by the transformations and the original variables as read into the computer which have not been replaced by a transformation.

Card 4a: Transformation Card

The number of transformation cards is NTR given on Card 4.

Label	Format	Card Col.	Contents
	I2	1-2	Transformation Code (given below)
	I2	3-4	Storage location of the new variable X(I)
	I2	5-6	First factor location X(J)
	I2	7-8	Second factor location X(K)
	F10.5	9-18	Constant factor C

The changes in location of the variables are sequential with respect to the above transformation cards so the user must be careful not to destroy any variables needed in later transformations of the sequence.

Transformation Code No.	Operation
1	X(I) = LN(X(J)), base e
2	X(I) = LOG(X(J)), base 10
3	X(I) = SIN(X(J))
4	X(I) = COS(X(J))
5	X(I) = TAN(X(J))
6	$X(I) = e^{X(J)}$
7	$X(I) = X(J)^{C}$
8	$X(I) = \lvert X(J) \rvert$
9	If X(J) = 0 then X(I) = C

Transformation Code No.	Operation
10	X(I) = X(J) + X(K)
11	X(I) = X(J) + C
12	X(I) = X(J) - X(K)
13	X(I) = X(J)*X(K)
14	X(I) = X(J)*C
15	X(I) = X(J)/X(K)
16	X(I) = X(J)/C
17	X(I) = C/X(J)
18	X(I) = X(J)
19	X(I) = I, trend variable

Card 5: Variable Name Card

Label	Format	Card Col.	Contents
	5A4	1-20	Name of 1st variable
	5A4	21-40	Name of 2nd variable
	.		.
	.		.
	.		.

continue on subsequent cards until all variables including the dependent (totally NV + NADD) are named; no dependent variable to be named if SEQ = T.

Card 6: Equation Control Card

Label	Format	Card Col.	Contents
MQT	L1	1	= T Marquardt algorithm = F Generalized Marquardt algorithm
NI	I3	2-4	Max. number of iterations allowed
NP	I3	5-7	Total number of parameters ($NP \leq 25$)
NC	I3	8-10	Number of linear constraints ($NC \leq 5$)
NLR	I3	11-13	Number of linear relations in parameters ($NLR \leq 5$)
IO	I3	14-16	Number of omitted parameters
IDIAG	I3	17-19	Number of iterations for which diagnostics are to be printed = 0 No diagnostics > 0 Abbreviated diagnostics < 0 Detailed and abbreviated diagnostics
NRESD	I3	20-22	= 0 No printout and graph of residuals = 1 Printout only if SEQ = T Printout and graph if SEQ = F
NPAR	I3	23-25	= 0 Use final parameters of the previous equation as starting values = 1 User supplies the initial guesses for parameters

Label	Format	Card Col.	Contents
ICONST	I3	26-28	= 0 Program supplies constants = 1 User supplies constants (see Card 9 for definition of these constants)
HEAD	11A4	29-72	Equation Title

Card 7: Initial Parameter Value Card--Use only if NPAR is 1.

Initial values must meet the imposed constraints if NC $\neq$ 0.

Label	Format	Card Col.	Contents
B(1)	F10.0	1-10	Initial value of parameter 1
B(2)	F10.0	11-20	Initial value of parameter 2
.	.	.	.
.	.	.	.
.	.	.	.

continue on subsequent cards--8 values per card.

Card 8: Omitted Parameters Card

Required only if IO on Card 6 is non-zero.

Label	Format	Card Col.	Contents
IOL(1)	I2	1-2	Subscripts of 1st omitted parameter
IOL(2)	I2	3-4	Subscripts of 2nd omitted parameter
.	.	.	.
.	.	.	.
.	.	.	.
IOL(IO)			etc.

Card 9: Constant Card

Use only if ICONST on Card 6 is 1. Constants left blank or zero will be set equal to their nominal values.

Label	Format	Col.	Nominal Value	Contents
KKMAX	I3	1-3	50	Max. number of times that LAMBDA is incremented in one iteration
FS	F6.0	4-9	4	F-statistic value for support plane calculations
T	F6.0	10-15	2	t-statistic value for one parameter confidence limits
EPS1	E8.0	16-23	1.0E-02	Epsilon convergence criterion used in generalized algorithm before switching over to Marquardt algorithm
EPS2	E8.0	24-31	1.0E-05	Final epsilon convergence criterion
TAU	E8.0	32-39	1.0E-03	Used in epsilon convergence criterion to avoid division by zero
LAMBDA	F6.0	40-45	0.0	Initial value of LAMBDA
RHO	F6.0	46-51	.25	The upper bound of Fletcher's 'R' for increasing LAMBDA
SIGMA	F6.0	52-57	.75	The lower bound of Fletcher's 'R' for decreasing LAMBDA
CRTGAM	F6.0	58-63	30.0	Critical angle for Gamma Epsilon Convergence criterion

Label	Format	Col.	Nominal Value	Contents
OMEGA	E8.0	64-71	1.0E-31	Break point for non-singularity in using Cholesky decomposition and inversion
ETA	E8.0	72-79	1.0E-06	Constant used in the new algorithm--never less than 1.0E-15

Card 10: Constraints Input Cards

Required only if NC on Card 6 is non-zero. Number of cards needed is determined by NC and NP. There are 8 values per card. Each constraint starts a new card.

Label	Format	Card Col.	Contents
C(1,1)	F10.0	1-10	Coef. of parameter 1 of 1st constraint
C(1,2)	F10.0	11-20	Coef. of parameter 2 of 1st constraint
. . . . .			
C(1,NP)	F10.0		Coef. of parameter NP of 1st constraint
C(2,1)	F10.0	1-10	Coef, of parameter 1 of 2nd constraint
. . . . .			
C(NC,NP)	F10.0		Coef. of parameter NP of NC^{th} constraint

Card 11: Linear Relation Input Cards

Required only if NLR on Card 6 is non-zero. There are 8 values per card. Each linear relation starts a new card.

Label	Format	Card Col.	Contents
H(1,1)	F10.0	1-10	Coef. of parameter 1 of 1st relation
H(1,2)	F10.0	11-20	Coef. of parameter 2 of 1st relation
. . . . .			
H(1,NP)	F10.0		Coef. of parameter NP of 1st relation
H(2,1)	F10.0	1-10	Coef. of parameter 1 of 2nd relation
. . . . .			
H(NLR,NP)	F10.0		Coef. of parameter NP of NLRth relation

Several equations may be estimated during one run by repeating Cards 6 through 11 as necessary.

III. Dimensioned Variables in the Program

X(200,25): Independent variables (200 observations and 25 parameters)

Y(200): Dependent variable

F(200): Predicted function

Q(200): Residual term defined as Y - F

B(25): Parameters to be estimated

P(25): Partial derivatives of function F with respect to parameters

PRNT(200,5): Auxiliary variable used in FCODE and PCODE which can be written out observation by observation with residuals

A(25,25): Product matrix P^tP of derivatives

C(5,25): Coefficient matrix of linear constraints for parameters

H(5,25): Coefficient matrix of linear relations for parameters

High dimensions can be used by changing dimension statements if necessary.

COMMON statements should be used in every subroutine. The user subroutines should be written as follows:

1) SUBROUTINE FCODE(B, Q, SUMQ, PHI)

```
COMMON NV, ND, NP, IT, NDC, IO, SE RSQ, NPAR, MQT, NC, NLR,
       EPS1, KKMAX, EPS2, TAU, LAMBDA, RHO, SIGMA, CRTGAM,
       IDIAG, NPT, SEQ, OMEGA, FS, IOLEO, IOL(25), T,
       SSQR(2), NI, NRESD, ETA

COMMON/MARDAT/X(200,25), Y(200), F(200), PRNT (200,5)

DOUBLE PRECISION P, A, B(25), SSQR, PRNT, PHI, G, ADIAG,
       DELTA, TEMPAR, DIMENSION Q(200)


Statements to evaluate F(I), Q(I) for I = 1, 2, ..., ND,
       SUMQ and PHI


RETURN

END
```

2) SUBROUTINE PCODE(I)

```
Same blank COMMON statements as used in FCODE

COMMON/MARDAT/X(200,25), Y(200), F(200), PRNT(200,5), Q(200),
       B(25), P(25), A(25,25), G(25), ADIAG(25), DELTA(25),
       TEMPAR(25)
```

DOUBLE PRECISION P, A, B, G, ADIAG, DELTA, SSQR, TEMPAR, PRNT

Statement to evaluate P(K) for K = 1, 2, ..., NP

RETURN

END

3) SUBROUTINE SUBZ

If it is used, same blank and labeled COMMON statements as used in PCODE(I) are required, otherwise only a RETURN and END statements are necessary.

IV. Solution of System of Equations

Given a system of nonlinear equations defined as

$$f_i(b_1, b_2, \ldots, b_p) = 0, \quad i = 1, 2, \ldots, n,$$

this program can be used to solve the system of equations by setting the logical variable SEQ equal to .TRUE.

When this option is used, no data inputs are needed, and the "parameters" used throughout the program are simply the "variables" to be solved. It is not necessary that the number of nonlinear equations equals the number of variables to be solved, however, the solution may not be unique.

The error term q_i is defined as the negative of f_i for i = 1, 2, ..., n, and at a given iteration of the algorithm, q_i is the difference between $f_i(\underline{b})$ and zero. It is like having the dependent variable of

the regression algorithm equal to zero for each observation. The partial derivatives p_{ij} are defined as $\partial f_i / \partial b_j$ for $j = 1, 2, \ldots, p$.

There are as many sets of derivatives as there are equations. All of them must be computed in subroutine PCODE. These subroutines should be written as follows:

1) SUBROUTINE FCODE (B, Q, SUMQ, PHI)

COMMON statements as listed before

Statements to evaluate Q(I) for i = 1, 2, ..., ND, SUMQ and PHI; no F(I) is needed. Remember that Q(I) is just the negative of the function F(I)

RETURN

END

2) SUBROUTINE PCODE(I)

COMMON statements as listed before

GO TO (1, 2, ..., ND)I

1 Statements to evaluate

P(J) = ∂F(1)/∂B(J) for J = 1, 2, ..., NP

GO TO 200

2 Statements to evaluate

P(J) = ∂F(2)/∂B(J) for J = 1, 2, ..., NP

GO TO 200

```
      ND  Statements to evaluate
          P(J) = ∂F(ND)/∂B(J) for J = 1, 2, ..., NP

     200  RETURN
          END
```

The output of this option will end up with the printout of solved "parameters" with no statistical parts associated.

V. Output

The program initially prints the main heading of the problem, followed by data information.

THERE ARE ____ OBSERVATIONS AND ____ VARIABLES ON EACH RECORD

THE DATA WILL BE TRANSFORMED (printed when NTRAN = 1)

____ OBSERVATIONS WILL BE PRINTED.

FORMAT( )

ORIGINAL DATA

OBS 1. 2. . . .

TRANSFORMED DATA (if NTRAN = 1)

OBS 1. 2. . . .

a) Then the equation title is printed, along with input variable information.

EQUATION NO.

MAXIMUM ITERATIONS=

NUMBER OF PARAMETERS+

CONSTRAINT(S)= (if NC ≠ 0)

LINEAR RELATION(S)= (if NLR ≠ 0)

ALGORITHM

THE DEPENDENT VARIABLE IS

EQUATION VARIABLES

1. . . .

. . . . . .

INITIAL PARAMETERS

PARAMETER(1)=

. . . . . .

OMITTED PARAMETERS ARE (or NO OMITTED PARAMETERS)

PROGRAM CONSTANTS

KKMAX=	FS=	T=
EPSILON1=	TAU=	LAMBDA=
RHO=	CRITICAL GAMMA=	OMEGA=
SIGMA=	EPSILON2=	ETA=

CONSTRAINT (if NC ≠ 0, listed by row, 8 columns each row)

LINEAR RELATION (if NLR ≠ 0, listed by row, 8 columns each row)

INITIAL ERROR SUM OF SQUARES=

INITIAL STANDARD ERROR ESTIMATE=

b) With IDIAG < 0, detailed and abbreviated diagnostics are printed at each iteration in the format:

DIAGNOSTICS FOR ITERATION ____ ROUND ____

LAMBDA= LAMBDAO= GAMMA= FLETCHER'S R= SSQR(2)=

(if GAMMA is critical, above line is replaced by

GAMMA CRITICAL; DELM= SSQR(2)= FLETCHER'S R=)

1 2 . . .

DELTA

SUMMARY DIAGNOSITICS FOR ITERATION NUMBER

1 2 . . .

DELTA

PARAMETER

SSQR(1)= SSQR(2)= SUMQ= LAMBDAO=

GAMMA= LAMBDA= FLETCHER'S R= SE2=

With IDIAG > 0, only the abbreviated SUMMARY DIAGNOSTICS are printed. With IDIAG = 0, none of these are printed.

c) One of the following messages will be printed after the final parameter estimates have been obtained:

Message	Contents
EPSILON CONVERGENCE	all estimates pass the ε-test.
GAMMA EPSILON CONVERGENCE	each estimate pass ε-test as GAMMA is critical

Message	Contents
GAMMA LAMBDA CONVERGENCE	LAMBDA > 1 while GAMMA > 90°, calculation has been under large rounding error
MAXIMUM ITERATIONS DONE	some estimates have not passed ε-test when iteration is greater than the pre-assigned cut-off value
PARAMETERS NOT CONVERGING	estimates cannot improve ϕ when the maximum number of times that LAMBDA can be increased in one iteration is reached (standard value 50)

After this message, the following line is printed:

TOTAL NUMBER OF ITERATIONS WAS

with one exception is the case for MAXIMUM ITERATIONS DONE where the following line is printed instead:

THE FOLLOWING PARAMETERS DID NOT PASS THE EPSILON CRITERIA OF ____

d) If NRESD $\neq$ 0, a table of residuals is printed together with some optional output by observation depending on what NPT is.

OBS PRED RESIDUAL

A plot of the residuals always appears then, unless SEQ is true.

e) Printout as follows is given except for the case of PARAMETERS NOT CONVERGING.

TOTAL SUM OF SQUARES=

RESIDUAL SUM OF SQUARES=

SUM OF RESIDUALS=

STANDARD ERROR OF ESTIMATE=

THE MULTIPLE R-SQUARED=

DEGREES OF FREEDOM=

DURBIN-WATSON D STATISTIC=

The multiple coefficient of correlation R^2 is measured by the ratio of regression sum of squares to the total sum of squares. The calculation of R^2 is usually to clarify how successfully the regression explains the variation in the dependent variable. The Durbin-Watson D Statistic defined as

$$D = \sum_{t=1}^{n-1} (\hat{q}_{t+1} - \hat{q}_t)^2 / \sum_{t=1}^{n} \hat{q}_t^2$$

is used in a test for serial correlation analysis involving time series data, i.e., the case where the error q_t at any time t is correlated with q_{t-1}.

f) Detailed statistical results are then listed:

SCALED P TRANSPOSE P

PARAMETER COVARIANCE MATRIX/(SIGMA SQR)

A line as follows is printed if $P'P$ is singular:

SINGULAR MATRIX IN CONFIDENCE LIMIT ROUTINE, STD

ERRORS AND CONFIDENCE LIMITS BELOW ARE RELATIVE.

PARAMETER CORRELATION MATRIX (error message will appear if any)

With NLR $\neq$ 0, statistical results for linear relations are listed:

COVARIANCE MATRIX/(SIGMA SQR) FOR LINEAR RELATION(S)

OF PARAMETERS

RELATION STANDARD ERROR

The confidence region is printed, along with the standard error in the format:

APPROXIMATE 95% LINEAR CONFIDENCE LIMITS:

PARAMETER	PARAMETER ESTIMATE	STANDARD ERROR	ONE-PARAMETER LOWER	ONE-PARAMETER UPPER	SUPPORT PLANE LOWER	SUPPORT PLANE UPPER

g) The printout will repeat if there are more than one equation to be estimated. After the job is completed, a summary is listed:

SUMMARY OF RESULTS

EQUATION NO

PARAMETER

STD ERROR

STANDARD ERROR OF ESTIMATE= RESIDUAL SUM OF SQUARES=

MULTIPLE R-SQUARED=

VI. Variables Used in Program

Label	Meaning
A(I,J)	The $(i,j)^{th}$ element of a symmetric positive definite matrix to be inverted.
ACON	A logical indicator to show final convergence of the process.
ADIAG(I)	The i^{th} diagonal element of P'P matrix (or the standard error of estimated parameter in subroutine CLIM).
B(I)	Value of i^{th} parameter.
C(I,J)	The coefficient of j^{th} parameter of i^{th} constraint.
CHOL	Subroutine to carry out the Cholesky decomposition of a matrix.
CLIM	Subroutine to carry out the statistical calculations.
CONS	Subroutine to calculate the increments to parameters when constraints are imposed.
CONS3	An initial value added to diagonal elements of a matrix to circumvent the singularity.
CRTGAM	Critical angle of GAMMA.
DATER	Subroutine to print out the date of running the program.
DELTA(I)	Increment to i^{th} parameter.
DELM	Multiplier used in decreasing DELTA when GAMMA is critical.
DET	Determinant of matrix A.
DURBIN	Durbin-Watson D Statistic.
EIGNL	Lower bound of the smallest eigenvalue of matrix A.

Label	Meaning
EPS	Convergence criterion
EPSILON(I)	Epsilon value of i^{th} parameter when the algorithm is terminated by maximum iterations specified.
EPS1	Convergence criterion for New Algorithm to convert to Marquardt Algorithm.
EPS2	Final convergence criterion.
ETA	A constant added to diagonal elements of a matrix in the new algorithm.
F(I)	Predicted value of dependent variable at i^{th} data point.
FCODE	Subroutine to calculate the predicted values, residuals, and residual sum of squares.
FMT	Format of data input.
FPT	META symbol used to print out date of running the program.
FS	F-statistic.
G(I)	Right-hand side of normal equations.
GAMC	COSINE value of GAMMA
GAMCRT	A logical indicator showing GAMMA critical.
GAMMA	The angle between DELTA and G.
GRAPH	Subroutine plotting the residuals.
H(I,J)	The coefficient of j^{th} parameter of i^{th} linear relation.
HEAD(I)	Heading
HEADSTORE(I)	Working storage for HEAD(I).

Label	Meaning
ICONST	Program constants option indicator.
IDIAG	Diagnostics option indicator.
IGAM	Indicator showing when GAMMA is greater than 90°.
IMPRVD	A logical indicator to show improvement of residual sum of squares.
IO	Number of omitted parameters.
IOL(I)	Location of i^{th} omitted parameter.
IOLEO	A logical indicator showing no omitted parameter.
IOS	Working storage for IOL(I).
IPAGE	A page counter.
IT	Iteration counter.
K	Counter; argument of subroutine PARAOUT.
KKMAX	Maximum number of increases in LAMBDA during a single iteration to get improvement in error sum of squares.
KL	Working storage for ratio of NP and MAXWIDE.
KLINES	same as KL
KNP	same as KL
K4	Counter in a loop to avoid singularity of final P'P matrix.
LAMBDA	Lagrange multiplier imposed to achieve an improved error sum of squares.
LAMBDAO	Smallest non-zero value of LAMBDA.
LINCNT	Line counter.

Label	Meaning
LINMAX	Maximum number of lines per page.
LOOP	Counter.
LOPCL	One parameter lower confidence limit.
LSPCL	Support plane lower limit.
L1	A logical indicator showing Epsilon convergence.
L2	A logical indicator showing Gamma Epsilon convergence.
MATOUT	Subroutine to print out data and covariance matrix.
MAXIT	A logical indicator showing maximum iterations done.
MAXND	Maximum number of data points.
MAXRES	Maximum value of residuals.
MAXWIDE	Maximum number of columns of matrix printout.
METHOUT	Code of criteria of convergence.
MQT	A logical indicator of the algorithm used.
MQUADT	Subroutine to carry out the primary computations of the nonlinear algorithm.
N	The size of matrix to be factored or inverted.
NADD	Number of variables to be added.
NAME	Name of variables.
NC	Number of constraints.
NCHK	Counter for singularity correction.
NCOL	Number of the column in a matrix to be printed out.
ND	Number of data points.

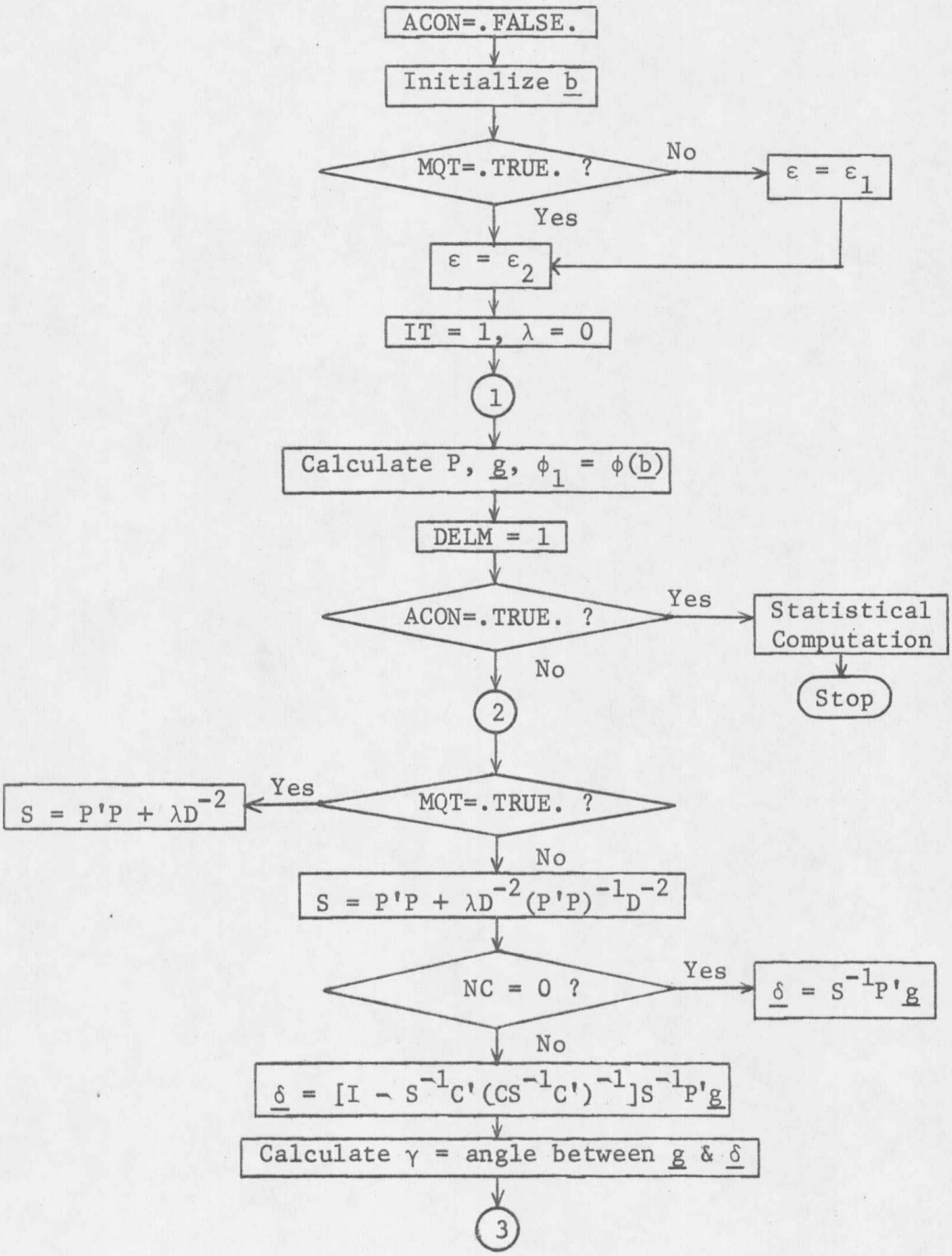
Label	Meaning
NDATE(I)	Date number.
NDC	Degrees of freedom.
NEQ	Equation counter for running several models.
NI	Maximum number of iterations.
NLOC(I)	Location of i^{th} variable.
NLR	Number of linear relations for parameters.
NP	Number of parameters.
NPAR	Initial value option indicator for parameters.
NPC	Degrees of freedom of F-statistic used to compute joint confidence region.
NPRNT	Data printout option indicator.
NPT	Auxiliary printout option indicator.
NRESD	Residual printout and graph option indicator.
NROW	Number of row in a matrix to be printed.
NTR	Number of data transformations to be performed.
NU	Increment factor of LAMBDA.
NSUBZ	Subroutine SUBZ option indicator.
NTRAN	Data transformation option indicator.
NV	Number of input variables.
OMEGA	Singularity criterion for matrix inversion or Cholesky decomposition.
OPCF	One parameter confidence interval.

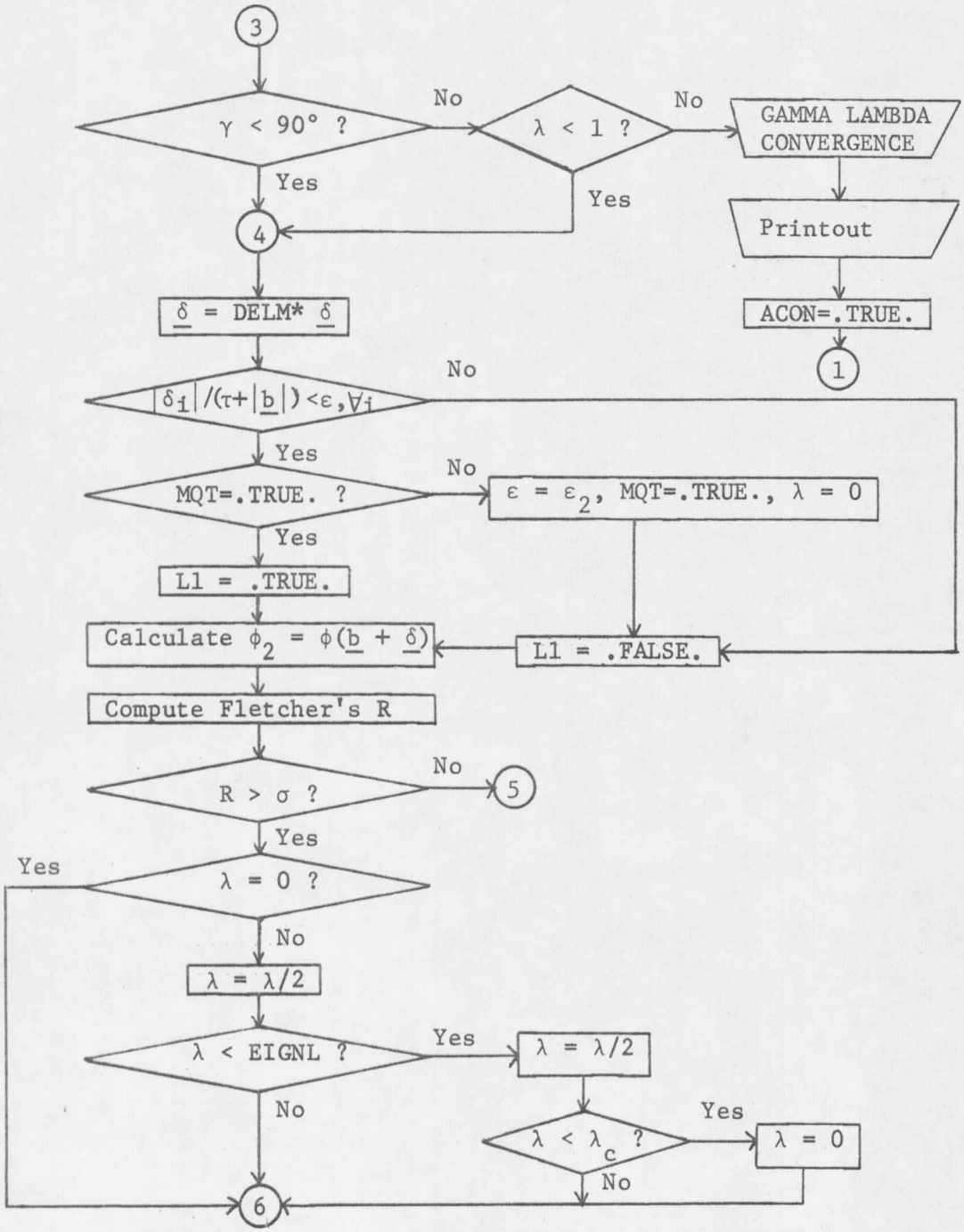
Label	Meaning
P(I)	Partial derivatives of the regression equation w.r.t. i^{th} parameter.
PAGER	Subroutine to print out page number.
PARAOUT	Subroutine to print out parameters.
PCODE	Subroutine to calculate partial derivatives.
PHI	Residual sum of squares.
PIVOT	Working storage.
PPINV(I)	Working storage for matrix A.
PRELAM	Working storage for LAMBDA.
PRNT(I,J)	Auxiliary working storage linking subroutines FCODE and PCODE.
Q(I)	Residual of i^{th} data point.
R	Fletcher's R.
RHO	Upper bound of Fletcher's R to increase LAMBDA.
RSQ	Multipler R-squared.
S	Working storage.
SAVSUM	Sum of all Y's.
SDSQ	Squared length of vector DELTA.
SE	Standard error.
SEQ	A logical indicator for a system of nonlinear equations solving option.
SE2	Standard error when temporary parameters used.
SGDEL	Inner product of vector G and DELTA.

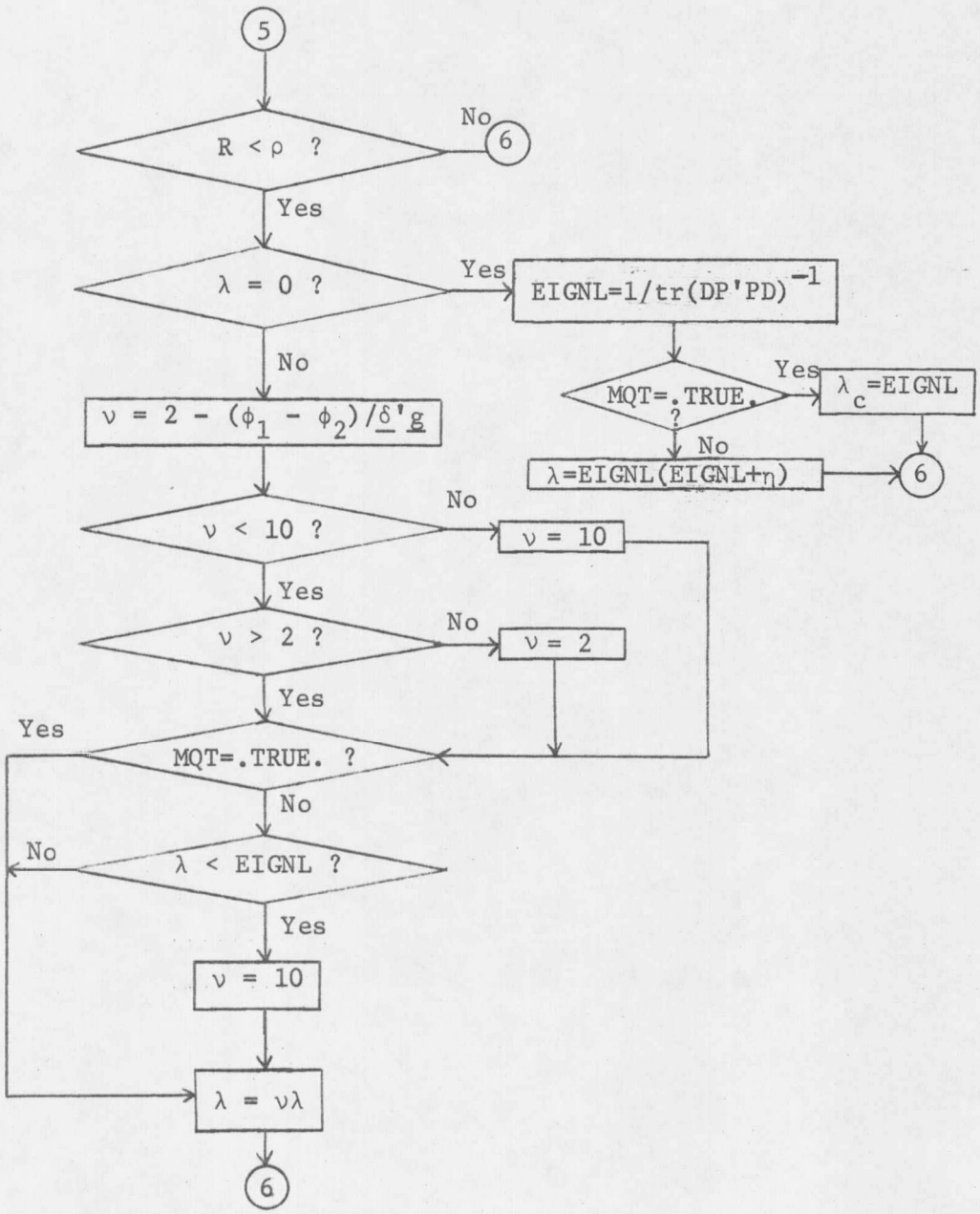
Label	Meaning
SGSQ	Squared length of vector G.
SIGMA	Lower bound of Fletcher's R to decrease LAMBDA.
SPCF	Support plane of confidence region.
SSQR	Residual sum of squares.
SUB	Integer function for using vector storage of a matrix.
SUBZ	Subroutine SUBZ.
SUM	Diagonal element after Cholesky composition.
SUMMARY	Subroutine to provide summary results when multiple solutions are obtained.
SUMQ	Sum of residuals.
SUMQ2	Working storage for SUMQ.
SUMYSQ	Squared length of vector Y.
SUM1	Predicted reduction of SSQR used to calculate Fletcher's R.
SUM3	Off-diagonal elements of matrix $\underline{\delta}'A\underline{\delta}$.
SYMINV	Subroutine to invert symmetric positive definite matrix.
T	Student t-value used in confidence interval.
TAU	Constant used in convergence test to avoid division by zero.
TEMP	Working storage for final $\|Q\|$ in subroutine GRAPH.
TEMPAR(I)	Working storage for B(I) to test new parameter increments.

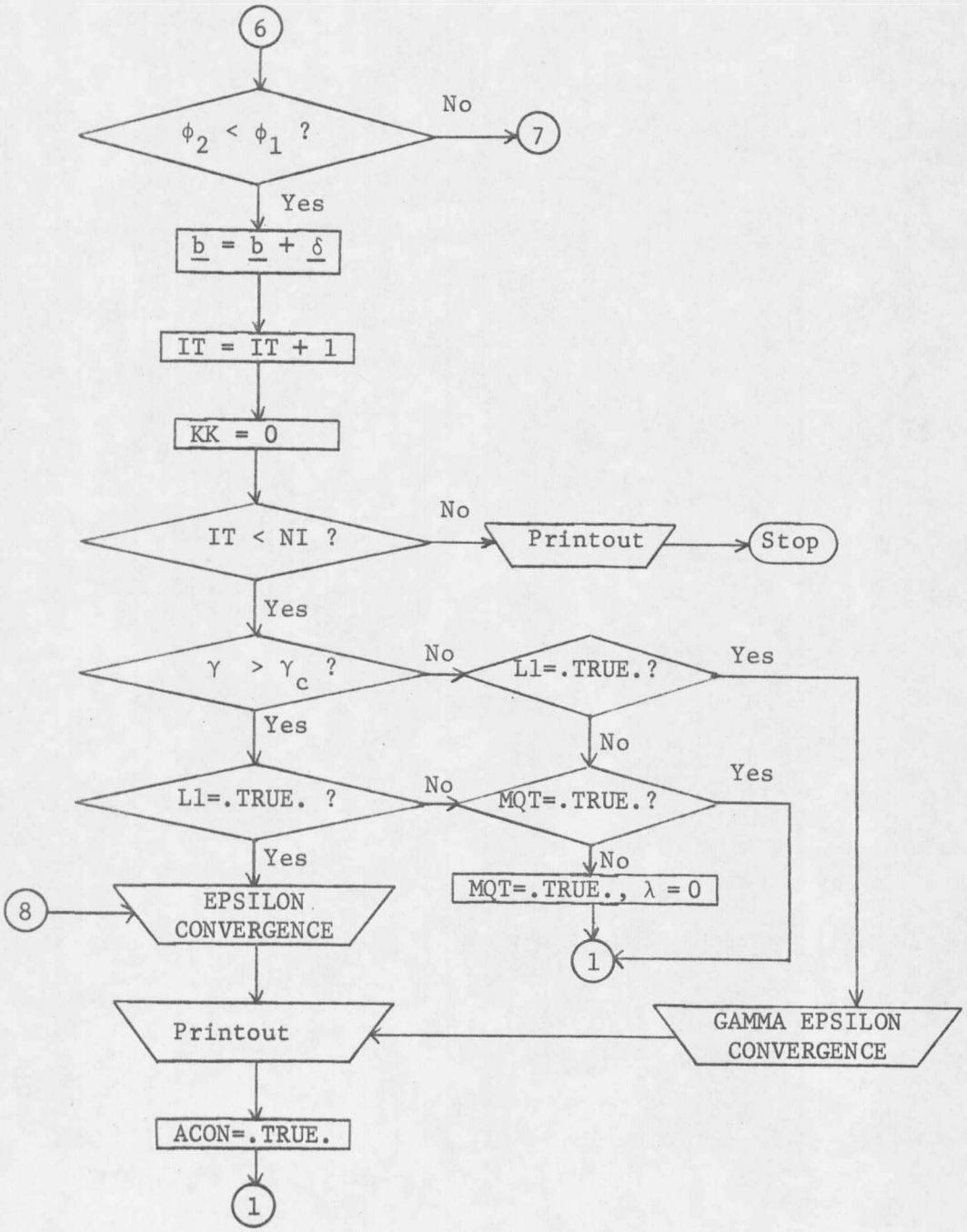
Label	Meaning
TMA	Working storage for A.
TR	Trace of matrix A^{-1}.
TRAN	Subroutine to carry out data transformation.
TRN	Data transformation coding.
UOPCL	One parameter upper confidence limit.
USPCL	Upper support plane of confidence region.
V	Working storage.
W	Working storage.
X(I,J)	The i^{th} observation of j^{th} independent variable.
XHEAD(I)	Number assigned to i^{th} parameter.
Y(I)	The i^{th} observation of dependent variable.

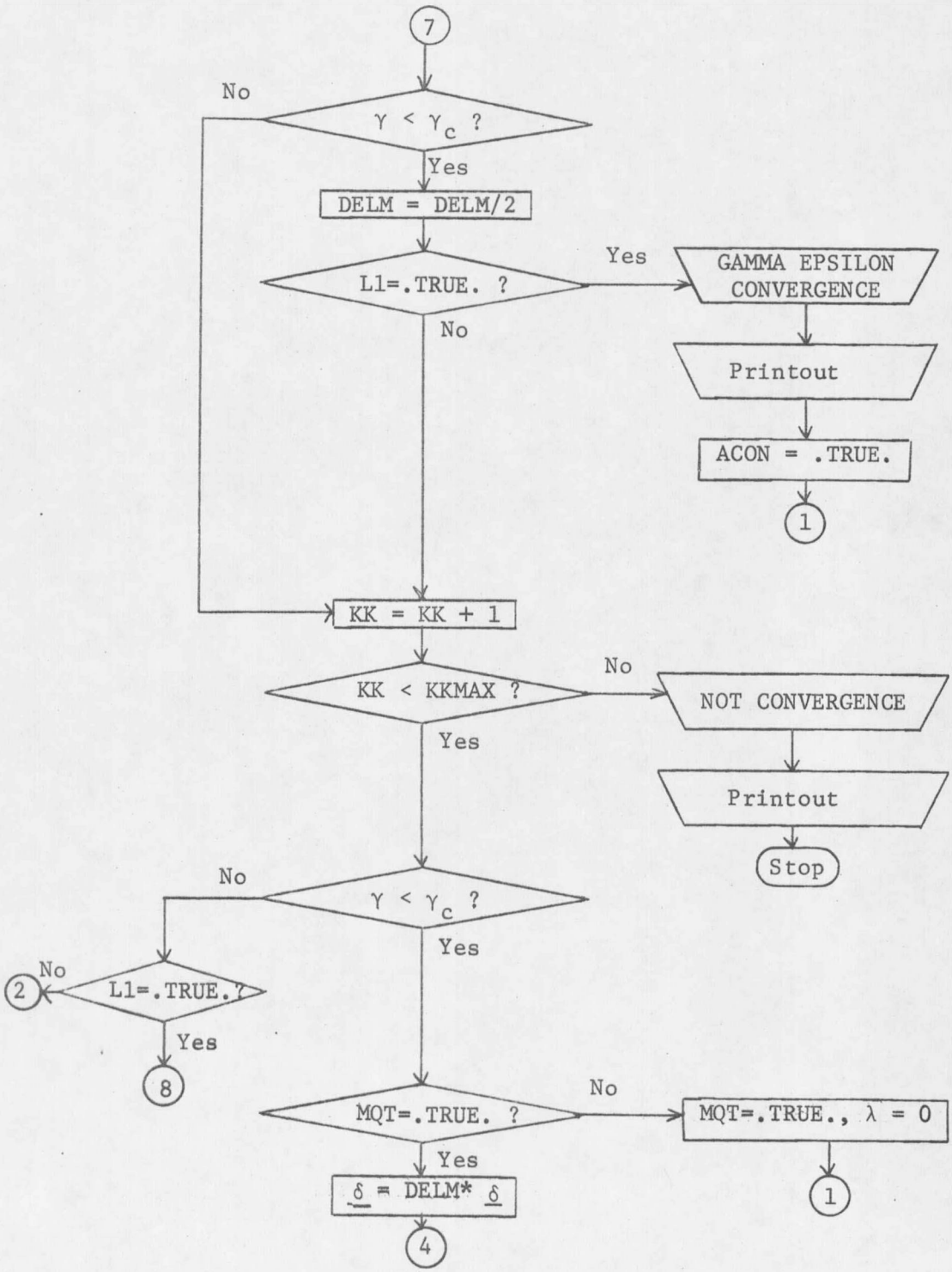
APPENDIX B: FLOW DIAGRAM




3
γ < 90° ?
No
λ < 1 ?
No
GAMMA LAMBDA CONVERGENCE
Yes
Yes
4
Printout
δ = DELM* δ
ACON=.TRUE.
1
No
|δi|/(τ+|b|)<ε,∀i
Yes
MQT=.TRUE. ?
No
ε = ε2, MQT=.TRUE., λ = 0
Yes
L1 = .TRUE.
Calculate φ2 = φ(b + δ)
L1 = .FALSE.
Compute Fletcher's R
R > σ ?
No
5
Yes
Yes
λ = 0 ?
No
λ = λ/2
λ < EIGNL ?
Yes
λ = λ/2
No
λ < λc ?
Yes
λ = 0
No
6


5
R < ρ ?
No
6
Yes
λ = 0 ?
Yes
EIGNL=1/tr(DP'PD)^-1
No
MQT=.TRUE. ?
Yes
λc=EIGNL
No
λ=EIGNL(EIGNL+η)
6
ν = 2 - (φ1 - φ2)/δ'g
ν < 10 ?
No
ν = 10
Yes
ν > 2 ?
No
ν = 2
Yes
Yes
MQT=.TRUE. ?
No
No
λ < EIGNL ?
Yes
ν = 10
λ = νλ
6


6
$\phi_2 < \phi_1$?
No
7
Yes
$\underline{b} = \underline{b} + \underline{\delta}$
IT = IT + 1
KK = 0
IT < NI ?
No
Printout
Stop
Yes
$\gamma > \gamma_c$?
No
L1=.TRUE.?
Yes
Yes
No
L1=.TRUE. ?
No
MQT=.TRUE.?
Yes
Yes
No
8
EPSILON CONVERGENCE
MQT=.TRUE., λ = 0
1
Printout
GAMMA EPSILON CONVERGENCE
ACON=.TRUE.
1


7
No
γ < γc ?
Yes
DELM = DELM/2
L1=.TRUE. ?
Yes
GAMMA EPSILON CONVERGENCE
No
Printout
ACON = .TRUE.
1
KK = KK + 1
KK < KKMAX ?
No
NOT CONVERGENCE
Yes
Printout
Stop
No
γ < γc ?
Yes
No
2
L1=.TRUE.?
Yes
8
MQT=.TRUE. ?
No
MQT=.TRUE., λ = 0
Yes
δ = DELM* δ
1
4

APPENDIX C: SAMPLE PROBLEM

The model is $y = b_1 e^{b_2 X} + b_3 e^{b_3 X}$

I. Lists of Data and Control Cards

1.	7.3	.060
2.	6.24	.029
3.	5.4	.018
4.	4.71	.090
5.	4.13	.093
6.	3.65	.073
7.	3.24	.021
8.	2.88	.045
9.	2.58	.076
10.	2.33	.096
11.	2.09	.094
12.	1.89	.053
13.	1.72	.057
14.	1.57	.096
15.	1.45	.043
16.	1.34	.065
17.	1.25	.082
18.	1.17	.091
19.	1.11	.030
20.	1.05	.026

```
F 20  3  1   1 20  2  8NLRGN SAMPLE PROBLEM
(3F10.0)
0.0
 3-1 1 2
11 2 2  -.088
14 3 3  4.
10 2 2 3
COUNTING INTEGER        RANDOM NUMBER
F 20  4  1  0  0 -1   1  0  0SAMPLE FUNCTION
6.0          -1.0        2.5          -.03
1.0           0.         1.           0.
```

II. Subroutines

```
      SUBROUTINE FCODE(B,Q,SUMQ,PHI)
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
      DOUBLE PRECISION P,A,B(25),G,ADIAG,DELTA,SSQR,TEMPAR,PRNT,PHI
      DIMENSION Q(200)
      SUMQ=PHI=0.
      DO 10 I=1,ND
      PRNT(I,1)=EXP(B(2)*X(I,1))
      PRNT(I,2)=EXP(B(4)*X(I,1))
      F(I)=B(1)*PRNT(I,1)+B(3)*PRNT(I,2)
      Q(I)=Y(I)-F(I)
      SUMQ=SUMQ+Q(I)
10    PHI=PHI+Q(I)**2
      RETURN
      END
      SUBROUTINE PCODE(I)
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25),TEMPAR(25)
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,TEMPAR,PRNT
      P(1)=PRNT(I,1)
      P(2)=B(1)*P(1)*X(I,1)
      P(3)=PRNT(I,2)
      P(4)=B(3)*P(3)*X(I,1)
      RETURN
      END
```

```
      SUBROUTINE SUBZ
C*************************************************************
C   NLRGN SAMPLE PROBLEM
C   OPTION TO TRANSFORM DATA TO DEVIATION UNITS.
C*************************************************************
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25),TEMPAR(25)
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,TEMPAR,PRNT
      READ(5,900)OPT
900   FORMAT(F10.0)
      IF(OPT.EQ.0.)GO TO 50
      DO 40 J=1,NV
      SUMX=0.
      DO 20 I=1,ND
20    SUMX=SUMX+X(I,J)
      XBAR=SUMX/ND
      DO 40 I=1,ND
40    X(I,J)=X(I,J)-XBAR
50    RETURN
      END
```

III. Output

```
NONLINEAR LEAST SQUARES      NLRGN SAMPLE PROBLEM   1300 MAR, '75 PG 1

  THERE ARE  20 OBSERVATIONS AND  3   VARIABLES ON EACH RECORD

  THE DATA WILL BE TRANSFORMED.

   20 OBSERVATIONS WILL BE PRINTED.

  FORMAT (3F10.0)


  ORIGINAL DATA

OBS      1.           2.           3.
  1   .10000E 01   .73000E 01   .60000E-01
  2   .20000E 01   .62400E 01   .29000E-01
  3   .30000E 01   .54000E 01   .18000E-01
  4   .40000E 01   .47100E 01   .90000E-01
  5   .50000E 01   .41300E 01   .93000E-01
  6   .60000E 01   .36500E 01   .73000E-01
  7   .70000E 01   .32400E 01   .21000E-01
  8   .80000E 01   .28800E 01   .45000E-01
  9   .90000E 01   .25800E 01   .76000E-01
 10   .10000E 02   .23300E 01   .96000E-01
 11   .11000E 02   .20900E 01   .94000E-01
 12   .12000E 02   .18900E 01   .53000E-01
 13   .13000E 02   .17200E 01   .57000E-01
 14   .14000E 02   .15700E 01   .96000E-01
 15   .15000E 02   .14500E 01   .43000E-01
 16   .16000E 02   .13400E 01   .65000E-01
 17   .17000E 02   .12500E 01   .82000E-01
 18   .18000E 02   .11700E 01   .91000E-01
 19   .19000E 02   .11100E 01   .30000E-01
 20   .20000E 02   .10500E 01   .26000E-01
```

NONLINEAR LEAST SQUARES NLRGN SAMPLE PROBLEM 1300 MAR 28,'75 PG 2

TRANSFORMED DATA

OBS	1.	2.
1	.10000E 01	.74520E 01
2	.20000E 01	.62680E 01
3	.30000E 01	.53840E 01
4	.40000E 01	.49820E 01
5	.50000E 01	.44140E 01
6	.60000E 01	.38540E 01
7	.70000E 01	.32360E 01
8	.80000E 01	.29720E 01
9	.90000E 01	.27960E 01
10	.10000E 02	.26260E 01
11	.11000E 02	.23780E 01
12	.12000E 02	.20140E 01
13	.13000E 02	.18600E 01
14	.14000E 02	.18660E 01
15	.15000E 02	.15340E 01
16	.16000E 02	.15120E 01
17	.17000E 02	.14900E 01
18	.18000E 02	.14460E 01
19	.19000E 02	.11420E 01
20	.20000E 02	.10660E 01

NONLINEAR LEAST SQUARES SAMPLE FUNCTION 1300 MAR 28,'75 PG 3

EQUATION NO. 1
MAXIMUM ITERATIONS= 20
NUMBER OF PARAMETERS= 4

CONSTRAINT(S)= 1

NEW ALGORITHM

THE DEPENDENT VARIABLE IS RAMDOM NUMBER

EQUATION VARIABLES

1. ..... COUNTING INTEGER

INITIAL PARAMETERS

PARAMETER(1)= .600000E 01
PARAMETER(2)= -.100000E 01
PARAMETER(3)= .250000E 01
PARAMETER(4)= -.300000E-01

NO OMITTED PARAMETERS

PROGRAM CONSTANTS

KKMAX= 50	FS= 4.0000	T= 2.0000
EPSILON1= .100E-01	TAU= .100E-02	LAMBDA= .000E 00
RHO= .25	CRITICAL GAMMA= 30.00	OMEGA= .100E-59
SIGMA= .75	EPSILON2= .100E-04	ETA= .100E-05

CONSTRAINT(S)

1.0000 .0000 1.0000 .0000

INITIAL ERROR SUM OF SQUARES=.44959074E 02
INITIAL STANDARD ERROR OF ESTIMATE=.16262388E 01

DIAGNOSTICS FOR ITERATION 1 ROUND 1

LAMBDA= .000E 00 LAMBDAO= .000E 00 GAMMA= .671E 02 FLETCHER'S R= .844E 00 SSQR(2)=.75801407E 01

	1.	2.	3.	4.
DELTA	-.31796E 01	.56217E 00	.31796E 01	-.81171E-01

SUMMARY DIAGNOSTICS FOR ITERATION NUMBER 1

	1.	2.	3.	4.
DELTA	-.31796E 01	.56217E 00	.31796E 01	-.81171E-01
PARAMETER	.28204E 01	-.43783E 00	.56796E 01	-.11117E 00

SSQR(1)= 4.49590742E 01 SSQR(2)= 7.58014072E 00 SUMQ= 1.208490E 01 LAMBDAO=.000000E 00

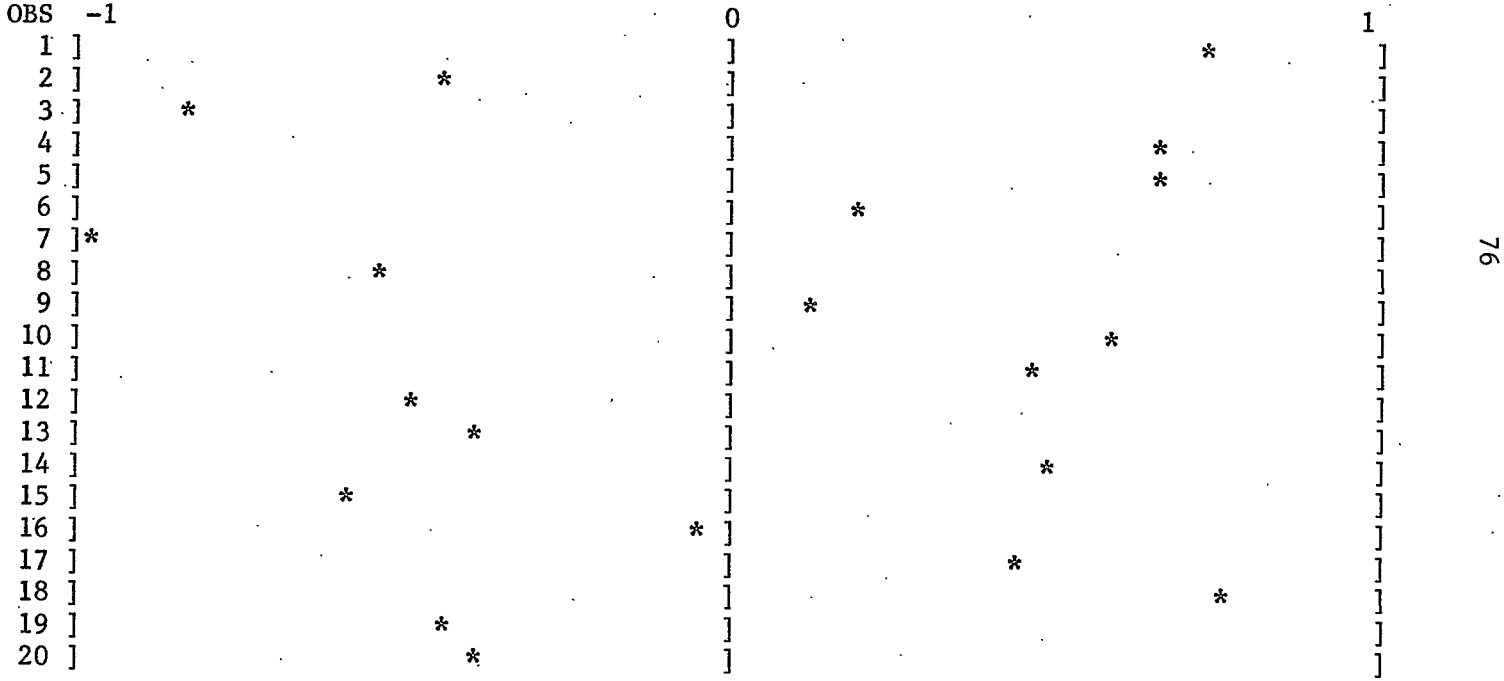
GAMMA= 67.0758 LAMBDA= .000000E 00 FLETCHER'S R= 8.435900E-01 SE2= 6.677504E-01

EPSILON CONVERGENCE

TOTAL NUMBER OF ITERATIONS WAS 7

OBS	PRED	RESIDUAL		
.745200E 01	.731879E 01	.133211E 00	.79928E 00	.93992E 00
.626800E 01	.634313E 01	-.751362E-01	.63885E 00	.88345E 00
.538400E 01	.553367E 01	-.149668E 00	.51062E 00	.83037E 00
.498200E 01	.485882E 01	.123184E 00	.40813E 00	.78048E 00
.441400E 01	.429323E 01	.120765E 00	.32621E 00	.73359E 00
.385400E 01	.381656E 01	.374365E-01	.26074E 00	.68952E 00
.323600E 01	.341243E 01	-.176435E 00	.20840E 00	.64809E 00
.297200E 01	.306768E 01	-.956764E-01	.16657E 00	.60916E 00
.279600E 01	.277168E 01	.243225E-01	.13314E 00	.57256E 00
.262600E 01	.251588E 01	.110117E 00	.10641E 00	.53816E 00
.237800E 01	.229338E 01	.846233E-01	.85055E-01	.50583E 00
.201400E 01	.209856E 01	-.845585E-01	.67983E-01	.47544E 00
.186000E 01	.192689E 01	-.668936E-01	.54338E-01	.44687E 00
.186600E 01	.177469E 01	.913076E-01	.43431E-01	.42003E 00
.153400E 01	.163895E 01	-.104946E 00	.34714E-01	.39479E 00
.151200E 01	.151720E 01	-.520229E-02	.27746E-01	.37107E 00
.149000E 01	.140744E 01	.825548E-01	.22177E-01	.34878E 00
.144600E 01	.130802E 01	.137982E 00	.17726E-01	.32782E 00
.114200E 01	.121755E 01	-.755472E-01	.14168E-01	.30813E 00
.106600E 01	.113490E 01	-.688953E-01	.11324E-01	.28962E 00

PLOT OF RESIDUALS
ABSOLUTE VALUE OF LARGEST RESIDUAL = 1



TOTAL SUM OF SQUARES= .6327047729E 02

RESIDUAL SUM OF SQUARES= .2044240466E 00

SUM OF RESIDUALS= .4254531860E-01

STANDARD ERROR OF ESTIMATE= .1096583009E 00

THE MULTIPLE R-SQUARED= .997

DEGREES OF FREEDOM= 17.

DURBIN-WATSON D STATISTIC= .1780E 01

SCALED P TRANSPOSE P

	1.	2.	3.	4.
1.	.10000E 01	.78246E 00	.85897E 00	.40424E 00
2.		.10000E 01	.96077E 00	.76504E 00
3.			.10000E 01	.80316E 00
4.				.10000E 01

PARAMETER COVARIANCE MATRIX/(SIGMA SQR)

	1.	2.	3.	4.
1.	.17115E 03	.52152E 01	-.17115E 03	.21876E 01
2.		.16371E 00	-.52152E 01	.65661E-01
3.			.17115E 03	-.21876E 01
4.				.28313E-01

PARAMETER CORRELATION MATRIX

	1.	2.	3.	4.
1.	.10000E 01	.98524E 00	-.10000E 01	.99375E 00
2.	*	.10000E 01	-.98524E 00	.96445E 00
3.			.10000E 01	-.99375E 00
4.				.10000E 01

APPROXIMATE 95% LINEAR CONFIDENCE LIMITS

	PARAMETER	STANDARD	ONE-PARAMETER		SUPPORT PLANE	
PARAMETER	ESTIMATE	ERROR	LOWER	UPPER	LOWER	UPPER
1.	.47678E 01	.14346E 01	.18986E 01	.76370E 01	-.16480E 01	.11184E 02
2.	-.22404E 00	.44369E-01	-.31278E 00	-.13530E 00	-.42246E 00	-.25618E-01
3.	.37322E 01	.14346E 01	.86296E 00	.66014E 01	-.26836E 01	.10148E 02
4.	-.61960E-01	.18452E-01	-.98863E-01	-.25057E-01	-.14448E 00	.20558E-01

SUMMARY OF RESULTS

EQUATION NO. 1 EPSILON CONVERGENCE

	1.	2.	3.	4.
PARAMETER	.47678E 01	-.22404E 00	.37322E 01	-.61960E-01
STD ERROR	.14346E 01	.44369E-01	.14346E 01	.18452E-01

STANDARD ERROR OF THE ESTIMATE= .10966E 00 RESIDUAL SUM OF SQUARES= .20442E 00

MULTIPLE R-SQUARED= .997

APPENDIX D: COMPUTER PROGRAM

```
C*********************************************************************
C********************  NLRGN  ****************************************
C  A NON-LINEAR LEAST SQUARES REGRESSION ALGORITHM                     *
C*********************************************************************
C FILE 5 IS PARAMETER INPUT.                                          *
C FILE 6 IS OUTPUT TO LP.                                             *
C FILE 7 SAVES THE RESULTS OF EACH EQUATION FOR THE                   *
C SUMMARY OUTPUT.                                                     *
C FILE 15 IS DATA INPUT.                                              *
C*********************************************************************
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25),TEMPAR(25),PPINV(650)
     $,C(5,25),H(5,25)
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,TEMPAR,PRNT
     $,PPINV
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
     $,NPRNT,MAXWIDE,NAME(25,5)
      INTEGER XHEAD,HEAD,HEADSTOR(11)
      DATA XHEAD/' 1. ',' 2. ',' 3. ',' 4. ',' 5. ',' 6. ',' 7. ',
     1' 8. ',' 9. ',' 10.',' 11.',' 12.',' 13.',' 14.',' 15.'
     2,' 16.',' 17.',' 18.',' 19.',' 20.' ,' 21.',' 22.',' 23.',
     3' 24.',' 25.'/
C     4' 32.',' 33.',' 34.',' 35.',' 36.',' 37.',' 38.',' 39.',' 40.',
C     5' 41.',' 42.',' 43.',' 44.',' 45.',' 46.',' 47.',
C     6' 48.',' 49.',' 50.'/
      DIMENSION FMT(20)
      DATA MAXND,MAXNV,MAXNP/200,25,25/
      REAL LAMBDA
      LOGICAL IOLEO,MQT,SEQ
C*********************************************************
C LINMAX IS THE NUMBER OF LINES PER PAGE OF OUTPUT.
      LINMAX=40
C MAXWIDE IS THE NUMBER OF COLUMNS OF NUMBERS THAT
C WILL FIT ACROSS ONE PAGE.
C*********************************************************
      IPAGE=0
      CALL DATER
C*************************************************************
C READ THE INITIAL CARD
C*************************************************************
      READ(5,901,END=130)SEQ,ND,NV,NSUBZ,NTRAN,NPRNT,NPT,MAXWIDE,HEAD
```

```
901   FORMAT(L1,7I3,11A4)
      IF(MAXWIDE.EQ.0)MAXWIDE=10
      CALL PAGER
      IF(SEQ)WRITE(6,900)ND,NV;LINCNT=LINCNT+3;GO TO 45
900   FORMAT('   SYSTEM EQUATIONS SOLVING',/,'   THERE ARE ',I3,
     $' EQUATIONS AND ',I2,' VARIABLES TO BE SOLVED',/)
      WRITE(6,902)ND,NV
902   FORMAT('   THERE ARE ',I3,' OBSERVATIONS AND ',
     $I2,'   VARIABLES ON EACH RECORD',/)
      LINCNT=LINCNT+2
      IF(ND.LE.1)CALL EXIT
      IF(NTRAN.EQ.1)WRITE(6,905);GO TO 37
905   FORMAT('   THE DATA WILL BE TRANSFORMED.'/)
37    IF(NPRNT.LE.0)NPRNT=1
      WRITE(6,906)NPRNT
      LINCNT=LINCNT+4
906   FORMAT('   ',I3,' OBSERVATIONS WILL BE PRINTED.',/)
      IF(ND.GT.MAXND.OR.NV.GT.MAXNV) GO TO 80
C*************************************************************
C READ IN THE FORMAT FOR DATA INPUT.
C*************************************************************
      READ(5,903,END=130) FMT
903   FORMAT(20A4)
      WRITE(6,904) FMT
904   FORMAT('   FORMAT ',20A4,/)
      LINCNT=LINCNT+2
C*************************************************************
C READ IN THE DATA
C*************************************************************
      DO 40 I=1,ND
40    READ(15,FMT,END=120)(X(I,J),J=1,NV)
C*************************************************************
C     WRITE OUT DATA.
C*************************************************************
43    K=1
      CALL MATOUT(NPRNT,NV,K)
C*************************************************************
C TRANSFORM THE DATA IF REQUIRED
C*************************************************************
      IF (NSUBZ .NE. 0) CALL SUBZ
      IF(NTRAN.EQ.1) CALL TRAN
C*************************************************
C   READ VARIABLE  NAMES
C*************************************************
45    READ(5,965)((NAME(I,J),J=1,5),I=1,NV)
965   FORMAT(20A4)
```

```
      IF(SEQ)GO TO 50
470   DO 35 I=1,ND
35    Y(I)=X(I,NV)
C*************************************************************
C STORE THE MAIN HEADING FOR USE IN THE SUMMARY.
      DO 5 I=1,11
5     HEADSTOR(I)=HEAD(I)
C*************************************************************
      NV=NV-1
50    NEQ=0.0
C*************************************************************
C READ AN EQUATION CARD AND BEGIN  CALCULATIONS
C*************************************************************
10    READ(5,907,END=200)MQT,NI,NP,NC,NLR,IO,IDIAG,
     $NRESD,NPAR,ICONST,HEAD
907   FORMAT(L1,9I3,11A4)
      CALL PAGER
      NEQ=NEQ+1
78    WRITE(6,908)NEQ,NI,NP
908   FORMAT('   EQUATION NO. ',I2,/,'   MAXIMUM ITERATIONS= ',I3,
     $     /,'   NUMBER OF PARAMETERS= ',I2,/)
      LINCNT=LINCNT+4
      IF(NC.EQ.0)GO TO 87
      WRITE(6,986)NC
986   FORMAT('   CONSTRAINT(S)=',I2,/)
      LINCNT=LINCNT+2
87    IF(NLR.EQ.0)GO TO 88
      WRITE(6,987)NLR
987   FORMAT('   LINEAR RELATION(S)=',I2,/)
      LINCNT=LINCNT+2
88    IF(MQT)GO TO 55
      WRITE(6,970)
970   FORMAT('   NEW ALGORITHM'/)
      GO TO 85
55    WRITE(6,985)
985   FORMAT('   MARQUARDT ALGORITHM'/)
85    LINCNT=LINCNT+2
      IF(SEQ)GO TO 60
      WRITE(6,990)(NAME(NV+1,J),J=1,5)
990   FORMAT('   THE DEPENDENT VARIABLE IS ', 5A4,/)
      LINCNT=LINCNT+2
C*************************************************************
C     ASSIGN VARIABLE  NAMES .
C*************************************************************
60    WRITE(6,915)
915   FORMAT('   EQUATION VARIABLES' //)
```

```
      LINCNT =LINCNT+3
      DO 400 I=1,NV
      WRITE(6,966)XHEAD(I),(NAME(I ,J),J=1,5)
966   FORMAT(3X,A4,' ..... ',5A4)
400   LINCNT=LINCNT+1
C*************************************************************
C READ IN INITIAL PARAMETERS.
C*************************************************************
      DO 72 J=1,NP
72    B(J)=0.0
      IF(LINMAX.LT.LINCNT+NP+2) CALL PAGER
      IF(NPAR.EQ.1) GO TO 71
      READ(5,912,END=130)(B(J),J=1,NP)
912   FORMAT(8F10.0)
      GO TO 73
71    BACKSPACE 7
      READ(7)I,I,IOL,B,ADIAG
      WRITE(6,969)
969   FORMAT(/,'   THE PARAMETER VALUES FROM THE PREVIOUS'
     $   ' EQUATION WILL BE USED AS STARTING VALUES.')
      LINCNT=LINCNT+2
73    WRITE(6,940)
940   FORMAT(/'INITIAL PARAMETERS'/)
      WRITE(6,950) (I,B(I),I=1,NP)
950   FORMAT(' PARAMETER(',I2,')=',1E16.6)
      LINCNT=LINCNT+NP+3
C*************************************************************
C     OMITTED PARAMETERS
C*************************************************************
95    IOLE0=IO.LE.0
      IF(IOLE0) GO TO 240
      READ(5,971,END=130)(IOL(J),J=1,IO)
971   FORMAT(39I2)
      IF(LINMAX.LT.LINCNT+2) CALL PAGER
      WRITE(6,930) IO,(IOL(I),I=1,IO)
930   FORMAT(/,I2,'    OMITTED PARAMETER(S), NUMBER(S)',20I3)
      LINCNT=LINCNT+2
      GO TO 160
240   IF(LINCNT+2.GE.LINMAX) CALL PAGER
      WRITE(6,150);LINCNT=LINCNT+2
150   FORMAT(/,'   NO OMITTED PARAMETERS')
C*************************************************
C PROGRAM CONSTANTS ARE ESTABLISHED BELOW.
C*************************************************
160   GO TO (15,16) ICONST+1
16    READ(5,20,END=130)  KKMAX,FS,T,EPS1,EPS2,TAU,LAMBDA,RHO,SIGMA,
```

```
     1 CRTGAM,OMEGA,ETA
20    FORMAT(I3,2F6.0,3E8.0,4F6.0,2E8.0)
      IF(KKMAX.LE.0) KKMAX=50
      IF(FS.LE.0.) FS=4.0
      IF(T.LE.0.) T=2.0
      IF(EPS1.LE.0.)EPS1=1.0E-02
      IF(EPS2.LE.0.) EPS2=1.0E-05
      IF(TAU.LE.0.) TAU=1.0E-03
      IF(LAMBDA.LE.0.) LAMBDA=00.0
      IF(RHO.LE.0.) RHO=0.25
      IF (SIGMA.EQ.0) SIGMA=0.75
      IF(CRTGAM.LE.0.) CRTGAM=30.
      IF(OMEGA.LE.0.) OMEGA=1.0E-60
      IF(ETA.LE.0.)ETA=1.0E-06
      GO TO 115
15    KKMAX=50
      FS=4.0
      T=2.0
      EPS1=1.0E-02
      EPS2=1.0E-05
      TAU=1.0E-03
      LAMBDA=00.0
      RHO=0.25
      SIGMA=0.75
      CRTGAM=30.0
      OMEGA=1.0E-60
      ETA=1.0E-06
115   IF(LINCNT+6.GT.LINMAX) CALL PAGER
      WRITE(6,920) KKMAX,FS,T,EPS1,TAU,LAMBDA,RHO,CRTGAM,OMEGA,SIGMA
     $,EPS2,ETA
920   FORMAT(/,'   PROGRAM CONSTANTS',/,7H KKMAX=,I4,13X,3HFS=,F8.4,15X
     1 ,2HT=,F8.4,/,10H EPSILON1=,E10.3,4X,4HTAU=,E10.3,12X,7HLAMBDA=,
     2 E10.3,/,' RHO=',F5.2,14X,15HCRITICAL GAMMA=,F6.2,5X,6HOMEGA=,
     3 E10.3,/,' SIGMA=',F5.2,12X,'EPSILON2=',E10.3,7X,'ETA=',E10.3)
      LINCNT=LINCNT+5
      IF(NC.EQ.0)GO TO 124
      KL=NP/8+1
      IF(LINCNT+3+KL*NC.GT.LINMAX)   CALL PAGER
      WRITE(6,800)
800   FORMAT(/,'   CONSTRAINT(S)',/)
      DO 110 I=1,NC
      READ(5,910,END=130)(C(I,J),J=1,NP)
110   WRITE(6,911)(C(I,J),J=1,NP)
910   FORMAT(8F10.0)
      LINCNT=LINCNT+KL*NC+3
124   IF(NLR.EQ.0)GO TO 125
```

```
      KL=NP/8+1
      IF(LINCNT+3+KL*NLR.GT.LINMAX) CALL PAGER
      WRITE(6,810)
810   FORMAT(/,'   LINEAR RELATION(S)',/)
      DO 90 I=1,NLR
      READ(5,910,END=130)(H(I,J),J=1,NP)
90    WRITE(6,911)(H(I,J),J=1,NP)
      LINCNT=LINCNT+3+KL*NLR
911   FORMAT(8F10.4)
C*************************************************************
C CALL THE MAIN CALCULATION ROUTINE.
125   CALL MQUADT
C*************************************************************
      GO TO 10
80    WRITE(6,980)MAXND,MAXNP
980   FORMAT('1TOO MANY OBSERVATIONS OR TOO MANY VARIABLES.',/
     $     ' MAXIMUM OBSERVATIONS= ',I3,/
     $     ' MAXIMUM VARIABLES= ',I2)
      GO TO 100
130   WRITE(6,135)
135   FORMAT('1UNEXPECTED  END OF CONTROL FILE. RUN TERMINATED.')
      GO TO 100
120   WRITE(6,945) I-1
945   FORMAT('1UNEXPECTED  END OF DATA FILE AFTER RECORD ',I3)
      GO TO 100
200   IF(SEQ)GO TO 100
      DO 205 I=1,11
205   HEAD(I)=HEADSTOR(I)
      CALL SUMMARY
100   CALL EXIT
      END
      SUBROUTINE MQUADT
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLE0,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25),TEMPAR(25),PPINV(650)
     $,C(5,25),H(5,25)
      COMMON/LOGIC/ACON
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,TEMPAR,PRNT
      DOUBLE PRECISION  SDSQ,SGSQ,SGDEL,DELM,SUM1,SUM3
      DOUBLE PRECISION PPINV ,GAMC
      REAL LAMBDA,NU,LAMBDAO
      LOGICAL L1,L2,MAXIT,GAMCRT,ACON,IOLE0,IMPRVD,MQT,SEQ
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
```

```
     $,NPRNT,MAXWIDE
      INTEGER XHEAD,HEAD ,SUB
C*****************************************************************
C     THIS FUNCTION ASSIGNS STORAGE VECTOR LOCATIONS TO
C     TRIANGULAR MATRIX ELEMENTS, COLUMN BY COLUMN.
      SUB(I,J)=I+(J*J-J)/2
C*****************************************************************
      NDC=ND-NP+IO+NC
      IF(SEQ)NDC=ND
      MAXIT=.FALSE.
      ACON=.FALSE.
      EPS=EPS1
      IF(MQT)EPS=EPS2
      IT=1
      LAMBDAO=0.0
      KLINES=NP/(MAXWIDE+1)
      KLINES=KLINES+1
      CALL FCODE(B,Q,SUMQ,SSQR(1))
      SE=DSQRT(SSQR(1)/NDC)
      IF(LINCNT+7.GT.LINMAX) CALL PAGER
      LINCNT=LINCNT+7
      IF(SEQ)WRITE(6,20)SSQR(1);GO TO 35
20    FORMAT(///,'INITIAL ERROR SUM OF SQUARES=',E13.8,///)
      WRITE(6,40)SSQR(1),SE
40    FORMAT(///,' INITIAL ERROR SUM OF SQUARES=',E13.8,/,
     $' INITIAL STANDARD ERROR OF ESTIMATE=',E13.8,//)
      SUMYSQ=SAVSUM=0.
      DO 30 I=1,ND
      SAVSUM=SAVSUM+Y(I)
30    SUMYSQ=SUMYSQ+Y(I)**2
35    KK=0
C*****************************************************
C START AN ITERATION BY CALCULATING THE PARTIALS.
C*****************************************************
160   DO 175 I=1,NP
      G(I)=0.
      DO 175 J=1,NP
175   A(I,J)=0.
      DO 190 K=1,ND
      CALL PCODE (K)
      IF(IOLE0) GO TO 180
C*****************************************************
C TAKE CARE OF OMITTED PARAMETERS.
C*****************************************************
      DO 170 I=1,IO
      IOS=IOL(I)
```

```
170   P(IOS)=0.
C*****************************************************
C CALCULATE THE LOWER TRIANGLE OF P'P WHICH AFTER SCALING IS RETAINED
C THROUGHOUT THE ITERATION.
C*****************************************************
180   DO 190 I=1,NP
      G(I)=G(I)+P( I)*Q(K)
      DO 190 J=1,I
190   A(I,J)=A(I,J)+P(I)*P(J)
195   IF(IOLE0) GO TO 220
C*****************************************************
C TAKE CARE OF THE OMITTED PARAMETERS.
C*****************************************************
      DO 210 I=1,IO
      IOS=IOL(I)
      DO 200 J=1,IOS-1
      A(IOS,J)=0.
200   CONTINUE
      A(IOS,IOS)=1.0
210   CONTINUE
C*****************************************************
C IF CONVERGENCE IS COMPLETE GO CALL THE STATISTICAL
C SUBROUTINE 'CLIM'.
C*****************************************************
220   IF(ACON) GO TO 740
C*****************************************************
C PREPARE TO CALCULATE PARAMETER IMPROVEMENTS.
C (LINEAR REGRESSION ON RESIDUALS)
C BEGIN BY SCALING P'P
C*****************************************************
      DO 230 I=1,NP
      IF(A(I,I).EQ.0.0)WRITE(6,910)I,I;RETURN
910   FORMAT('0ELEMENT (',I2,',',I2,') OF P''P IS ZERO')
 230  ADIAG(I)=DSQRT(A(I,I))
      DELM=1.0
      G(1)=G(1)/ADIAG(1)
      DO 233 I=2,NP
      G(I)=G(I)/ADIAG(I)
      DO 233 J=1,I-1
 233  A(I,J)=A(I,J)/(ADIAG(I)*ADIAG(J))
C*****************************************************
C IF LAMBDA IS ZERO SKIP TO STATEMENT LABEL 275.
C*****************************************************
      IF(LAMBDA.EQ.0.0) GO TO 275
215   IF(MQT)GO TO 237
```

```
C*****************************************************************
C  CALCULATE (P'P+ETA*I) INVERSE.
C*****************************************************************
      A(1,1)=1.0+ETA
      DO 205 I=2,NP
      A(I,I)=1.0+ETA
      DO 205 J=1,I-1
205   A(J,I)=A(I,J)
      CALL SYMINV(&206,NP)
      GO TO 207
206   WRITE(6,961);RETURN
961   FORMAT('   SINGULAR MATRIX IN SYMINV AT 206')
207   DO 255 I=1,NP
      DO 255 J=1,I
255   PPINV(SUB(J,I))=A(J,I)
C*****************************************************************
C CALCULATE THE CHOLESKY DECOMPOSITION OF THE TRANSFORMED
C P'P MATRIX.
C*****************************************************************
235   NCHK=0
236   IF(MQT)GO TO 237
      A(1,1)=1.0+PPINV(SUB(1,1))*LAMBDA
      DO 260 I=2,NP
      A(I,I)=1.0+PPINV(SUB(I,I))*LAMBDA
      DO 260 J=1,I-1
260   A(J,I)=A(I,J)+PPINV(SUB(J,I))*LAMBDA
      GO TO 239
C*****************************************************************
C   MARQUARDT ALGORITHM IF PREFERRED.
C*****************************************************************
237   A(1,1)=1.+LAMBDA
      DO 238 I=2,NP
      A(I,I)=1.+LAMBDA
      DO 238 J=1,I-1
238   A(J,I)=A(I,J)
C*****************************************************************
C   INCREMENT COMPUTATION IF CONSTRAINTS INVOLVED.
C*****************************************************************
239   IF(NC.NE.0) CALL CONS(&286,&225,&999)
      CALL CHOL(&225,NP)
      GO TO 250
225   LAMBDA=LAMBDA*10.
      NCHK=NCHK+1
      WRITE(6,962)
      LINCNT=LINCNT+1
      IF(NCHK.GT.9)RETURN
```

```
962   FORMAT('   SINGULAR MATRIX IN CHOL AT 225')
      GO TO 236
C*****************************************************************
C CALCULATE THE CHOLESKY DECOMPOSITION  OF P'P WHEN LAMBDA
C IS ZERO.
C*****************************************************************
275   A(1,1)=1.0
      DO 240 I=2,NP
      A(I,I)=1.0
      DO 240 J=1,I-1
240   A(J,I)=A(I,J)
C*************************************************************
C   INCREMENT COMPUTATION IF CONSTRAINTS INVOLVED.
C*************************************************************
      IF(NC.NE.0) CALL CONS(&286,&202,&999)
201   CALL CHOL(&202,NP)
      GO TO 250
202   LAMBDAO=10**(-15)*(10**(-15)+ETA)
      IF(MQT)LAMBDAO=1.0E-15
      LAMBDA=LAMBDAO
      WRITE(6,930)
      LINCNT=LINCNT+1
930   FORMAT('   SINGULAR MATRIX IN CHOL AT 202')
      IF(MQT)GO TO 237
      GO TO 215
C*************************************************************
C UPPER TRIANGLE OF A NOW CONTAINS THE L' OF THE CHOLESKY
C DECOMPOSITION.
C*************************************************************
250   DELTA(1)=G(1)/A(1,1)
      DO 270 I=2,NP
      SUM1=0.0
      DO 265 J=1,I-1
265   SUM1=SUM1+A(J,I)*DELTA(J)
270   DELTA(I)=(G(I)-SUM1)/A(I,I)
      DELTA(NP)=DELTA(NP)/A(NP,NP)
      DO 285 I=NP-1,1,-1
      SUM1=0.0
      DO 280 J=I+1,NP
280   SUM1=SUM1+A(I,J)*DELTA(J)
285   DELTA(I)=(DELTA(I)-SUM1)/A(I,I)
C*************************************************************
C INCREMENTS TO PARAMETER VALUES HAVE NOW BEEN COMPLETED.
C*************************************************************
C*************************************************************
C COMPUTATION OF EXIT CRITERIA USING THE ANGLE GAMMA.
```

```
C MAINLY USED TO PREVENT EXCESSIVE ITERATING  WHEN THE
C ONLY CHANGES ARE FROM ROUNDING ERROR.
C*********************************************************
286    SGSQ=0.0
       SDSQ=0.0
       SGDEL=0.0
       DO 290 I=1,NP
       SGSQ=SGSQ+G(I)**2
       SDSQ=SDSQ+DELTA(I)**2
290    SGDEL=SGDEL+G(I)*DELTA(I)
C CALCULATE INCREASE IN SSE FOR LINEAR REGRESSION.
       SUM3=0.0
       DO 292 I=2,NP
       DO 292 J=1,I-1
 292   SUM3=SUM3+A(I,J)*DELTA(I)*DELTA(J)
       SUM1=2.0*(SGDEL-SUM3)-SDSQ
       IF(SUM1.LE.0.0)GOTO 900
C SUM1 ABOVE USED LATER TO GET FLETCHER'S R
C NOW CHANGE SCALED DELTA INTO SAME UNITS AS PARAMETERS.
       DO 293 I=1,NP
 293   DELTA(I)=DELTA(I)/ADIAG(I)
       IF(NP-IO.EQ.1) GO TO 320
       GAMC=SGDEL/DSQRT(SDSQ*SGSQ)
       IGAM=1
       IF(GAMC.GT.0.) GO TO 300
       GAMC=DABS(GAMC)
       IGAM=2
300    GAMMA=DARCOS(GAMC)*57.2957795
       IF(IGAM.EQ.1) GO TO 325
       GAMMA=180.-GAMMA
       IF(LAMBDA.LT.1.0) GO TO 325
C*********************************************************
C IF  GAMMA  HAS BECOME NEGATIVE AND LAMBDA IS ONE OR
C GREATER THEN THE CALCULATIONS HAVE GONE BAD.
C CONCLUDE CALCULATIONS VIA THE GAMMA LAMBDA CRITERIA.
C*********************************************************
       CALL PAGER
       WRITE(6,310) LAMBDA,GAMMA
310    FORMAT('0GAMMA LAMBDA CONVERGENCE'//' LAMBDA=',F8.4,5X,'GAMMA=',
      1 F8.4)
       METHOUT=3
       LINCNT=LINCNT+4
       SSQR(2)=SSQR(1)
       SE2=SE
       L2=.TRUE.
       GO TO 680
```

```
320   GAMMA=0.
325   GAMCRT=.TRUE.
      IF(GAMMA.GT.CRTGAM) GAMCRT=.FALSE.
C*****************************************************
C CHECK FOR PASSING THE EPSILON CRITERIA
C*****************************************************
330   DO 340 I=1,NP
      DELTA(I)=DELM*DELTA(I)
      IF (((DABS(DELTA(I)))/(TAU/ADIAG(I)+DABS(B(I)))).GE.EPS)GOTO 350
340   CONTINUE
      IF(MQT.AND.EPS.EQ.EPS2)L1=.TRUE.;GO TO 360
      EPS=EPS2
      IF(.NOT.MQT)MQT=.TRUE.;LAMBDA=0.0
350   L1=.FALSE.
C*****************************************************************
C BEGIN CHECK FOR IMPROVED ERROR SUM OF SQUARES.
C*****************************************************
360   DO 370 I=1,NP
370   TEMPAR(I)=B(I)+DELTA(I)
      CALL FCODE(TEMPAR,Q,SUMQ2,SSQR(2))
      SE2=DSQRT(SSQR(2)/NDC)
      IMPRVD=.FALSE.
      IF(SSQR(2).LT.SSQR(1))IMPRVD=.TRUE.
      IF(IT.EQ.1.AND.KK.EQ.0)GO TO 371
      IF(.NOT.IMPRVD.AND.GAMCRT) GO TO 405
C*************************************************************
C COMPUTE FLETCHERS 'R' RATIO AND BRANCH ACCORDINGLY.
C*****************************************************
371   PRELAM=LAMBDA
      R=(SSQR(1)-SSQR(2))/SUM1
C*********************************************************
C PRINT DETAILED DIAGNOSTICS IF REQUESTED
C*********************************************************
405   IF(IDIAG.GE.0.OR.(IT.GT.IABS(IDIAG).AND.IT+IABS(IDIAG).LT.NI))
     $   GO TO 440
      IF(LINCNT+4+(3*KLINES).GT.LINMAX) CALL PAGER
      WRITE(6,410)IT,KK+1
410   FORMAT(/,' DIAGNOSTICS FOR ITERATION ',I3,' ROUND ',I2,/)
      IF( GAMCRT)GO TO 430
      WRITE(6,420)LAMBDA,LAMBDAO ,GAMMA,R,SSQR(2)
420   FORMAT(' LAMBDA=',E9.3,' LAMBDAO=',E9.3,
     $ ' GAMMA=',E9.3,' FLETCHER''S R=',E9.3,' SSQR(2)=',E13.8)
      GO TO 435
430   WRITE(6,415)DELM,SSQR(2),R
415   FORMAT(' GAMMA CRITICAL;  ',' DELM=',E9.3,' SSQR(2)=',E13.8,
     $' FLETCHER''S R=',E9.3)
```

```
435   LINCNT=LINCNT+4
      K=0
      CALL PARAOUT(K)
      IF(.NOT.IMPRVD.AND.GAMCRT)GO TO 380
440   CONTINUE
C*************************************************************
C END OF DETAILED DIAGNOSTICS
C*************************************************************
      IF(R.GT.SIGMA.AND.LAMBDA.EQ.0.0)GO TO 380
      IF(R.GT.SIGMA) GO TO 373
      IF(R.LT.RHO.AND.LAMBDA.EQ.0.)GO TO 376
      IF(R.LT.RHO) GO TO 374
      GO TO 380
373   LAMBDA=LAMBDA/2.0
      IF(LAMBDA.LT.EIGNL)LAMBDA=LAMBDA/2.
      IF(LAMBDA.LT.LAMBDAO)LAMBDA=0.0
      GO TO 380
374   NU=2-(SSQR(1)-SSQR(2))/SGDEL
      IF(NU.GT.10.0)NU=10.0
      IF(NU.LT.2.0)NU=2.0
      IF(MQT)GO TO 378
      IF(LAMBDA.LT.EIGNL)NU=10.0
378   LAMBDA=LAMBDA*NU
      GO TO 380
C*****************************************************************
C  CALCULATE P'P INVERSE AND TRACE TO GET LAMBDAO.
C*****************************************************************
 376  A(1,1)=1.0
      DO 377 I=2,NP
      A(I,I)=1.0
      DO 377 J=1,I-1
377   A(J,I)=A(I,J)
      CALL SYMINV(&390,NP)
      TR=0.0
      DO 379 I=1,NP
379   TR=TR+A(I,I)
      GO TO 395
390   WRITE(6,940)
      LINCNT=LINCNT+1
940   FORMAT('   SINGULAR MATRIX IN SYMINV AT 390')
      TR=1.0E15
C*****************************************************************
C   ASSIGN EIGNL AS THE LOWEST BOUND OF THE SMALLEST EIGENVALUE
C   OF MATRIX P'P WHICH IS 1./TR.
C*****************************************************************
395   EIGNL=1./TR
```

```
      LAMBDAO=EIGNL*(EIGNL+ETA)
      IF(MQT)LAMBDAO=EIGNL
      LAMBDA=LAMBDAO
C*************************************************************
C IF THERE HAS BEEN NO IMPROVEMENT IN THE ERROR SUM
C OF SQUARES GO TO STATEMENT 540 AND CHECK FOR HAVING
C PASSED EITHER THE GAMMA EPSILON OR THE EPSILON CRITERION.
C*************************************************************
380   IF(.NOT.IMPRVD)GO TO 540
      DO 385 I=1,NP
385   B(I)=TEMPAR(I)
      SUMQ=SUMQ2
      SE=SE2
      IT=IT+1
      IF(IT.GE.NI)   GO TO 620
      KK=0
      IF(IDIAG.EQ.0.OR.(IT-1.GT.IABS(IDIAG).AND.IT+IABS(IDIAG).LT.NI))
     $GO TO 530
C*************************************************************
C ABBREVIATED DIAGNOSTICS ARE PRINTED BELOW IF THEY ARE CALLED FOR.
C*************************************************************
      IF(LINCNT+10+(4*KLINES).GT.LINMAX)CALL PAGER
      WRITE(6,400) IT-1
400   FORMAT(///,' SUMMARY DIAGNOSTICS FOR ITERATION NUMBER',I4)
      LINCNT=LINCNT+10
      K=1
      CALL PARAOUT(K)
      WRITE(6,510) SSQR(1),SSQR(2),SUMQ,LAMBDAO
510   FORMAT(/,9H SSQR(1)=,1PE15.8,5X,8HSSQR(2)=,1PE15.8,5X,5HSUMQ=,
     1 1PE13.6,5X,8HLAMBDAO=,1PE13.6)
      WRITE(6,520) GAMMA,PRELAM,R,SE2
520   FORMAT(/,7H GAMMA=,F10.4,5X,7HLAMBDA=,1PE13.6,5X,'FLETCHER''S R='
     1 ,1PE13.6,5X,4HSE2=,1PE13.6//)
C*************************************************************
C THE DIAGNOSTIC  PRINTING ENDS HERE.
C*************************************************************
C IF THE EPSILON TEST HAS BEEN PASSED AND GAMMA IS CRITICAL
C BEGIN CONCLUDING VIA GAMMA EPSILON CONVERGENCE.
C THIS RESULT IS DUE TO ROUNDING ERROR.
C*************************************************************
530   IF(L1.AND.GAMCRT) GO TO 545
C*************************************************************
C IF THE EPSILON TEST HAS BEEN PASSED BUT GAMMA IS NOT
C CRITICAL BEGIN CONCLUDING VIA EPSILON CONVERGENCE.
C*************************************************************
      IF(L1)GO TO 640
```

```
C*************************************************************
C BEGIN A NEW ITERATION BY CALCULATING THE PARTIALS.
C*************************************************************
      SSQR(1)=SSQR(2)
      IF(.NOT.MQT.AND.GAMCRT)MQT=.TRUE.;LAMBDA=0.0
      GO TO 160
C*************************************************************
C IF GAMMA IS NOT CRITICAL BEGIN A NEW ROUND
C IF GAMMA IS CRITICAL DECREASE THE DELTAS AND ATTEMPT TO
C PASS THE EPSILON TEST.
C NOTE THAT WE ENTER BELOW IF NO IMPROVEMENT WAS FOUND IN
C THE ERROR SUM OF SQUARES.
C*************************************************************
540   IF(.NOT.GAMCRT) GO TO 590
      DELM=DELM/2.
      IF(L1)GO TO 545
      KK=KK+1
      IF(KK.GT.KKMAX)GO TO 660
      IF(MQT)GO TO 330
      MQT=.TRUE.
      LAMBDA=0.0
      GO TO 160
545   CALL PAGER
      WRITE(6,550)
550   FORMAT(/,' GAMMA EPSILON CONVERGENCE'/)
      LINCNT=LINCNT+3
      L2=.TRUE.
      METHOUT=2
      IF(IMPRVD)GO TO 680
      CALL FCODE(B,Q,SUMQ,SSQR(2))
      SE2=SE
      GO TO 680
590   KK=KK+1
      IF(KK.GT.KKMAX) GO TO 660
      IF(.NOT.L1.AND.LAMBDA.EQ.LAMBDAO)GO TO 215
      IF(.NOT.L1)GO TO 235
      CALL FCODE(B,Q,SUMQ,SSQR(2))
      SE2=SE
      GO TO 640
620   IF(L1)GO TO 640
      CALL PAGER
      WRITE(6,630)
630   FORMAT(/,' MAXIMUM ITERATIONS DONE'/)
      LINCNT=LINCNT+3
      MAXIT=.TRUE.
      L2=.FALSE.
```

```
      METHOUT=4
      WRITE(6,975)EPS
975   FORMAT('   THE FOLLOWING PARAMETERS DID NOT PASS THE EPSILON'
     $     ' CRITERIA OF ',E10.5,/)
      LINCNT=LINCNT+2
      DO 635 I=1,NP
      EPSILON=DABS(DELTA(I))/((TAU/ADIAG(I))+DABS(B(I)))
      IF(EPSILON.GE.EPS)WRITE(6,980)XHEAD(I),EPSILON;LINCNT=LINCNT+1
635   CONTINUE
980   FORMAT(3X,A4,'      EPSILON= ',E10.5)
      WRITE(6,990)
      LINCNT=LINCNT+1
      GO TO 680
640   CALL PAGER
      WRITE(6,650)
650   FORMAT(/,' EPSILON CONVERGENCE'/)
      LINCNT=LINCNT+3
      METHOUT=1
      L2=.TRUE.
      GO TO 680
660   CALL PAGER
      WRITE(6,670)
670   FORMAT(/,' PARAMETERS NOT CONVERGING'/)
      LINCNT=LINCNT+3
      METHOUT=5
      L2=.FALSE.
680   WRITE(6,690)IT
690   FORMAT (4X,'TOTAL NUMBER OF ITERATIONS WAS ',I5,/)
      LINCNT=LINCNT+2
      IF(NRESD.EQ.0) GO TO 715
C*************************************************************
C WRITE PREDICTED AND RESIDUAL VALUES.
C*************************************************************
      IF(SEQ)WRITE(6,960);GO TO 701
700   WRITE(6,985)
701   LINCNT=LINCNT+3
      DO 705 I=1,ND
      IF(NPT.GT.0)GO TO 702
      IF(SEQ)WRITE(6,965)I,Q(I);GO TO 704
      WRITE(6,990)Y(I),F(I),Q(I)
      GO TO 704
702   IF(SEQ)WRITE(6,965)I,Q(I),(PRNT(I,J),J=1,NPT);GO TO 704
      WRITE(6,990)Y(I),F(I),Q(I),(PRNT(I,J),J=1,NPT)
704   LINCNT =LINCNT+1
      IF(LINCNT.LT.LINMAX)GO TO 705
      CALL PAGER
```

```
      IF(SEQ)WRITE(6,960)
      WRITE(6,985)
      LINCNT=LINCNT+1
705   CONTINUE
985   FORMAT('     OBS          PRED      RESIDUAL')
960   FORMAT(5X,' EQ     RESIDUAL')
990   FORMAT(3E13.6,X,5E12.5)
965   FORMAT(5X,I3,E14.6,5E13.5)
755   IF(.NOT.SEQ)CALL GRAPH
C*************************************************************
C END OF RESIDUAL OUTPUT.
C*************************************************************
715   IF(.NOT.L2.AND..NOT.MAXIT)GO  TO 750
      IF(SEQ)GO TO 736
      IF(LINCNT+21.GT.LINMAX) CALL PAGER
      SUMYSQ=SUMYSQ-((SAVSUM**2)/ND)
      WRITE(6,710) SUMYSQ
710   FORMAT(/,'  TOTAL SUM OF SQUARES= ',T38,E16.10)
      WRITE(6,720) SSQR(2),SUMQ
720   FORMAT(/,'  RESIDUAL SUM OF SQUARES=',T38,E16.10,/
     $      /,'  SUM OF RESIDUALS=',T38,E16.10)
      WRITE(6,730) SE2
      LINCNT=LINCNT+8
730   FORMAT(/,29H  STANDARD ERROR OF ESTIMATE=,T38,E16.10)
C*************************************************************
C BEGIN R-SQUARED.
C*************************************************************
745   RSQ=1-(SSQR(2)/SUMYSQ)
      WRITE(6,944)RSQ
944   FORMAT(/,'  THE MULTIPLE R-SQUARED= ',T38,F5.3)
      LINCNT=LINCNT+2
C*************************************************************
C END OF R-SQUARED DERIVATION.
C*************************************************************
735   WRITE(6,947)NDC
947   FORMAT(/,'  DEGREES OF FREEDOM= ',T36,I3,'.')
      LINCNT=LINCNT+2
C*****************************************************************
C  CALCULATE THE DURBIN-WATSON D STATISTIC.
C*****************************************************************
      DURBIN=0.0
      DO 725 I=1,ND-1
725   DURBIN=DURBIN+(Q(I+1)-Q(I))**2
      DURBIN=DURBIN/SSQR(2)
      WRITE(6,955)DURBIN
955   FORMAT(/,'  DURBIN-WATSON D STATISTIC= ',T38,E10.4)
```

```
      LINCNT=LINCNT+2
      SSQR(1)=SE2
      WRITE(7) SSQR,METHOUT,RSQ
736   ACON=.TRUE.
      GO TO 160
740   CALL CLIM
      RETURN
750   IF(LINCNT+2+(3*KLINES).GT.LINMAX) CALL PAGER
      WRITE(6,970)
970   FORMAT(/,'  LAST ESTIMATES')
      IF(.NOT.SEQ)WRITE(7)SSQR,METHOUT,RSQ
      K=-1
      CALL PARAOUT(K)
      WRITE(6,950)SSQR(2)
950   FORMAT(/,'  RESIDUAL SUM OF SQUARES=',E16.10)
      GO TO 999
900   WRITE(6,920)
920   FORMAT('   DELTAS ARE INCONSISTENT WITH LINEAR MODEL--SEVERE'
     $' COMPUTATIONAL ERRORS.')
999   RETURN
      END
      SUBROUTINE CLIM
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25),TEMPAR(25),PPINV(650)
     $,C(5,25),H(5,25)
      COMMON/SINGULAR/DET
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,TEMPAR,TMA(5,5)
     $,PPINV,S(25,5),PRNT
      LOGICAL IOLEO,SEQ
      REAL LOPCL,LSPCL,NPC,LAMBDA
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
     $,NPRNT,MAXWIDE,NAME(25,5)
      INTEGER XHEAD,HEAD ,SUB
C*************************************************************
C     THIS FUNCTION ASSIGNS STORAGE VECTOR LOCATIONS TO
C     TRIANGULAR MATRIX ELEMENTS, COLUMN BY COLUMN.
      SUB(I,J)=I+(J*J-J)/2
C*************************************************************
C*************************************************************
C PLACE P'P IN THE UPPER TRIANGLE OF A AND SAVE THE
C DIAGONAL ELEMENTS AS ADIAG IN PREPARATION OF INVERSION.
C*************************************************************
```

```
      IF(SEQ)GO TO 600
      CONS3=1.0E-15
      K4=0
      DO 4 I=1,NP
      G(I)=DSQRT(A(I,I))
      ADIAG(I)=A(I,I)
 4    A(I,I)=1.0
      DO 5 I=2,NP
      DO 5 J=1,I-1
 5    A(I,J)=A(I,J)/(G(I)*G(J))
701   DO 7 I=2,NP
      DO 7 J=1,I-1
7     A(J,I)=A(I,J)
      K=3
      IF(IDIAG.NE.0)CALL MATOUT(NP,NP,K)
      CALL SYMINV(&420,NP)
      IF(DET.LT.(1.0E-30)) WRITE(6,950)
 950  FORMAT(///,10X,'WARNING:  P-TRANSPOSE-P IS NEARLY SINGULAR')
C*************************************************************
C THE UPPER TRIANGLE OF A IS NOW SCALED P'P INVERSE.
C*************************************************************
C CALCULATE COVARIANCE MATRIX WHEN CONSTRAINT(S) INVOLVED.
C*************************************************************
      IF(NC.EQ.0) GO TO 64
C*************************************************************
C   CALCULATE INVERSE(P'P)*C'.
C*************************************************************
C SCALE THE CONSTRAINT MATRIX C.
      DO 12 J=1,NP
      DO 12 I=1,NC
 12   C(I,J)=C(I,J)/G(J)
      DO 15 I=1,NP
      DO 15 J=1,NC
      S(I,J)=0.
      DO 14 K=1,I
  14  S(I,J)=S(I,J)+A(K,I)*C(J,K)
      IF(I.EQ.NP) GO TO 15
      DO 15 K=I+1,NP
      S(I,J)=S(I,J)+A(I,K)*C(J,K)
  15  CONTINUE
C*************************************************************
C   STORE UPPER TRIANGLE OF A INTO PPINV.
C*************************************************************
      DO 40 I=1,NP
      DO 40 J=I,NP
40    PPINV(SUB(I,J))=A(I,J)
```

```
      DO 42 I=1,NC
      DO 42 J=I,NC
      A(I,J)=0.
      DO 43 K=1,NP
43    A(I,J)=A(I,J)+C(I,K)*S(K,J)
42    CONTINUE
      IF(NC.EQ.1)GO TO 56
      CALL SYMINV(&520,NC)
      DO 45 I=1,NC
      DO 45 J=I,NC
45    TMA(I,J)=TMA(J,I)=A(I,J)
      DO 50 I=1,NP
      DO 50 J=I,NP
      V=0.
      DO 60 K=1,NC
      DO 55 M=1,NC
55    V=V+S(I,K)*TMA(M,K)*S(J,M)
60    CONTINUE
50    A(I,J)=PPINV(SUB(I,J))-V
      GO TO 64
56    V=1.0/A(1,1)
      DO 58 I=1,NP
      DO 58 J=I,NP
      A(I,J)=PPINV(SUB(I,J))-S(I,1)*S(J,1)*V
58    CONTINUE
C TRANSFORM SCALED COV MATRIX TO ORIGINAL UNITS
 64   DO 220 I=1,NP
      DO 220 J=I,NP
220   A(I,J)=A(I,J)/(G(I)*G(J))
      K=4
      CALL MATOUT(NP,NP,K)
C*************************************************************
C   COMPUTE COVARIANCE MATRIX OF LINEAR RELATION(S).
C*************************************************************
      IF(NLR.EQ.0)GO TO 65
      DO 20 I=1,NP
      DO 20 J=I,NP
  20  PPINV(SUB(I,J))=A(I,J)
      DO 25 I=1,NP
      DO 25 J=1,NLR
      S(I,J)=0.0
      DO 22 K=1,I
  22  S(I,J)=S(I,J)+A(K,I)*H(J,K)
      IF(I.EQ.NP) GO TO 25
      DO 25 K=I+1,NP
      S(I,J)=S(I,J)+A(I,K)*H(J,K)
```

```
  25  CONTINUE
      DO 27 I=1,NLR
      DO 27 J=I,NLR
      A(I,J)=0.0
      DO 26 K=1,NP
  26  A(I,J)=A(I,J)+H(I,K)*S(K,J)
  27  CONTINUE
      K=6
      CALL MATOUT(NLR,NLR,K)
      IF(LINCNT+3+NLR.GT.LINMAX)CALL PAGER
      WRITE(6,750)
750   FORMAT(/,T3,'RELATION',T17,'STANDARD ERROR',/)
      DO 35 I=1,NLR
      G(I)=SE*DSQRT(A(I,I))
35    WRITE(6,850)I,G(I)
850   FORMAT(7X,I3,7X,E11.5)
      LINCNT=LINCNT+3+NLR
      DO 36 I=1,NP
      DO 36 J=I,NP
36    A(I,J)=PPINV(SUB(I,J))
C*****************************************************
C CALCULATE THE PARAMETER CORRELATION MATRIX AND WRITE
C IT.
C*****************************************************
65    DO 10 I=1,NP
10    G(I)=DSQRT(A(I,I))
      DO 30 I=1,NP
      A(I,I)=1.0
      DO 30 J=I+1,NP
      A(I,J)=A(I,J)/(G(I)*G(J))
30    CONTINUE
      K=5
      CALL MATOUT(NP,NP,K)
      DO 70 I=1,NP
70    ADIAG(I)=SE*G(I)
85    NPC=NP-IO+NC
      IF(LINCNT+5+NP.GT.LINMAX) CALL PAGER
      WRITE(6,80)
80    FORMAT(/,' APPROXIMATE 95% LINEAR CONFIDENCE LIMITS')
C*********************************************************
C STORE THE ANSWERS ON FILE 7 FOR USE IN THE SUMMARY.
C*********************************************************
      WRITE(7) NP,IO,IOL,B,ADIAG
C*********************************************************
C WRITE OUT THE ANSWERS TO THIS EQUATION.
C*********************************************************
```

```
      WRITE(6,90)
90    FORMAT(/,T13,'PARAMETER',T27,'STANDARD',T47,'ONE-PARAMETER',
     $   T75,'SUPPORT PLANE',/' PARAMETER   ESTIMATE',T29,'ERROR',
     $   T45,2('LOWER',7X,'UPPER',11X))
      DO 150 I=1,NP
      IF(IOLE0) GO TO 110
      DO 100 J=1,IO
      IF(I.EQ.IOL(J)) GO TO 130
100   CONTINUE
110   SPCF=SQRT(NPC*FS)*ADIAG(I)
      OPCF=ADIAG(I)*T
      LOPCL=B(I)-OPCF
      UOPCL=B(I)+OPCF
      LSPCL=B(I)-SPCF
      USPCL=B(I)+SPCF
      WRITE(6,120) XHEAD(I),B(I),ADIAG(I),LOPCL,UOPCL,LSPCL,USPCL
120   FORMAT(3X,A4,X,2(3X,E11.5),2(5X,2(E11.5,X)))
      GO TO 150
130   WRITE(6,140) XHEAD(I)
140   FORMAT(3X,A4,13X,'OMITTED')
150   CONTINUE
      LINCNT=LINCNT+5+NP
      GO TO 800
420   WRITE(6,430)
      LINCNT=LINCNT+2
430   FORMAT(/,' SINGULAR MATRIX IN CONFIDENCE LIMIT ROUTINE. STD '
     $  'ERRORS  AND CONFIDENCE LIMITS BELOW ARE RELATIVE.')
      CONS3=CONS3*10
      DO 464 I=1,NP
464   A(I,I)=1.0+CONS3
      K4=K4+1
      IF(K4.LT.11) GO TO 701
      DO 465 I=1,NP
465   ADIAG(I)=9999.99
      OPCF=SPCF=0.0
      GO TO 85
520   WRITE(6,900)
900   FORMAT('   LINEAR DEPENDENCE IN CONSTRAINTS')
800   RETURN
600   DO 610 I=NP+1,25
610   B(I)=0.0
      KL=NP/(MAXWIDE+1)+1
      IF(LINCNT+7+3*KL.GT.LINMAX)CALL PAGER
      WRITE(6,620)SSQR(2)
620   FORMAT(////,'SOLUTION OF SYSTEM EQUATIONS',//,' ERROR'
     $' SUM OF SQUARES=',E16.10)
```

```
      K=-1
      CALL PARAOUT(K)
      RETURN
      END
      SUBROUTINE CONS(*,*,*)
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25),TEMPAR(25),PPINV(650)
     $,C(5,25),H(5,25)
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,TEMPAR,PRNT
     $ ,PPINV,C1(5,200),S(25,5),W(5),V
      EQUIVALENCE(PRNT,C1)
      CALL SYMINV(&7,NP)
      GO TO 5
7     RETURN 2
C SCALE THE CONSTRAINT MATRIX, C.
 5    DO 8 J=1,NP
      DO 8 I=1,NC
 8    C1(I,J)=C(I,J)/ADIAG(J)
C*************************************************************
C   CALCULATE INVERSE(A)*C'  AND STORE IN MATRIX S.
C*************************************************************
      DO 10 I=1,NP
      DO 10 J=1,NC
      S(I,J)=0.
      DO 11 K=1,I
11    S(I,J)=S(I,J)+A(K,I)*C1(J,K)
      IF(I.EQ.NP)GO TO 10
      DO 12 K=I+1,NP
12    S(I,J)=S(I,J)+A(I,K)*C1(J,K)
10    CONTINUE
C*************************************************************
C   UNCONSTRAINED INCREMENT.
C*************************************************************
      DO 15 M=1,NP
      DELTA(M)=0.
      DO 13 L= 1,M
13    DELTA(M)=DELTA(M)+A(L,M)*G(L)
      IF(M.EQ.NP)GO TO 15
      DO 14 L=M+1,NP
14    DELTA(M)=DELTA(M)+A(M,L)*G(L)
15    CONTINUE
C*************************************************************
```

```
C  CALCULATE C*INVERSE(A)*C' AND STORE IN UPPER TRIANGLE OF A.
C*****************************************************************
      DO 30 I=1,NC
      DO 30 J=I,NC
      A(I,J)=0.0
      DO 20 K=1,NP
20    A(I,J)=A(I,J)+C1(I,K)*S(K,J)
30    CONTINUE
      IF(NC.EQ.1)GO TO 100
C*****************************************************************
C  GET INVERSE OF C*INVERSE(A)*C'.
C*****************************************************************
      CALL SYMINV(&35,NC)
      GO TO 36
35    WRITE(6,970);RETURN 3
970   FORMAT('   LINEAR DEPENDENCE IN CONSTRAINTS.')
C*****************************************************************
C  CALCULATE C*DELTA AND STORE AS P.
C*****************************************************************
36    DO 40 I=1,NC
      P(I)=0.
      DO 40 K=1,NP
40    P(I)=P(I)+C1(I,K)*DELTA(K)
      DO 70 I=1,NC
      W(I)=0.
      DO 60 K=1,I
60    W(I)=W(I)+A(K,I)*P(K)
      IF(I.EQ.NC)GO TO 70
      DO 65 K=I+1,NC
65    W(I)=W(I)+A(I,K)*P(K)
70    CONTINUE
C*****************************************************************
C  CALCULATE S*W AND STORE AS P.
C*****************************************************************
      DO 80 I=1,NP
      P(I)=0.
      DO 80 K=1,NC
80    P(I)=P(I)+S(I,K)*W(K)
      GO TO 130
C*****************************************************************
C  CALCULATION OF CORRECTION TO DELTA VECTOR WHEN ONLY
C  ONE    CONSTRAINT.
C*****************************************************************
100   V=0.
      DO 110 K=1,NP
110   V=V+C1(1,K)*DELTA(K)
```

```
      DO 120 I=1,NP
120   P(I)=S(I,1)*V/A(1,1)
130   DO 140 I=1,NP
140   DELTA(I)=DELTA(I)-P(I)
      RETURN 1
      END
      SUBROUTINE PAGER
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
      INTEGER XHEAD,HEAD
      IPAGE=IPAGE+1
      WRITE(6,10) HEAD,NDATE,IPAGE
 10   FORMAT('1NONLINEAR LEAST SQUARES ',4X, 11A4,5X,4A4,X,'PG.',I3/)
      LINCNT=2
      RETURN
      END
      SUBROUTINE DATER
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
      INTEGER XHEAD,HEAD
      DATA FPT/8Z90000006/
S        LI,6     NDATE
S        CAL1,8   FPT
      RETURN
      END
      SUBROUTINE    SYMINV(*,N)
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLE0,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25)
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,PRNT
      REAL LAMBDA
      LOGICAL IOLE0,MQT,SEQ
C*****************************************************************
C     SUBROUTINE    TO INVERT A POSITIVE DEFINITE MATRIX BY
C     CHOLESKY    METHOD
C*****************************************************************
      DOUBLE    PRECISION DET,SUM
C*****************************************************************
C     FACTOR    A INTO A=(LL')
C*****************************************************************
      CALL CHOL(&30,N)
C*****************************************************************
C     INVERT    UPPER TRIANGULAR MATRIX (L').  THIS MUST BE
C     DONE    FROM COLUMN N,N-1,...,1 SO AS NOT TO DESTROY
C     NEEDED    ELEMENTS.
```

```
C*************************************************************
      DO    15 L=1,N
      K=N+1-L
      A(K,K)=1.0/A(K,K)
      DO    10 M=L,N
      J=N+1-M
      IF(J.EQ.K)    GO TO 10
      SUM=0.0
      J1=J+1
      DO    5 I=J1,K
 5    SUM=SUM-A(J,I)*A(I,K)
      A(J,K)=SUM/A(J,J)
 10   CONTINUE
 15   CONTINUE
C*************************************************************
C     NOW    A CONTAINS L TRANSPOSE INVERSE
C     MULTIPLY    INVERTED LOWER TRIANGLES TOGETHER IN REVERSE
C     ORDER.
C*************************************************************
      DO    25 I=1,N
      DO    25 J=I,N
      SUM=0.0
      DO    20 K=J,N
 20   SUM=SUM+A(I,K)*A(J,K)
      A(I,J)=SUM
 25   CONTINUE
      RETURN
 30   RETURN1
      END
      SUBROUTINE    CHOL(*,N)
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25)
      COMMON/SINGULAR/DET
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,PRNT
      REAL LAMBDA
      LOGICAL IOLEO,MQT,SEQ
C*************************************************************
C     CHOLESKY    DECOMPOSITION OF A POSITIVE DEFINITE
C     MATRIX    STORED AS AN UPPER TRIANGLE, COLUMN BY COLUMN
C*************************************************************
      DOUBLE    PRECISION DET,SUM,PIVOT
      DET=1.0
```

```
      DO    25 I=1,N
      LOOP=I-1
      DO    20 J=I,N
      SUM=0.0
      IF(LOOP.LE.0)    GO TO 10
      DO    5 K=1,LOOP
 5    SUM=SUM+A(K,I)*A(K,J)
 10   SUM=A(I,J)-SUM
      IF(I.NE.J)    GO TO 15
C*****************************************************************
C     TEST    FOR FAILURE OF POSITIVE DEFINITENESS
C*****************************************************************
      IF(SUM.LE.OMEGA)     RETURN1
C*****************************************************************
C     TRANSFORM    PIVOT
C*****************************************************************
      PIVOT=DSQRT(SUM)
      A(I,J)=PIVOT
      DET=DET*PIVOT
      PIVOT=1.0/PIVOT
      GO    TO 20
C*****************************************************************
C     COMPUTE    OFF-DIAGONAL TERMS
C*****************************************************************
 15   A(I,J)=SUM*PIVOT
 20   CONTINUE
 25   CONTINUE
C*****************************************************************
C     COMPUTE    DET(A)=DET(L)*DET(L')
C*****************************************************************
      DET=DET*DET
      IF(DET.LE.OMEGA)RETURN 1
      RETURN
      END
      SUBROUTINE MATOUT(NROW,NCOL,K)
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25)
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
     $,NPRNT,MAXWIDE
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,PRNT
      INTEGER XHEAD,HEAD
      I1=1;I2=MIN0(NCOL,MAXWIDE);NCHK=0
1     IF(K-2)10,20,140
10    IF(LINCNT+16.GT.LINMAX)CALL PAGER
      WRITE(6,911);GO TO 40
```

```
911   FORMAT(/,3X,'ORIGINAL DATA'/)
20    IF(LINCNT+16.GT.LINMAX)CALL PAGER
      WRITE(6,912)
912   FORMAT(/,'   TRANSFORMED DATA'/)
40    WRITE(6,901)(XHEAD(J),J=I1,I2)
901   FORMAT(' OBS',10(4X,A4,4X))
      LINCNT=LINCNT+4
      IF(NCHK.EQ.1)GO TO 150
      I=1
      REPEAT 160, WHILE I.LE.NROW
150   WRITE(6,902)I,(X(I ,J),J=I1,I2)
902   FORMAT(X,I3,X,10(E11.5,X))
      I=I+1
      LINCNT=LINCNT+1
      IF(LINCNT.GE.LINMAX)NCHK=1;GO TO 1
160   CONTINUE
      IF(I2.GE.NCOL) GO TO 170
      I1=I2+1
      I2=MIN0(NCOL,I1+MAXWIDE-1)
      NCHK=0
      IF(LINMAX.LE.LINCNT+10) GO TO 1
      GO TO 40
170   RETURN
140   KLINES=NCOL/(MAXWIDE+1)
      KLINES=(NCOL+1)*(KLINES+1)+3-(NCOL-MAXWIDE)
      IF(LINCNT+KLINES.GT.LINMAX.AND.KLINES.LE.LINMAX) CALL PAGER
      I=1
      LINCNT=LINCNT+3
55    GO TO (45,50,60,70)K-2
45    WRITE(6,914); GO TO 100
914   FORMAT(/,'   SCALED P TRANSPOSE P ',/)
50    WRITE(6,915);GO TO 100
915   FORMAT(/,'   PARAMETER COVARIANCE MATRIX/(SIGMA SQR)',/)
60    WRITE(6,918);GO TO 100
918   FORMAT(/,'   PARAMETER CORRELATION MATRIX ',/)
70    WRITE(6,919);GO TO 100
919   FORMAT(/,'   COVARIANCE MATRIX/(SIGMA SQR) FOR LINEAR RELATION(S)
     $OF PARAMETERS.',/)
100   CONTINUE
105   WRITE(6,920)(XHEAD(J),J=I1,I2)
920   FORMAT(5X,10(4X,A4,4X))
      LINCNT=LINCNT+1
      I=1
      REPEAT 125, WHILE I .LE. I1
      WRITE(6,921)XHEAD(I),(A(I,J),J=I1,I2)
115   LINCNT=LINCNT+1
```

```
      I=I+1
125   CONTINUE
      REPEAT 120, WHILE I.LE.I2
      WRITE(6,922)XHEAD(I),I-I1,(A(I,J),J=I,I2)
      GO TO 215
922   FORMAT(X,A4,N(12X),10(E11.5,X))
921   FORMAT(X,A4,10(E11.5,X))
215   I=I+1
       LINCNT=LINCNT+1
120   CONTINUE
      IF(I2.GE.NCOL) GO TO 130
      I1=I2+1
      I2=MIN0(I1+MAXWIDE-1,NCOL)
      GO TO 105
130   RETURN
      END
      SUBROUTINE PARAOUT (K)
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLE0,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25)
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
     $,NPRNT,MAXWIDE
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,PRNT
      INTEGER XHEAD,HEAD
      REAL LAMBDA
      LOGICAL IOLE0,MQT,SEQ
C*********************************************************************
C WRITE PARAMETER ESTIMATES, DELTAS, OR STANDARD ERRORS.
C*********************************************************************
      I1=1;I2=MIN0(MAXWIDE,NP)
5     WRITE(6,901)(XHEAD(I),I=I1,I2)
      LINCNT=LINCNT+2
901   FORMAT(/,10X,10(4X,A4,4X))
      IF(K)20,10,10
10    WRITE(6,902)(DELTA(I),I=I1,I2)
      LINCNT=LINCNT+1
902   FORMAT('   DELTA  ',10(E11.5,X))
      IF(K.EQ.0) GO TO 30
20    WRITE(6,903)(B(I),I=I1,I2)
903   FORMAT(' PARAMETER ',10(E11.5,X))
      LINCNT=LINCNT+1
      IF(K.NE.-5) GO TO 30
      WRITE(6,904)(ADIAG(I),I=I1,I2)
```

```
904   FORMAT(' STD ERROR ',10(E11.5,X))
      LINCNT=LINCNT+1
30    IF(I2.EQ.NP)GO TO 40
      I1=I2+1
      I2=MIN0(I1+MAXWIDE-1,NP)
      GO TO 5
40    RETURN
      END
      SUBROUTINE TRAN
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLE0,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25)
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,PRNT
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
     $,NPRNT,MAXWIDE
      INTEGER XHEAD,HEAD
      REAL LAMBDA
      LOGICAL IOLE0,MQT,SEQ
      DIMENSION  TRN(20,5),NLOC(25)
      IMAX=NV
      READ(5,902) NTR,NADD,NLOC
902   FORMAT(27I2)
901   FORMAT(4I2,F10.5)
      DO 1 J=1,NTR
1     READ(5,901)(TRN(J,I),I=1,5)
      DO 600 N=   1,ND
      DO 500 I=1,NTR
      GO TO(20,30,40,50,60,70,80,90,100,110,120,130,140,
     $      150,160,170,180,190,200)TRN(I,1)
20    X(N,TRN(I,2))=ALOG(X(N,TRN(I,3)))
      GO TO 500
30    X(N,TRN(I,2))=ALOG10(X(N,TRN(I,3)))
      GO TO 500
40    X(N,TRN(I,2))=SIN(X(N,TRN(I,3)))
      GO TO 500
50    X(N,TRN(I,2))=COS(X(N,TRN(I,3)))
      GO TO 500
60    X(N,TRN(I,2))=TAN(X(N,TRN(I,3)))
      GO TO 500
70    X(N,TRN(I,2))=EXP(X(N,TRN(I,3)))
      GO TO 500
80    X(N,TRN(I,2))=X(N,TRN(I,3))**TRN(I,5)
      GO TO 500
```

```
90    X(N,TRN(I,2))=ABS(X(N,TRN(I,3)))
      GO TO 500
100   IF(X(N,TRN(I,3)).EQ.0)X(N,TRN(I,2))=TRN(I,5)
      GO TO 500
110   X(N,TRN(I,2))=X(N,TRN(I,3))+X(N,TRN(I,4))
      GO TO 500
120   X(N,TRN(I,2))=X(N,TRN(I,3))+TRN(I,5)
      GO TO 500
130   X(N,TRN(I,2))=X(N,TRN(I,3))-X(N,TRN(I,4))
      GO TO 500
140   X(N,TRN(I,2))=X(N,TRN(I,3))*X(N,TRN(I,4))
      GO TO 500
150   X(N,TRN(I,2))=X(N,TRN(I,3))*TRN(I,5)
      GO TO 500
160   X(N,TRN(I,2))=X(N,TRN(I,3))/X(N,TRN(I,4))
      GO TO 500
170   X(N,TRN(I,2))=X(N,TRN(I,3))/TRN(I,5)
      GO TO 500
180   X(N,TRN(I,2))=TRN(I,5)/X(N,TRN(I,3))
      GO TO 500
190   X(N,TRN(I,2))=X(N,TRN(I,3))
      GO TO 500
200   X(N,TRN(I,2))=N
500   IMAX=MAX0(IMAX,TRN(I,2))
      IF(NLOC(1).EQ.0) GO TO 600
      DO 620 I=1,IMAX
620   Y(I)=X(N,I)
      DO 630 I=1,NV+NADD
630   X(N,I)=Y(NLOC(I))
600   CONTINUE
      NV=NV+NADD
      K=2
      WRITE(6,910)
910   FORMAT(3/)
      LINCNT=LINCNT+3
      CALL MATOUT(NPRNT,NV,K)
      RETURN
      END
      SUBROUTINE SUMMARY
      COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
     $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
     $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
      COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
     $,Q(200),B(25),P(25),A(25,25)
     $,G(25),ADIAG(25),DELTA(25)
      DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,PRNT
```

```
      COMMON/PAGECM/LINCNT,LINMAX,HEAD(11),NDATE(4),IPAGE,XHEAD(25)
     $,NPRNT,MAXWIDE
      INTEGER XHEAD,HEAD
      REAL LAMBDA
      LOGICAL IOLE0,MQT,SEQ
      REWIND 7
      K=-5
      IT=0
      CALL PAGER
      WRITE(6,900)
  900 FORMAT(20X,'SUMMARY OF RESULTS'//)
      LINCNT=LINCNT+3
   10 READ(7,END=50)SSQR,METHOUT,RSQ
      IT=IT+1
      IF(METHOUT.NE.5)    GO TO 20
      IF(LINCNT+3.LT.LINMAX) GO TO 15
      CALL PAGER
      WRITE(6,900)
      LINCNT=LINCNT+3
   15 LINCNT=LINCNT+3
      WRITE(6,901)IT
  901 FORMAT(/,'  EQUATION NO.',I2,' DID NOT CONVERGE'/)
      GO TO 10
   20 READ(7)NP,IO,IOL,B,ADIAG
      KNP=NP/MAXWIDE
      IF(LINCNT+4+(4*KNP+1).LT.LINMAX) GO TO 5
      CALL PAGER
      WRITE(6,900)
      LINCNT=LINCNT+3
    5 LINCNT=LINCNT+2
      GO TO(21,22,23,24)METHOUT
   21 WRITE(6,902)IT
  902 FORMAT(/,'  EQUATION NO. ',I2,10X,'EPSILON CONVERGENCE')
      GO TO 25
   22 WRITE(6,903) IT
  903 FORMAT(/,'  EQUATION NO. ',I2,10X,'GAMMA EPSILON CONVERGENCE')
      GO TO 25
   23 WRITE(6,904) IT
  904 FORMAT(/,'  EQUATION NO. ',I2,10X,'GAMMA LAMBDA CONVERGENCE')
      GO TO 25
   24 WRITE(6,905)IT
  905 FORMAT(/,'  EQUATION NO. ',I2,10X,'MAXIMUM ITERATIONS')
   25 IF(IO.EQ.0)GO TO 40
      DO 30 I=1,NP
      DO 30 J=1,IO
   30 IF(I.EQ.IOL(J))B(I)=ADIAG(I)=0.0
```

```
40     CALL PARAOUT(K)
       WRITE(6,906)SSQR,RSQ
       LINCNT=LINCNT+2
906    FORMAT('  STANDARD ERROR OF THE ESTIMATE= ',E11.5,
      $       ' RESIDUAL SUM OF SQUARES= ',E11.5,
      $      /,'  MULTIPLE R-SQUARED= ',F5.3)
       GO TO 10
50     WRITE(6,907)
907    FORMAT('1',5/,10X,'JOB COMPLETED')
       RETURN
       END
       SUBROUTINE GRAPH
       COMMON NV,ND,NP,IT,NDC,IO,SE,RSQ,NPAR,MQT,NC,NLR,EPS1
      $,KKMAX,EPS2,TAU,LAMBDA,RHO,SIGMA,CRTGAM,IDIAG,NPT,SEQ
      $,OMEGA,FS,IOLEO,IOL(49),T,SSQR(2),NI,NRESD,ETA
       COMMON/MARDAT/ X(200,25),Y(200),F(200),PRNT(200,5)
      $,Q(200),B(25),P(25),A(25,25)
      $,G(25),ADIAG(25),DELTA(25)
       DOUBLE PRECISION P,A,B,G,ADIAG,DELTA,SSQR,PRNT
       COMMON/PAGECM/LINCNT,LINMAX,HEAD(11) ,NDATE(4),IPAGE,XHEAD(25)
      $,NPRNT,MAXWIDE
       REAL MAXRES
       CALL PAGER
       WRITE(6,901)
901    FORMAT(/,T39,'PLOT OF RESIDUALS',/,T28,'ABSOLUTE VALUE',
      $    ' OF LARGEST RESIDUAL = 1',/,' OBS  -1',T47,'0',T87,'1')
       LINCNT=LINCNT+4
       MAXRES=0.0
       DO 10 I=1,ND
       TEMP=ABS(Q(I))
10     IF(TEMP.GT.MAXRES)MAXRES=TEMP
       DO 20 I=1,ND
       TEMP=Q(I)/MAXRES
       N=47+(TEMP/0.025)
       WRITE(6,902) I,N
902    FORMAT(X,I3,X,']',T47,']',TN,'*',T88,']')
       LINCNT=LINCNT+1
       IF(LINCNT.NE.LINMAX) GO TO 20
       CALL PAGER
       WRITE(6,901)
       LINCNT=LINCNT+4
20     CONTINUE
       RETURN
       END
```

LITERATURE CITED

1 Broyden, C. G. "A Class of Methods for Solving Nonlinear Simultaneous Equations." Math. Comp., XXI (1965), 368-381.

2 Chipman, J. S. "On Least Squares with Insufficient Observations." J. Amer. Stat. Assoc., LIX (1964), 1078-1111.

3 Draper, N. R., and Smith, H. Applied Regression Analysis. New York: John Wiley & Sons, Inc. 1966.

4 Fletcher, R. "A Modified Marquardt Subroutine for Nonlinear Least Squares." Harwell Report, AERE, R. 6799. 1971.

5 Hartley, H. D. "The Modified Gauss-Newton Method for the Fitting of Nonlinear Regression Functions by Least Squares." Technometrics, III, No. 2 (1961), 269-280.

6 Hoerl, A. E. "Application of Ridge Analysis to Regression Problems." Chem. Eng. Prog., LVIII (1962), 54-59.

7 _____, and Kennard, R. W. "Ridge Regression. Biased Estimation for Nonorthogonal Problems." Technometrics, XII, No. 1 (1970), 55-67.

8 _____. "Ridge Regression. Applications to Nonorthogonal Problems." Technometrics, XII, No. 1 (1970), 68-82.

9 Kizner, W. "A Numerical Method for Finding Solutions of Nonlinear Equations." J. Soc. Indust. Appl. Math, XII (1964), 424-428.

10 Levenberg, K. "A Method for the Solution of Certain Nonlinear Problems in Least Squares." Quar. Appl. Math, II (1944), 164-168.

11 Marquardt, D. W. "An Algorithm for Least Squares Estimation of Nonlinear Parameters." J. Soc. Indust. Appl. Math, XI (1963), 431-441.

12 _____. "Generalized Inverse, Ridge Regression, Biased Linear Estimation and Nonlinear Estimation." Technometrics, XII, No. 3 (1970), 591-612.

13 _____, and Stanley, R. H. "NLIN2-Least Squares Estimation of Nonlinear Parameters." Supplement to SDA 3094 (NLIN), mimeo manuscript.

14 Ortega, J. M., and Rheinboldt, W. C. Iterative Solution of Nonlinear Equations in Several Variables. New York: Academic Press. 1970.

15 Silvey, S. D. "Multicollinearity and Imprecise Estimation." J. Roy. Stat. Soc., B, XXXI, No. 3 (1969), 539-552.

16 Theil, H. Principles of Econometrics. New York: John Wiley & Sons, Inc. 1971.

Yeh, Ning-Chia
Numerical solution of nonlinear regressions
...