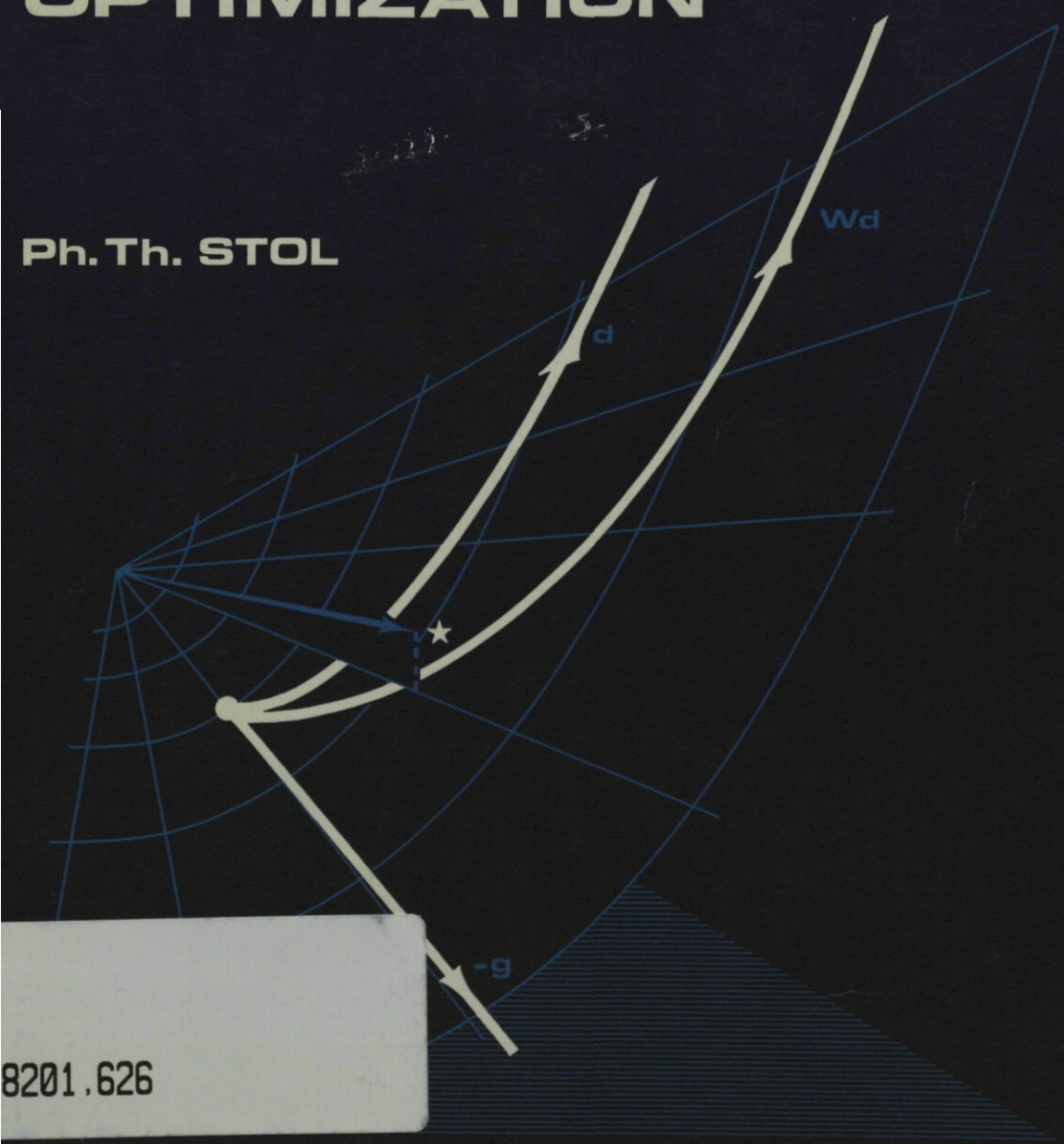
a contribution to theory and practice of

NONLINEAR PARAMETER OPTIMIZATION

Ph. Th. STOL



Ph. Th. Stol.

A contribution to theory and practice of nonlinear parameter optimization

Proefschrift
ter verkrijging van de graad van
doctor in de landbouwwetenschappen,
op gezag van de rector magnificus,
dr. ir. J. P. H. van der Want, hoogleraar in de virologie,
in het openbaar te verdedigen
op woensdag 28 mei 1975 des namiddags te vier uur
in de aula van de Landbouwhogeschool te Wageningen

Centre for Agricultural Publishing and Documentation

Wageningen – 1975

Abstract

STOL, PH. TH. (1975) A contribution to theory and practice of nonlinear parameter optimization. Doctoral thesis, Wageningen. 197 p., 19 tables, 47 figs, 56 refs, 2 appendices, Eng. and Dutch summaries.
Also: Agric. Res. Rep. (Versl. landbouwk. Onderz.) 835.

Nonlinear parameter optimization in least squares was studied from a point of view of differential geometry. Properties of curvilinear coordinates, scale factors and curvature were investigated. Parameters of the condition function were expressed as functions of algorithm parameters to generalize the formulas. The analysis of the convergence process cumulated in the development of procedures that accelerate convergence. Scale factors were used as weights to the differential correction vector to improve the direction of search. A method to correct for curvature, called back projection method, was developed. Use was made of the tangent plane on which the path of search on the fitting surface was projected. Deviations from the original direction were corrected by optimizing the angle of deviation and the step factor. The correspondence between rate of convergence and curvature of the path of search was illustrated with an example. A small geodesic curvature at the starting point indicates fast convergence. Curvature properties of the parametric curves appeared to be of more influence than those of the fitting surface. To avoid heavy oscillation of intermediate parameter values a method was developed that required the intermediate points to be the foot of a perpendicular from the terminal point of intermediate observation vectors thus producing paths of controlled approach. Since condition functions may have a complicated structure in that they can be implicit functions, sequential functions or can consist of mathematical models involving alternative functions, it was treated how first derivatives can be calculated and programmed systematically for these functions. Methods introduced were made operational by means of a FORTRAN program. A description of the use of the subprograms and instructions to modify the main program to suit the various algorithms and procedures developed are given in the Appendices.

UDC 519.281.2 : 518.62 : 516.8 : 681.32.06 and 513.736.7 : 513.735.43 : 518.5

This thesis will also be published as Agricultural Research Reports 835.

Printed by H. Veenman en Zonen B.V. – Wageningen

Stellingen

I

Alvorens over te gaan tot het doen van statistische uitspraken over uitkomsten van niet-lineaire regressieberekeningen dienen de drie stadia van linearisering van het vereffeningsoppervlak afzonderlijk te worden onderzocht.

Dit proefschrift.

II

De door Levenberg en later door Davies en Whitting gegeven geometrische interpretatie van het gebruik van gedempte normaalvergelijkingen is niet volledig zonder gebruik van het respons oppervlak.

Levenberg, K., 1944. A method for the solution of certain nonlineair problems in least squares. Q. appl. Math. 2: 164–168.

Davies, M. & I. J. Whitting., 1971. A modified form of Levenberg's correction. In: F. A. Lootsma (ed.). Numerical methods for non-linear optimization. Academic Press, London and New York. p. 191–201.

III

Het zonder meer gebruiken van alle dagen waarop geen neerslag is geregistreerd als dagen met een neerslagsom gelijk nul, doet geen volledig recht aan het opvatten van de neerslag als kansvariabele.

IV

Indien bij het bepalen van de verwachte duur van overschrijding van afvoertoppen van kleine rivieren uit de negatief exponentiële kansverdelingen van de dagafvoer, respectievelijk de topafvoer, geen rekening wordt gehouden met het stochastisch karakter van het optreden van topafvoeren, zal een te grote waarde voor deze grootheid worden gevonden.

Werkgroep Afvloeiingsfactoren, 1970. Tweede interimrapport, 84 p.

Stol, Ph. Th., 1973. Het gebruik van de negatief-exponentiële kansverdeling voor afvoergegevens en voor het bepalen van de maatgevende afvoer. Deel II: Toepassing op afvoergegevens van de Oude IJssel te Doesburg. ICW Nota 733: 42 p.

V

Bij het toepassen van een Thiessen-netwerk voor het bepalen van de gemiddelde gebiedsneerslag moet de kans dat neerslagmeting op de juiste punten plaats vindt gelijk aan nul worden gesteld.

Colenbrander, H. J. & Ph. Th. Stol., 1970. Neerslag en neerslagverdeling naar plaats en tijd. Deelrapport 5 in: Hydrologisch onderzoek in het Leerinkbeekgebied. Prov. Geld. Arnhem. p. 89–108.

VI

Alvorens over de in de praktijk gevonden samenhang tussen (a) de onderlinge afstand tussen neerslagwaarnemingsstations en (b) de correlatie tussen neerslaghoeveelheden in dezelfde tijdvakken op die stations gemeten, een oordeel uit te spreken is het noodzakelijk een analyse met behulp van verschillende neerslagmodellen uit te voeren.

Hutchinson, P., 1970. A contribution to the problem of spacing raingauges in rugged terrain. J. of Hydrology, 12: 1–14.

Stol, Ph. Th., 1972. The relative efficiency of the density of raingauge networks. J. of Hydrology, 15: 193–208.

VII

De motivering die heeft geleid tot de wijze van operationeel maken van metingen van grootheden dient als vast gegeven bij de in een data base verzamelde meetuitkomsten te worden opgeslagen en bij het verstrekken van deze te worden vermeld.

VIII

Gegevens aanwezig op maalstaten van poldergemalen dienen voor automatische verwerking toegankelijk te worden gemaakt en gehouden.

Stol, Ph. Th., 1971. Het vaststellen van afvoercoëfficiënten van poldergemalen met behulp van maalstaatgegevens. Waterschapsbelangen, 56: 303–306 en 323–326.

IX

Het is niet juist een ééndimensionale deeloplossing in een probleem waarin meer dan één parameter geoptimaliseerd moet worden als lokaal minimum op te vatten.

Aird, T., 1973. Computational solution of global nonlinear least squares problems. Ph. D. Thesis. Purdue University, West Lafayette, Indiana: 120 p.

X

Aangezien plaatsnaamborden op veel spoorwegstations niet onder de optimale hoek zijn opgesteld worden aldaar ter plaatse bekende wachtende reizigers bevoordeeld boven in treinen passerende vreemdelingen.

Proefschrift van Ph. Th. Stol
A contribution to theory and practice of nonlinear parameter optimization.

Preface

It is with great pleasure that I take the opportunity to express my appreciation to all who have contributed to the completion of this book.

I am most grateful to Prof. Dr. B. van Rootselaar for the many stimulating discussions and his valuable gifts of time and knowledge.

I am greatly indebted to Dr. Ir. C. van den Berg, Director of the Institute for Land and Water Management Research, for the privilege of offering me the opportunity to work on the here presented research and his stimulating personal interest in the progress made.

The subjects dealt with have benefited from the many nonlinear functions produced by Ir. W. C. Visser. I am grateful for his mathematical interest which stimulated me to generalize several of the proposed methods.

I express my sincere gratitude to Ir. E. W. Schierbeek MSF for helpful criticisms and the suggested better methods of presentation which highly clarified the structure of the book and improved the expository parts.

Greatly appreciated is the work done by Miss M. G. van Steenbergen who, assisted by Mrs. G. van den Berg-Buitenhuis and Mr. J. R. Maassen, calculated and tested parts of the algorithms before these could definitely be inserted in the program: Thanks are also due to Mr. W. van Doorne and Ing. J. B. H. M. van Gils for assistance in programming techniques and system analysis.

Special mention must be made of the assistance and hospitality of the Technical and Scientific Computing Centre of the Government Service for Land and Water Use, supervised by Ing. R. J. Bos, where Ing. G. Klok and Ing. R. H. Meiborg introduced me to operating systems and to the Centre's do-it-yourself system which makes the development and testing of programs most efficient and enjoyable.

I am indebted to Mrs. L. F. van Eijck-Voorham. It was stimulating to experience how handwritten notes, bristling with formulas, developed in a very short time into a complete manuscript by her professional typing skills.

I am grateful to Mr. M. Jansen who, under supervision of Mr. T. Klaassen, made the drawings and gave valuable suggestions for improving their illustrating purposes.

I wish to thank the Centre for Agricultural Publishing and Documentation for publishing the manuscript in its series 'Agricultural Research Reports' and Mr. R. J. P. Aalpol for his editorial remarks.

My greatest personal indebtness is undoubtedly due to my wife Mineke who encouraged this whole undertaking. Her patience and understanding for my absentmindedness in other matters made it possible that I remained inspired from start to

finish. Finally my appreciation to Janneke, Wouter and Philine who understood the necessity of their father doing his own homework during the completion of this study.

Curriculum vitae

De auteur werd op 27 november 1924 te Haarlem geboren. Na het beëindigen van zijn studie in de Cultuurtechniek aan de Landbouwhogeschool te Wageningen in 1952 werd hij aangesteld als provinciaal ingenieur voor Zuid-Holland van de Commissie Onderzoek Landbouwwaterhuishouding Nederland – TNO (COLN). Bij de afsluiting van dit onderzoek volgde in 1956 zijn aanstelling bij het Instituut voor Cultuurtechniek en Waterhuishouding (ICW) te Wageningen als hoofd van de afdeling Wiskunde. In deze hoedanigheid maakte hij studiereizen naar onder andere Canada en de Verenigde Staten. Gedurende enkele maanden in de jaren 1965 en 1973 trad hij voor de Economic Commission for Asia and the Far East (ECAFE) te Bangkok in enkele landen in het Verre Oosten op als docent in een aantal seminars over het gebruik van hydrologische en meteorologische gegevens en de toepassing van computers in de hydrologie. Sinds 1966 geeft hij een cursus statistiek aan de International Course in Hydraulic Engineering te Delft en sinds 1972 een cursus in de analyse van neerslaggegevens aan de Post-Graduate Course on Land Drainage te Wageningen. De auteur is algemeen adviseur van de Afdeling voor Technisch en Wetenschappelijk Rekenwerk van de Cultuurtechnische Dienst te Utrecht en voorzitter van de Contactgroep Archivering en Automatische Verwerking Hydrologische Gegevens TNO.

Contents

1 Introduction

1.1 General

Research workers who describe their problems with mathematical formulas in which variables and unknown parameters occur, have need for testing their working hypothesis with the aid of observations. But then they face a new problem, that of finding the best value of the parameters in their models.

With the aid of a test criterion the term 'best' can be made operational. The problem that now arises is a complicated one since most models consist of functions that are nonlinear in the parameters and iterative methods have to be employed to find 'best' or optimal values for them. Although computers work so fast that execution of some 100 iteration steps is acceptable for rather small-scale problems, there still are reasons to search for faster techniques. Execution of series of data on routine basis is one of them. Application of complicated models, with many variables and parameters, to a large number of observations requiring a manifold of a reasonable number of iterations, is a second.

These practical reasons necessitate a theoretical treatment to gain insight in optimization processes. Therefore, in the present study a theoretical approach has been made leading to a computer program in which newly developed optimization algorithms are incorporated.

Simplifying a complicated optimization procedure by plotting the test criterion or response as contour curves in a graph with two parameters as variables, apparently gives an indication of how to find the extremum starting from an arbitrarily chosen initial point. For most fitting problems this way of representation is inadequate, however. Least squares techniques have the advantage that except for the above indicated parameter space, use can be made of the space in which observed and calculated function values are plotted. In this observation space the function to be fitted is represented by a surface covered with a curvilinear coordinate system. The mathematical tools to investigate such surfaces are treated in Chapter 2. Analysis of the curvilinear coordinate system and its role in the search for more efficient paths on the fitting surface leads to new algorithms (Chapter 5), that produce paths closer to the final solution (Fig. 1). Other techniques that serve the same purpose are more empirically based on intermediate results in the tangent plane to the fitting surface and on results obtained along the axes of the parameter space (Chapter 6 and 10).

Main notions that are of importance in optimization techniques are briefly reviewed

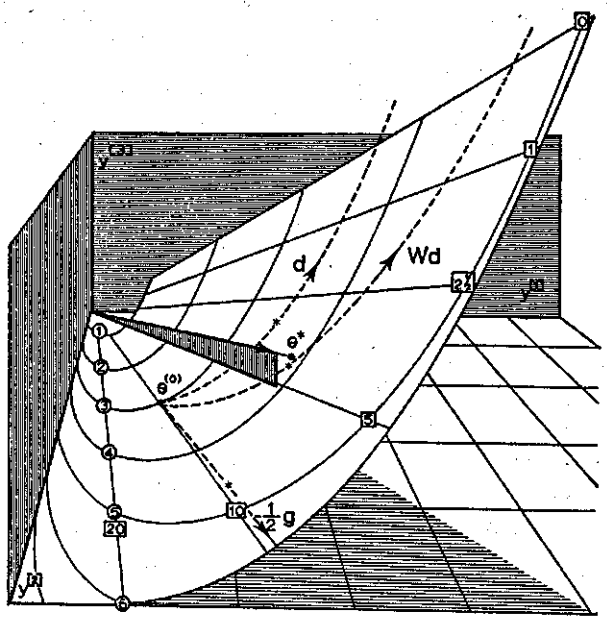


Fig. 1. Perspective drawing of an optimization problem with three directions of search. Given the condition function $y = a \exp(-bx^2)$ it is asked to optimize $\theta = (a,b)^T$ given the observation matrix $X = (y,x)$ where $y = (2.50, 3.80, 1.50)^T$ and $x = (0.3, 0.1, 0.5)^T$. The starting value is $\theta^{(0)} = (3, 10)^T$. In the figure three gradient directions are drawn. The steepest descent $(-\frac{1}{2}g)$ decreases the objective function (sum of squares) from $S(\theta^{(0)}) = 4.39$ to $S = 1.69$ with step factor $\lambda^* = 1.0$ in the figure denoted by an asterisk on the path. The modified Gauss-Newton method, operating with differential corrections d, decreases the response S to a lower value, viz. $S = 0.23$ taking the step factor in this direction $\lambda^* = 0.6$. Finally the path produced by weights assigned to d has been drawn. The weights w^D are obtained from differentials of scale factors (Chapter 5) with the aid of equation (5.3.3) and taken independent of λ. Along this path of search the response is reduced to $S = 0.06$, with step factor $\lambda^* = 1.2$. This path proceeds close to the final solution $\theta^* = (3.87, 4.09)^T$ with $S(\theta^*) = 0.05$.

in Chapter 3. Stress is laid on least squares methods since they are used in this study as the test criterion.

In Chapter 4 it is shown that the function subject to such fitting procedures not necessarily need be explicit but that the algorithms that use first derivatives can also be applied to a variety of functions including branched models of sequential functions, as those used for the analysis of time series with deterministic models.

The efficiency of iterative methods can be improved by choosing good starting approximations to the solution of the parameter values. It is reasonable to require these starting values from the research worker who developed the condition function. Optimization techniques, however, can lead to paths of progress in the convergence process that swing over the fitting surface before the terminal point is reached. The research worker then may require that intermediate solutions will not move to undesirable directions. For this reason in Chapter 11 an algorithm is developed that produces a fit along controlled paths to avoid heavy oscillation of intermediate solutions.

A subproblem is the determination of the minimum response in a given direction of search. Algorithms can fail when this subproblem is not adequately solved. Complications that arise from the special structure of the condition function often are detected during the execution phase of this subproblem. Special attention has therefore been given to the determination of the minimum response in one-dimensional search (Chapter 9).

To meet the demands of practice, a computer program has been developed that consists of subroutine subprograms that can be linked by a main program whose default deck structure can be modified to perform specific procedures, as for example the investigation of the properties of the function to be fitted (Chapters 8 and 12), as well as for the investigation of the properties of the applied algorithm and of the convergence process itself. This according to the field of interest of the user of the program (Chapter 13). To avoid time consuming programming of new parameter optimization problems the program is set up in such a manner that in principle only the new condition function and its derivatives with respect to the parameters, need be programmed (Chapter 7 and Appendix 1.4.1).

Some examples illustrate the developed algorithms. Examples of a hypothetical nature elucidate specific properties of fitting procedures. Formulas and data from research practice serve the same goal, without giving an opinion on their value.

The Appendices contain the complete program, a technical description to it, an example of the default output and the most important update instructions.

1.2 Objective

We are concerned with a function F, the condition function, of m real variables x ordered in a $1 \times m$ row vector of variables

$$x = (x_1,\ldots,x_j,\ldots,x_m)$$

and of p real parameters θ ordered in a $p \times 1$ column vector of parameters

$$\theta = (\theta_1,\ldots,\theta_k,\ldots,\theta_p)^T$$

which has to be optimized – in a numerical sense – or estimated – in a statistical sense – on basis of v ($v > p$) observations ordered in a $v \times m$ matrix of observations

$$X = \begin{bmatrix} x_1^{[1]},\ldots,x_j^{[1]},\ldots,x_m^{[1]} \\ \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \\ x_1^{[i]},\ldots,x_j^{[i]},\ldots,x_m^{[i]} \\ \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \\ x_1^{[v]},\ldots,x_j^{[v]},\ldots,x_m^{[v]} \end{bmatrix}$$

We assume that at least one of the parameters occurs nonlinearly in the condition function F and write

$$F(x,\theta) = F(x_1,\ldots,x_m,\theta_1,\ldots,\theta_p)$$

If $F = 0$ can be solved with respect to x_j, $1 \leqq j \leqq m$, producing function values y_j, it is asked to fit the observed values x_j, requiring the objective function S, giving the response

$$S_j(\theta) := \sum_{i=1}^{v} [x_j^{[i]} - y_j^{[i]}(\theta)]^2 \tag{1.2.1}$$

to be minimized with respect to θ, so it is asked to find

$$\min_{\theta} [S_j(\theta)]$$

which is supposed to be solved at $\theta =: \theta^*$.

1.3 Conditions

It is assumed that the condition function F has continuous first and second derivatives with respect to θ_k, $k = 1(1)p$, for $x_1^{[i]},\ldots,x_m^{[i]}$, $i = 1(1)v$.

The condition function may be either explicit or implicit and can be determined for all feasible choices of θ, F being subject to constraints of this type only. Because of (1.2.1) the same conditions hold for the objective function S_j.

Vectors of observed or calculated function values are elements of the v-dimensional Euclidean space, the observation space. Parameter vectors are elements of the p-dimensional Euclidean parameter space so $x \in E^v$ and $\theta \in E^p$. The norm in these spaces is defined by the Euclidean norm $\|x\| = (x^T x)^{\frac{1}{2}}$ and $\|\theta\| = (\theta^T \theta)^{\frac{1}{2}}$ respectively.

1.4 Terminology

The optimization process is an iterative process. Once n fitting cycles have been produced, the next updating of the parameter vector θ can be obtained from the iteration

$$\theta^{(n+1)}(\lambda) = \theta^{(n)} + \lambda s^{(n)} \tag{1.4.1}$$

where λ is a step factor and s a $p \times 1$ vector that produces a direction of search in the parameter space. The step length is given by $\lambda\|s\|$. If λ is determined such that

$$S_j(\theta^{*(n)}) := \min_{\lambda}[S_j(\theta^{(n+1)}(\lambda))]$$

the updating is called optimal for $\theta^{(n+1)} = \theta^{*(n)}$. The vector s is obtained by application of a particular algorithm. The process (1.4.1) terminates under control of stopping

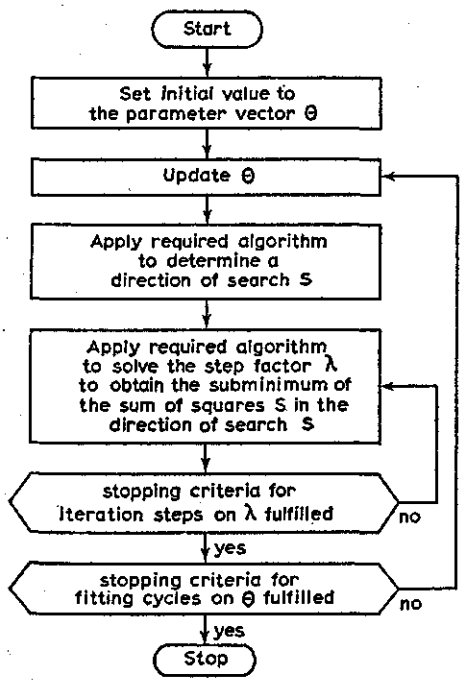


Fig. 2. Typical flowchart of an algorithm for one-dimensional optimal search in nonlinear optimization.

criteria at $\theta^{(t)} :=$ the θ for which all stopping criteria in the numerical process are fulfilled.

Convergence of the algorithm takes place if

$$\lim_{t \to \infty} \theta^{(t)} = \theta^*$$

Because the entire process consists of several iterative procedures, a distinction is made between algorithm fitting cycles and iteration steps as given in Fig. 2.

I Theory

2 Functions, vectors and spaces

2.1 Functions and vectors

Functions to be considered can generally be written in the implicit form

$$F(x,\theta) = 0 \tag{2.1.1}$$

The variables x are supposed to represent observable quantities. The matrix of observations X is to be interpreted as v row vectors of observed values, each with respect to m variables, where superscripts refer to observations, so

$$X = (x^{[1]},\ldots,x^{[i]},\ldots,x^{[v]})^T$$

It is convenient to consider X occasionally to be composed of column vectors

$$X = (x_1,\ldots,x_j,\ldots,x_m)$$

$F(x,\theta)$ is particularized for the observed values of the variables and for specified parameter values, e.g.

$$x^{[i]} = (x_1^{[i]},\ldots,x_j^{[i]},\ldots,x_m^{[i]}) \tag{2.1.2}$$

and

$$\theta^{(n)} = (\theta_1^{(n)},\ldots,\theta_k^{(n)},\ldots,\theta_p^{(n)})^T \tag{2.1.3}$$

Sequences of parameter values up to a terminal vector obtained in a sequence of fitting cycles, are represented by (2.1.3) where $n = 1(1)t$. This superscript will also be used for other vectors, matrices and scalars when referring to a particular fitting cycle.

For the ith vector of observations to the m variables and the nth vector of p parameter values we define

$$F^{[i]}(\theta^{(n)}) := F(x^{[i]},\theta^{(n)}), \qquad i = 1(1)v \tag{2.1.4}$$

In general, the condition of zero function value is not met by inserting (2.1.2) and (2.1.3) in (2.1.1). However, in all cases to be dealt with it is assumed that (2.1.1) can be solved with respect to the jth variable, now denoted by y_j, to obtain roots of F that can be represented by v condition equations

$$F^{[i]}(y_j,\theta^{(n)}) := F(x_1^{[i]},\ldots,x_{j-1}^{[i]},y_j,x_{j+1}^{[i]},\ldots,x_m^{[i]},\theta^{(n)}) = 0 \tag{2.1.5}$$

For any θ the relationship between implicit and explicit functions can thus be given by

$$F^{[i]}(y_j,\theta) = 0 \leftrightarrow y_j^{[i]} =: f_j^{[i]}(\theta) \tag{2.1.6}$$

It is assumed that solutions $y_j^{[i]}$, $i = 1(1)v$, can be obtained from (2.1.6) by either iterative methods or by simple evaluation. Numerical solutions $y_j^{[i]}$ are components of a new $v \times 1$ solution vector

$$y_j = f_j(\theta) \tag{2.1.7}$$

A $v \times 1$ vector of function values depends according to (2.1.4) on all observable magnitudes and is given by

$$F(X,\theta) := [F(x^{[1]},\theta),\ldots,F(x^{[v]},\theta)]^T$$

where, in general, $F(X,\theta) \neq 0$. From this definition it can be seen that the components of F do not depend on all observations simultaneously. Consequently the solution vector is obtained from

$$F(y_j,\theta) = [F^{[1]}(y_j^{[1]},\theta),\ldots,F^{[v]}(y_j^{[v]},\theta)]^T = 0 \tag{2.1.8}$$

where the ith component, $i = 1(1)v$, is a function of the ith component of the vector y_j alone.

Definition 2.1.1.: A vector function $v(u)$ is called a strict function of u if $v_i = v_i(u_i)$ for all i.

Instead of the parameter vector θ itself, functions of algorithm parameters will often be used to obtain the appropriate parameter values. These functions will be called parameter functions. They consist of scalars and vectors. Frequently use will be made of the scalar λ and the vector s. A parameter function in its general form reads

$$\theta(\lambda) := [\theta_1(\lambda,s),\ldots,\theta_p(\lambda,s)]^T$$

A strict parameter function is given by

$$\theta(\lambda) = [\theta_1(\lambda,s_1),\ldots,\theta_p(\lambda,s_p)]^T$$

A further special case is the linear and strict parameter function

$$\theta(\lambda) = \theta^{(n)} + \lambda s^{(n)} \tag{2.1.9}$$

which is sketched in Fig. 3. The locus of the terminal point of the vector θ in this case is a straight line in the parameter space. Exploration of the response surface (Fig. 5) along such a locus is called exploration along a one-dimensional path of search.

Solutions of (2.1.6) when subtraced from the corresponding observations produce the $v \times 1$ difference vector f_0

$$x_j - y_j =: f_{0j}(\theta) = [f_{0j}^{[1]}(\theta),\ldots,f_{0j}^{[v]}(\theta)]^T \tag{2.1.10}$$

Without loss of generality the chosen variable can be designated by $j = 1$.

The $p \times 1$ differential vector $d\theta$ is defined by

$$d\theta := (d\theta_1,\ldots,d\theta_k,\ldots,d\theta_p)^T$$

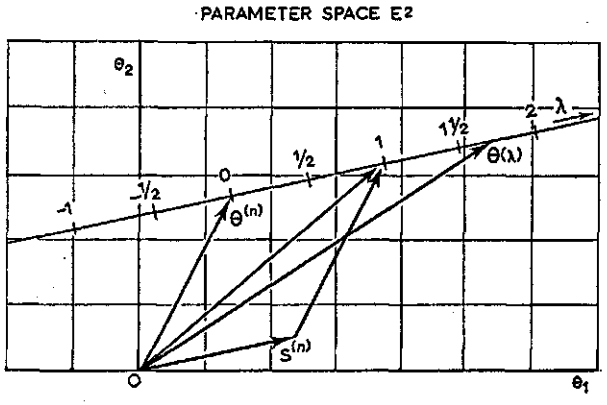


Fig. 3. Parameter space with a linear and strict parameter function $\theta(\lambda) = \theta^{(n)} + \lambda s^{(n)}$.

Differentiation with respect to each of the parameters is denoted by the vector differential operator ∇, defined as the $p \times 1$ vector

$$\nabla := (\frac{\partial}{\partial\theta_1},\ldots,\frac{\partial}{\partial\theta_p})^T =: (\partial_1,\ldots,\partial_p)^T$$

which acts on functions only. Using (2.1.6) without subscript j, we have for $i = 1(1)v$

$$\nabla^T f^{[i]} = (\partial_1 f^{[i]},\ldots,\partial_k f^{[i]},\ldots,\partial_p f^{[i]}) \tag{2.1.11}$$

which will be written

$$\nabla^T f^{[i]} = (f_1^{[i]},\ldots,f_k^{[i]},\ldots,f_p^{[i]}) \tag{2.1.12}$$

The total differential of $f^{[i]}$ reads

$$\begin{aligned} df^{[i]}(\theta) &= f_1^{[i]} d\theta_1 + \ldots + f_p^{[i]} d\theta_p \\ &= \nabla^T f^{[i]} d\theta \end{aligned} \tag{2.1.13}$$

In expressions as $\partial f(\theta^{(n)})/\partial\theta_k$ the order of operation is: differentiate the function f with respect to the kth parameter and then insert the particular value $\theta = \theta^{(n)}$ in the result.

The square of the length of a vector, say v, will be denoted by any of the expressions $\| v \|^2$, (v,v) or $v^T v$. The cosine of the angle between the directions given by u and v is written $\cos(u,v)$, the inner product of these vectors is denoted by (u,v) or $u^T v$.

Finally we define vectors and matrices consisting of numbers only, by e.g. $\mathfrak{Z} := (3,\ldots,3)^T$ and $\mathfrak{Z} := \| 3 \|$, dimensions being defined such that vector and matrix operations can be performed. A unit vector in the direction of the kth coordinate of an orthogonal reference system is denoted by I_k. Obviously $I_k^T v = v_k$ and for any matrix A we have $I_k^T A I_l = A_{kl}$. A unit matrix is denoted by I.

To clarify the structure of arguments of vector functions the right-hand side of

(2.1.9) is sometimes used in full, e.g. $u(\theta^{(n)} + \lambda s^{(n)})$. Composite functions like $u(\theta(\lambda))$ will often be abbreviated to $u(\lambda)$.

2.2 Spaces and curvilinear coordinates

The geometric representation of the vector function

$$y = f(\theta) \tag{2.2.1}$$

where from now on we drop the subscript j, is a p-dimensional surface in the v-dimensional Euclidean observation space E^v ($p < v$). This surface will be called the fitting surface. Its properties depend on the form of the condition function f and on the observed values.

The function

$$y = f(\theta_1,\ldots,\theta_{k-1},\theta_k^{(0)},\theta_{k+1},\ldots,\theta_p)$$

where $\theta_k^{(0)}$ is kept constant, is the locus of a subsurface of the fitting surface. It is the coordinate surface to be enumerated with $\theta_k^{(0)}$.

The function

$$y = f(\theta_1^{(0)},\ldots,\theta_{k-1}^{(0)},\theta_k,\theta_{k+1}^{(0)},\ldots,\theta_p^{(0)}) \tag{2.2.2}$$

where components with superscripts are kept constant, is briefly denoted by $f(\theta_k)$. It is the locus of a space curve in E^v satisfying (2.2.1) and so a curve on the fitting surface.

Each of the parameters produce a space curve according to (2.2.2). These parametric curves can be considered a curvilinear coordinate system on the fitting surface (cf. Struik, 1961). The implicit form reads

$$F(y,\theta_k) = 0, \qquad k = 1(1)p$$

A geometric representation is given in Fig. 4, where the E^v ($v = 3$) space is sketched. Values of $y^{[i]}$ are plotted on an orthogonal v-dimensional reference system of unit vectors of the $v \times v$ unit matrix I.

Vectors in E^v having their terminal point on the fitting surface are called position vectors, e.g. y given by (2.2.1). Position vectors depend on the value of the components of the $p \times 1$ parameter vector θ. This vector is an element of a p-dimensional Euclidean space E^p considering the values θ_k, $k = 1(1)p$, coordinates of an orthogonal $p \times p$ reference system I. Instead of the fitting surface the parameter space E^p will frequently be used, an example is given in Fig. 3.

An arbitrary curve on the fitting surface is given by the vector $f(\theta(\lambda))$ with parameter λ. This can also be considered a one-dimensional subspace that can be plotted on an E^1 parameter space for the algorithm parameter λ.

The sum of squares of deviations $x - y$ according to (1.2.1) can be written with (2.1.10) as

$$S = f_0^T f_0 \tag{2.2.3}$$

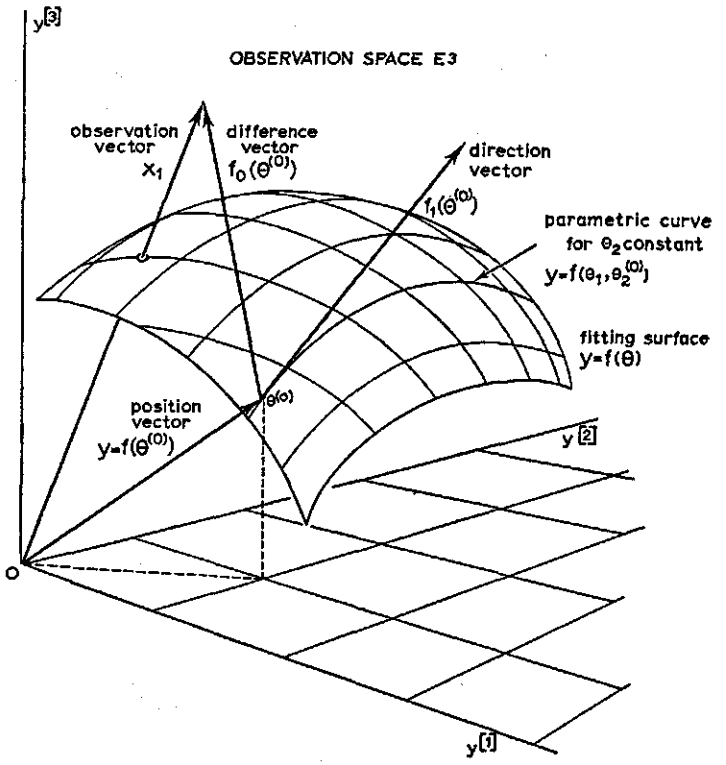


Fig. 4. Observation space and fitting surface, with vectors defined in Sections 2.1 and 2.2 evaluated at $\theta^{(0)}$ on the fitting surface.

and so the problem is the minimization of the square of the Euclidean norm of f_0.

Values of S can be assigned to the terminal point of the appropriate vectors θ. Plotting these values on a further coordinate the function $S = S(\theta)$ can be drawn in a $(p + 1)$-dimensional Euclidean space E^{p+1}, the response space, giving the response surface (Fig. 5). The objective function S generally cannot be considered a quadratic function. It is quadratic in the parameters only if the condition function is linear.

The projection of $S(\theta) =$ constant on the parameter space E^p produces contours in this space. The $p \times 1$ gradient vector $g(\theta) = \nabla S(\theta)$ represents the slope of the $(p + 1)$-dimensional response surface giving the direction of the greatest rate of change of S in the point θ, with magnitude $\| g \|$. The gradient in θ is perpendicular to contour $S =$ constant in space E^p (cf. Struik, 1961). Parameter functions depending on a single algorithm parameter λ produce an E^2 response subspace in E^{p+1}. The steepest descent response subspace along $N = -\frac{1}{2}g$ can be obtained from coordinate axes for S and λ (Fig. 5).

Evaluated for all observations, that means for all functions $f^{[i]}$, $i = 1(1)v$, equation (2.1.12) leads to the $v \times p$ Jacobian matrix

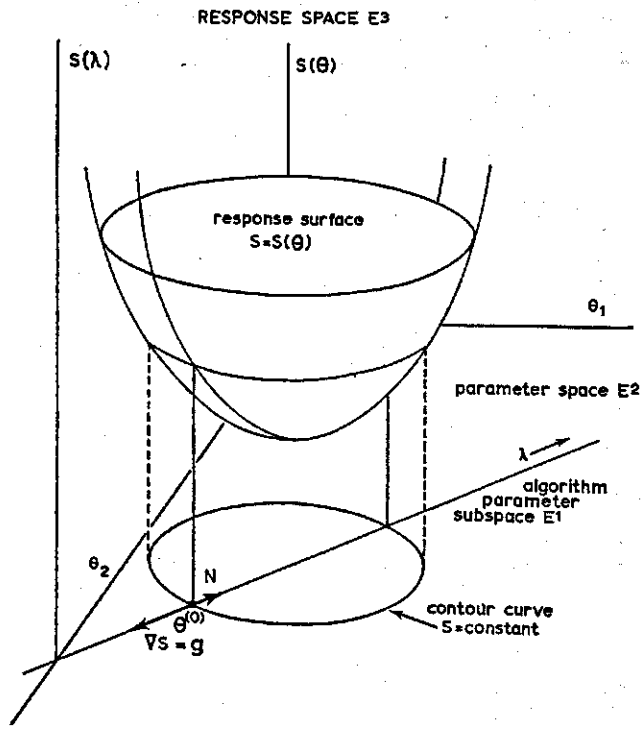


Fig. 5. Response space, response surface and parameter space with the gradient g and the normal $N = -\frac{1}{2}g$ at $\theta^{(0)}$ as defined in Section 2.2. The response space is an extension of the parameter space, see Fig. 3. In a response subspace E^2 with coordinate system $(\lambda, S(\lambda))$ a subminimum can be determined.

$$J_\theta^f := \begin{bmatrix} f_1^{[1]},\ldots,f_k^{[1]},\ldots,f_p^{[1]} \\ \cdot \quad\quad \cdot \quad\quad \cdot \\ f_1^{[i]},\ldots,f_k^{[i]},\ldots,f_p^{[i]} \\ \cdot \quad\quad \cdot \quad\quad \cdot \\ f_1^{[v]},\ldots,f_k^{[v]},\ldots,f_p^{[v]} \end{bmatrix} =: J \tag{2.2.4}$$

where an arbitrary element is given by $J_k^{[i]} = \dfrac{\partial f^{[i]}}{\partial \theta_k}$.

The matrix J consists of all partial derivatives of f with respect to the parameters, evaluated for all observations. So (2.2.4) is the matrix of column vectors

$$J = (f_1,\ldots f_k,\ldots,f_p) \tag{2.2.5}$$

where f_k is a $v \times 1$ dimensional direction vector in E^v tangent to the parametric curve for θ_k in the terminal point of the position vector $f(\theta)$. See Fig. 4. On the other hand J will sometimes be referred to as a matrix of row vectors of gradients of $f^{[i]}$, viz., see (2.1.11),

$$J = [\nabla^T f^{[1]},\ldots,\nabla^T f^{[i]},\ldots,\nabla^T f^{[v]}]^T \tag{2.2.6}$$

The length of the kth direction vector will be denoted by

$$h_k(\theta) := \| \partial_k f(\theta) \| = (f_k, f_k)^{\frac{1}{2}}, \qquad k = 1(1)p \tag{2.2.7}$$

the scale factor. Scale factors are considered components of the $p \times 1$ vector

$$h(\theta) = [h_1(\theta),\ldots,h_k(\theta),\ldots,h_p(\theta)]^T \tag{2.2.8}$$

The squares of these scale factors are the diagonal elements of the square symmetric $p \times p$ matrix $J^T J$.

Definition 2.2.1.: Let U be a $p \times p$ matrix with elements $U_{ij} = 0$ if $i \neq j$ and $U_{ij} = v_i \neq 0$ if $i = j$, then U will be denoted by

$$v^D := (v_1,\ldots,v_p)^D =: U$$

and the inverse matrix by

$$v^d := (\frac{1}{v_1},\ldots,\frac{1}{v_p})^D =: U^{-1}$$

From this definition we observe that $v^d v = I$ so $v = v^D I$ and $v^D v^d = I$.

Unit vectors in the direction of the derivatives are now obtained by

$$J(\theta)\, h^d(\theta) \tag{2.2.9}$$

with dimensions $v \times p$ and $p \times p$ giving for the product $v \times p$ again.

As the cosine of the angle between two vectors is given by the inner product of unit vectors in their directions, the cosine matrix of the direction vectors of J is consequently given by the square of (2.2.9) giving the $p \times p$ cosine matrix

$$C := h^d J^T J h^d \tag{2.2.10}$$

The $p \times 1$ cosine vector c which consists of the cosines of f_0 with each direction vector (Fig. 4) is equal to

$$c := h^d J^T f_0 \| f_0 \|^{-1} \tag{2.2.11}$$

and will be called the vector of partial cosines.

2.3 Linear approximation to the fitting surface

If it is required to approximate $f(\theta^{(1)})$ starting from the position vector $f(\theta^{(0)})$, we can define the $p \times 1$ difference vector $\Delta^{(0)}$ in the parameter space by

$$\theta^{(1)} = \theta^{(0)} + \Delta^{(0)} \tag{2.3.1}$$

which is a linear and strict parameter function.

Taylor expansion of the position vector $f(\theta^{(0)} + \Delta^{(0)})$ gives for any component the expression

$$f(\theta^{(1)}) = f(\theta^{(0)}) + \frac{\partial f(\theta^{(0)})}{\partial \theta_1} \Delta_1^{(0)} + \ldots + \frac{\partial f(\theta^{(0)})}{\partial \theta_p} \Delta_p^{(0)} + \delta(\Delta^{(0)}) \tag{2.3.2}$$

where the remainder $\delta(\Delta^{(0)})$ is defined by (2.3.2). Obviously from (2.3.2) and (2.3.1) $\delta(\theta) = 0$. Using (2.1.11), equation (2.3.2) can be written

$$f(\theta^{(1)}) = f(\theta^{(0)}) + \nabla^T f(\theta^{(0)}) \Delta^{(0)} + \delta(\Delta^{(0)})$$

Evaluated for all observations, so for all functions f, this becomes – making use of (2.2.6) –

$$f(\theta^{(1)}) = f(\theta^{(0)}) + J(\theta^{(0)}) \Delta^{(0)} + \delta(\Delta^{(0)}) \tag{2.3.3}$$

The tangent plane to the fitting surface evaluated at $\theta^{(0)}$ is spanned by the (column) direction vectors of J and is given by the linear expression

$$l(\Delta) := f(\theta^{(0)}) + J(\theta^{(0)}) \Delta$$

where Δ is an arbitrary vector producing on the right-hand side linear combinations of the column vectors of J. The vector l is the $\nu \times 1$ position vector of the tangent plane.

Equation (2.3.3) expresses that without the remainder $\delta(\Delta^{(0)})$ the position vector $f(\theta^{(1)})$ can be approximated by $f(\theta^{(0)})$ and a linear combination of the direction vectors.

Linear approximation with the aid of a tangent plane concerns three important properties of the fitting surface. They are:

- the fitting surface is replaced by a plane,
- the parametric curves are replaced by straight lines having zero curvature, the directions being fixed at their value at $\theta^{(0)}$,
- the scale factors are fixed at their value at $\theta^{(0)}$.

Consequently linear approximation produces in general a nonorthogonal linear coordinate system on the tangent plane. If the condition function itself is linear in θ_k, $k = 1(1)p$, the Jacobian J is constant and

$$y = J\theta \tag{2.3.4}$$

2.4 Optimization condition and normal equations

We consider the optimization problem solved at $\theta^* = \theta^{(i)}$ when $S(\theta)$ according to (2.2.3) is minimum. So, using (2.1.10) and (2.1.7),

$$S(\theta) = [x - f(\theta)]^T [x - f(\theta)] \tag{2.4.1}$$

has to be minimum, hence the gradient of S must vanish to obtain a stationary point θ, for which it is required that

$$g(\theta) = \nabla S(\theta) = 0 \tag{2.4.2}$$

This gradient (Section 2.2) is a $p \times 1$ vector whose kth component reads

$$\frac{\partial}{\partial \theta_k}[x - f(\theta)]^T[x - f(\theta)] =$$

$$[-f_k(\theta)]^T[x - f(\theta)] + [x - f(\theta)]^T[-f_k(\theta)], \qquad k = 1(1)p \tag{2.4.3}$$

which is equal to

$$g_k(\theta) = -2[f_k(\theta)]^T[x - f(\theta)], \qquad k = 1(1)p \tag{2.4.4}$$

Worked out for all parameters the gradient becomes in virtue of (2.2.5)

$$g(\theta) = -2[J(\theta)]^T f_0(\theta) \tag{2.4.5}$$

The terminal $\theta^{(t)}$ of a sequence of parameter vectors $\theta^{(n)}$ producing the desired solution is conditioned by (2.4.5) and (2.4.2) so by

$$[J(\theta^{(t)})]^T[x - f(\theta^{(t)})] = 0$$

The argument of the position vector can be replaced by $\theta^{(t-1)}$, hence with (2.3.1) and (2.3.3) producing the condition

$$[J(\theta^{(t)})]^T[x - f(\theta^{(t-1)}) - J(\theta^{(t-1)})\Delta^{(t-1)} - \delta(\Delta^{(t-1)})] = 0$$

Defining the $p \times p$ matrix

$$M(\theta^{(t)},\theta^{(t-1)}) := [J(\theta^{(t-1)} + \Delta^{(t-1)})]^T J(\theta^{(t-1)}) \tag{2.4.6}$$

we arrive finally at an implicit expression for $\Delta^{(t-1)}$, viz.

$$\Delta^{(t-1)} = M^{-1}[J(\theta^{(t-1)} + \Delta^{(t-1)})]^T[f_0(\theta^{(t-1)}) - \delta(\Delta^{(t-1)})] \tag{2.4.7}$$

Under the assumption that in the arguments of (2.4.6) and (2.4.7) the contribution of $\Delta^{(t-1)}$ to the function values can be neglected, (2.4.7) gives the solution for $\Delta^{(t-1)}$, which reads

$$\Delta^{(t-1)} = M^{-1}J^T f_0$$

where $\delta(\Delta) \simeq \delta(0) = 0$, leaving it understood that evaluation of the right-hand side is with respect to $\theta^{(t-1)}$.

In this and further expressions, M is taken to be the square symmetric matrix

$$M(\theta) = [J(\theta)]^T J(\theta) \tag{2.4.8}$$

It will be convenient to introduce the $p \times 1$ vector

$$N(\theta) := [J(\theta)]^T f_0(\theta) \tag{2.4.9}$$

which is called the normal. It is related to the gradient g by (2.4.5) so

$$g(\theta) = \nabla S(\theta) = -2N(\theta) \tag{2.4.10}$$

hence giving the direction of greatest rate of change for decreasing function values of

S. N gives the direction of steepest descent in the parameter space (see Fig. 5).

Using the new symbols the normal equations are given by

$$M\Delta^{(t-1)} = N \tag{2.4.11}$$

where

$$M = \begin{bmatrix} (f_1,f_1),\ldots,(f_1,f_p) \\ \vdots \quad\quad \vdots \\ (f_p,f_1),\ldots,(f_p,f_p) \end{bmatrix}, \quad N = \begin{bmatrix} (f_0,f_1) \\ \vdots \\ (f_0,f_p) \end{bmatrix} \tag{2.4.12}$$

Because of (2.3.1) the solution of the normal equations is a direction of search to be inserted in (1.4.1) and to be applied according to the scheme in Fig. 2. In this case the algorithm is the unmodified Gauss-Newton method for iterative solution of $\theta^{(t)}$ with differential corrections d as direction of search, obtained by

$$d = M^{-1}N \tag{2.4.13}$$

With the cosine matrix $C = h^d M h^d$ from (2.2.10) the solution reads

$$d = h^d C^{-1} h^d N$$

The geometric meaning of this optimization procedure is given in Fig. 6. The response S with respect to the fitting surface is given by (f_0,f_0). The length of the difference vector is minimum at $\theta^{(t)}$ if $f_0(\theta^{(t)})$ is perpendicular to all direction vectors at $\theta^{(t)}$, so

$$f_0(\theta^{(t)}) \perp J(\theta^{(t)}) \tag{2.4.14}$$

As $\theta^{(t)}$ is essentially unknown, evaluation of (2.4.14) is with respect to a known parameter vector $\theta^{(t-1)}$ say. The exact condition then reads

$$f_0(\theta^{(t-1)}) - J(\theta^{(t-1)})d^{(t-1)} - \delta(d^{(t-1)}) \perp J(\theta^{(t)}) \tag{2.4.15}$$

where Jd is a linear combination of vectors that span the tangent plane A, producing the foot of the perpendicular from the terminal point of the observation vector x on A, see Fig. 6. The remainder vector δ is the difference vector of the projection of x on A and on the fitting surface respectively.

The vector Jd will be called the total tangent, the difference vector to the tangent plane A will be denoted by ${}^A f_0$. Analogous to (2.2.3) the sum of squares with respect to the tangent plane is defined by

$${}^A S =: ({}^A f_0)^T ({}^A f_0) \tag{2.4.16}$$

The cosine of the angle α between the observation vector x and the position vector y will be called the multiple cosine. The cosine of the angle β between the difference vector f_0 and the total tangent Jd will be called the total cosine. The cosine of the angles between the difference vector f_0 and the direction vectors f_k will be called partial

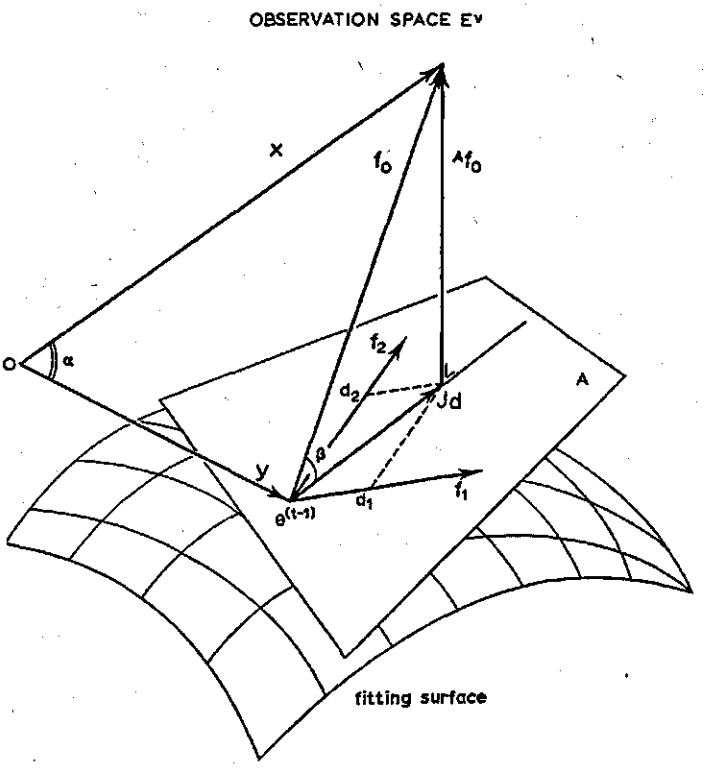


Fig. 6. Fitting surface and tangent plane A. The tangent plane is spanned by the column vectors of J: the direction vectors f_1 and f_2. The difference between the observation vector x and the position vector y – the difference vector f_0 – is projected on the tangent plane giving the vector Jd, which is a linear combination of the direction vectors. The difference vector with respect to A is denoted by ${}^A f_0$.

cosines. They are components of the vector c in (2.2.11).

If the difference between the fitting surface and the linear approximation is sufficiently small, all vectors can be evaluated at $\theta^{(t-1)}$ which yields instead of (2.4.15)

$$f_0(\theta^{(t-1)}) - J(\theta^{(t-1)})d^{(t-1)} \perp J(\theta^{(t-1)})$$

and normal equations according to (2.4.11) through (2.4.13) are found again.

In the linear case of (2.3.4), starting at $\theta = 0$, the normal is equal to $N = J^T x$ and the minimization of $\|x-y\|^2$ leads to

$$\theta = M^{-1} J^T x$$

which gives the solution in one step because the matrices on the right-hand side do not depend on θ.

2.5 Useful derivatives

In further sections use will be made of first and second derivatives of functions of parameters and algorithm parameters. Basic formulas are derived in this Section.

Let a $q \times 1$ vector of algorithm parameters be

$$\lambda := (\lambda_1,\ldots,\lambda_j,\ldots,\lambda_q)^T, \qquad q < p$$

The parameter function $\theta(\lambda)$ then is a function of q algorithm parameters. Now let the $q \times 1$ vector differential operator with respect to each of these parameters be

$$\nabla' := (\frac{\partial}{\partial\lambda_1},\ldots,\frac{\partial}{\partial\lambda_j},\ldots,\frac{\partial}{\partial\lambda_q})^T \tag{2.5.1}$$

then the $p \times q$ Jacobian reads

$$J_\lambda^\theta = \begin{bmatrix} \frac{\partial\theta_1}{\partial\lambda_1},\ldots,\frac{\partial\theta_1}{\partial\lambda_q} \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \frac{\partial\theta_p}{\partial\lambda_1},\ldots,\frac{\partial\theta_p}{\partial\lambda_q} \end{bmatrix} = (\frac{\partial\theta}{\partial\lambda_1},\ldots,\frac{\partial\theta}{\partial\lambda_q}) \tag{2.5.2}$$

We observe that for one algorithm parameter this Jacobian reduces to $J_\lambda^\theta = \dfrac{d\theta}{d\lambda}$ which for linear and strict parameter functions $\theta = \theta^{(0)} + \lambda s$ again reduces to $J_\lambda^\theta = s$.

First derivatives of the position vector $f(\theta(\lambda))$. An arbitrary element of J_λ^f reads

$$\frac{\partial}{\partial\lambda_j} f^{[i]}(\theta(\lambda)) = \nabla^T f^{[i]} \frac{\partial\theta}{\partial\lambda_j} \tag{2.5.3}$$

Referring to (2.2.6) and (2.5.2) we notice that derivatives for all combinations of i and j can be obtained from the chain rule

$$J_\lambda^f = J_\theta^f J_\lambda^\theta \tag{2.5.4}$$

Second derivatives of the position vector $f(\theta)$. An arbitrary element of the Jacobian reads

$$J_k^{[i]} = \frac{\partial f^{[i]}(\theta)}{\partial\theta_k}$$

The second derivative with respect to the lth parameter is

$$J_{kl}^{[i]} := \frac{\partial^2 f^{[i]}(\theta)}{\partial\theta_l \partial\theta_k}, \qquad l = 1(1)p \tag{2.5.5}$$

The elements $J_{kl}^{[i]}$ for $i = 1(1)v$, $k = 1(1)p$, $l = 1(1)p$ cannot be ordered in a matrix. However, only one special case will be involved in our considerations and therefore we define

$$J_2 := (f_{11},\ldots,f_{1p},f_{22},\ldots,f_{2p},\ldots,f_{pp}) \tag{2.5.6}$$

which is a $v \times \binom{p+1}{2}$ matrix. The length of the vectors in (2.5.6) can be obtained from the $\binom{p+1}{2} \times \binom{p+1}{2}$ matrix

$$M_{22} := J_2^T J_2 \tag{2.5.7}$$

It is assumed that the order of differentiation can be interchanged.

First derivative of the difference vector $f_0(\theta(\lambda))$. This derivative reads

$$\frac{d}{d\lambda} f_0(\theta(\lambda)) = \frac{d}{d\lambda}[x - f(\theta(\lambda))]$$

with (2.5.4) this becomes for a 1×1 algorithm parameter vector

$$= -J\frac{d\theta}{d\lambda}$$

If the parameter function is linear and strict this reduces to

$$\frac{d}{d\lambda} f_0(\theta(\lambda)) = -Js$$

which is the vector opposite to the total tangent vector in Fig. 6 if the direction of search is chosen by $s = d$.

First derivative of the Jacobian $J(\theta(\lambda))$ *multiplied by* f_0. The kth column of (2.2.4) gives with the aid of (2.1.13)

$$\frac{df_k}{d\lambda} = \left(\nabla^T f_k^{[1]}\frac{d\theta}{d\lambda},\ldots,\nabla^T f_k^{[i]}\frac{d\theta}{d\lambda},\ldots,\nabla^T f_k^{[v]}\frac{d\theta}{d\lambda}\right)^T \tag{2.5.8}$$

The scalar product with f_0 produces in the right-hand side

$$\left(f_{k1}^{[1]}\frac{d\theta_1}{d\lambda} + \ldots + f_{kp}^{[1]}\frac{d\theta_p}{d\lambda}\right)f_0^{[1]} + \ldots + \left(f_{k1}^{[v]}\frac{d\theta_1}{d\lambda} + \ldots + f_{kp}^{[v]}\frac{d\theta_p}{d\lambda}\right)f_0^{[v]}$$

Collecting terms that contain $\frac{d\theta_k}{d\lambda}$, $k = 1(1)p$, this becomes

$$\left(f_0^{[1]}f_{k1}^{[1]} + \ldots + f_0^{[v]}f_{k1}^{[v]}\right)\frac{d\theta_1}{d\lambda} + \ldots + \left(f_0^{[1]}f_{kp}^{[1]} + \ldots + f_0^{[v]}f_{kp}^{[v]}\right)\frac{d\theta_p}{d\lambda}$$

hence giving

$$\frac{df_k}{d\lambda} = [(f_0,f_{k1}),\ldots,(f_0,f_{kp})]\frac{d\theta}{d\lambda}$$

Finally, applied to all columns $k = 1(1)p$ of J we obtain the $p \times 1$ vector

$$\left[\frac{d}{d\lambda}J(\theta(\lambda))\right]^T f_0 = N_{02}\frac{d\theta}{d\lambda}$$

where

$$N_{02} := \begin{bmatrix} (f_0,f_{11}),\ldots,(f_0,f_{1p}) \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad \cdot \\ (f_0,f_{1p}),\ldots,(f_0,f_{pp}) \end{bmatrix} \tag{2.5.9}$$

the symbol N_{02} being suggested by the form of the normal given in (2.4.12).

First derivatives of the square of the scale factors $h_k^2(\theta)$. These derivatives reads

$$\frac{dh_k^2(\theta)}{d\theta_l} = \frac{d}{d\theta_l}f_k^T f_k = 2f_k^T f_{kl}, \qquad l = 1(1)p$$

For all k and l the results can be collected in the $p \times p$ matrix

$$M_{12} := \begin{bmatrix} (f_1,f_{11}),\ldots,(f_1,f_{1p}) \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad \cdot \\ (f_p,f_{p1}),\ldots,(f_p,f_{pp}) \end{bmatrix} \tag{2.5.10}$$

the symbol M_{12} being suggested by the form of the matrix M in (2.4.12). This matrix is not symmetrical.

Second derivatives of the response $S(\theta)$. The $p \times p$ matrix with elements

$$G_{kl} := \frac{\partial^2 S(\theta)}{\partial\theta_l\partial\theta_k}, \qquad k = 1(1)p, \qquad l = 1(1)p \tag{2.5.11}$$

is called the Hessian matrix of S. Omitting arguments, the kth component of (2.4.3) gives

$$\frac{\delta}{\partial\theta_l}\frac{\partial}{\partial\theta_k}S = 2f_k^T f_l - 2f_0^T f_{kl}$$

For all k and l the results can be collected in the $p \times p$ matrix

$$G = 2M - 2N_{02} \tag{2.5.12}$$

In the linear case $y = J\theta$ the second derivatives vanish which causes $N_{02} = 0$ and so

$$G = 2M = 2J^T J$$

This means that in this case the Hessian of the response equals two times the square of the Jacobian (cf. e.g. Fletcher, 1969b; Powell, 1972a).

First derivatives of the response $S(\theta(\lambda))$. The vector of first derivatives is obtained by application of (2.5.1) which yields

$$\nabla' S(\theta(\lambda)) = \begin{bmatrix} \nabla^T S(\partial\theta/\partial\lambda_1) \\ \cdot \\ \cdot \\ \cdot \\ \nabla^T S(\partial\theta/\partial\lambda_q) \end{bmatrix}$$

and, in virtue of (2.4.10)

$$= \begin{bmatrix} g^T(\partial\theta/\partial\lambda_1) \\ \cdot \\ \cdot \\ \cdot \\ g^T(\partial\theta/\partial\lambda_q) \end{bmatrix} = -2 \begin{bmatrix} N^T(\partial\theta/\partial\lambda_1) \\ \cdot \\ \cdot \\ \cdot \\ N^T(\partial\theta/\partial\lambda_q) \end{bmatrix} \tag{2.5.13}$$

where g and N depend on $\theta(\lambda)$. The expression (2.5.13) set equal to a $q \times 1$ vector 0 can be written

$$g^T J_\lambda^\theta = 0^T \tag{2.5.14}$$

This is the condition for a minimum in a q-dimensional subspace of E^p ($q \leqq p$). Mostly $q = 1$ and the search one-dimensional. Then we have

$$-2[N(\theta(\lambda))]^T \frac{d\theta}{d\lambda} = 0 \tag{2.5.15}$$

which means that the normal must be orthogonal to the direction of search at $\theta(\lambda)$. For linear and strict parameter functions this reduces to the conditon

$$N(\theta(\lambda)) \perp s$$

Table 1. Summary of matrices containing scalar products of vectors

Used symbol	Arbitrary element	Type	Dimension
S	(f_0, f_0)	scalar	1×1
N	(f_0, f_k)	vector	$p \times 1$
N_{02}	(f_0, f_{kl})	symmetric matrix	$p \times p$
M	(f_k, f_l)	symmetric matrix	$p \times p$
M_{12}	(f_k, f_{kl})	matrix	$p \times p$
M_{22}	(f_{kl}, f_{ij})	matrix	$\binom{p+1}{2} \times \binom{p+1}{2}$

Second derivative of the response $S(\theta(\lambda))$ for linear and strict parameter functions. First we consider $\frac{d}{d\lambda} N(\theta(\lambda))$. Analogous to (2.5.3) we have

$$\frac{d}{d\lambda} N(\theta(\lambda)) = [\nabla^T(f_0, f_1)s, \ldots, \nabla^T(f_0, f_p)s]^T \tag{2.5.16}$$

The derivative of an arbitrary component of N reads, with $f_0 = x - f$

$$\frac{\partial}{\partial \theta_l}[(x - f)^T f_k] = -f_l^T f_k + f_0^T f_{kl}, \qquad l = 1(1)p$$

The first component of the vector in (2.5.16) becomes

$$(-f_1^T f_1 + f_0^T f_{11}, \ldots, -f_1^T f_p + f_0^T f_{1p})s$$

and finally with (2.5.15), (2.4.12) and (2.5.9)

$$\frac{d^2 S(\theta(\lambda))}{d\lambda^2} = -2\frac{d}{d\lambda}[N(\theta(\lambda))]^T s$$

$$= 2s^T(M - N_{02})s \tag{2.5.17}$$

and with the Hessian defined in (2.5.12)

$$\frac{d^2 S(\theta(\lambda))}{d\lambda^2} = s^T G s \tag{2.5.18}$$

A summary of matrices containing scalar products of vectors is given in Table 1.

2.6 Quadratic approximation to the response surface

We assume that the response surface in the neighbourhood of a stationary point can be approximated by a quadratic expression. The Taylor expansion of $S(\theta)$ about $\theta(\lambda)$, where θ is a linear and strict function of Δ, then gives

$$S(\theta + \lambda\Delta) - S(\theta) = \lambda\Delta^T g + \tfrac{1}{2}\lambda^2\Delta^T G\Delta + o(\lambda^3) \tag{2.6.1}$$

The first derivatives are obtained from (2.5.13) and the second derivatives from (2.5.18) and so the expression in least squares problems reads

$$= -2\lambda\Delta^T N + \lambda^2\Delta^T(M - N_{02})\Delta + o(\lambda^3) \tag{2.6.2}$$

At a stationary point θ^* we have $g = 0$ and the matrix G must at least be positive semidefinite to produce a minimum, hence for λ sufficiently small (2.6.1) yields

$$S(\theta^* + \lambda\Delta) \geqq S(\theta^*)$$

In the linear case $y = J\theta$, the response S is a quadratic function of the parameters given by

$$S(\theta + \lambda\Delta) = f_0^T f_0 - 2\lambda\Delta^T J^T f_0 + \lambda^2\Delta^T J^T J\Delta \tag{2.6.3}$$

since N_{02} then equals 0. The Hessian is in this case positive definite because $\frac{1}{2}G$ reduces to M and $\Delta^T M\Delta$ is equal to $\| J\Delta \|^2$, the square of the length of a vector that is a linear combination of the direction vectors. Equation (2.6.3) expresses that $S = \| f_0 - \lambda J\Delta \|^2$, which is also clear from Fig. 6 where now the tangent plane is the linear fitting surface.

2.7 Scale factors and arc length

2.7.1 *Scale factors*

In general the scale factors are functions of all p parameters. Their total differential is thus given by

$$dh_k = \frac{\partial h_k}{\partial\theta_1} d\theta_1 + \ldots + \frac{\partial h_k}{\partial\theta_p} d\theta_p$$

$$= \nabla^T h_k \, d\theta$$

For all scale factors this leads to

$$dh = J_\theta^h \, d\theta$$

An arbitrary element (k,l) of this Jacobian reads

$$\partial_l h_k(\theta) = \frac{\partial}{\partial\theta_l}(f_k^T f_k)^{\frac{1}{2}}$$

giving

$$\tfrac{1}{2}(f_k^T f_k)^{-\frac{1}{2}} 2(f_k^T f_{kl}) = \frac{f_k^T f_{kl}}{h_k} \tag{2.7.1}$$

Using the notation defined in definition 2.2.1 and equation (2.5.10) the total differential of the scale factors is found to be

$$dh = h^d M_{12} d\theta$$

2.7.2 *Arc length*

Regarding arc lengths, let s represent the arc length along the kth parametric curve, then the differential reads

$$ds(\theta_k) = [(\frac{df^{[1]}}{d\theta_k})^2 + \ldots + (\frac{df^{[v]}}{d\theta_k})^2]^{\frac{1}{2}} d\theta_k$$

$$= [f_k^T f_k]^{\frac{1}{2}} d\theta_k = h_k(\theta)\, d\theta_k$$

The arc length between $\theta_k^{(1)}$ and $\theta_k^{(2)}$ is given by

$$s = \int_{\theta_k^{(1)}}^{\theta_k^{(2)}} h_k(\theta)\, d\theta_k \tag{2.7.2}$$

If $s(\lambda)$ represents the arc length along an arbitrary curve on the fitting surface with parameter λ, the differential is

$$ds(\lambda) = [(\frac{\partial f^{[1]}}{\partial \theta_1}\frac{d\theta_1}{d\lambda} + \ldots + \frac{\partial f^{[1]}}{\partial \theta_p}\frac{d\theta_p}{d\lambda})^2 + \ldots +$$

$$(\frac{\partial f^{[v]}}{\partial \theta_1}\frac{d\theta_1}{d\lambda} + \ldots + \frac{\partial f^{[v]}}{\partial \theta_p}\frac{d\theta_p}{d\lambda})^2]^{\frac{1}{2}} d\lambda$$

$$= [(\nabla^T f^{[1]}\frac{d\theta}{d\lambda})^2 + \ldots + (\nabla^T f^{[v]}\frac{d\theta}{d\lambda})^2]^{\frac{1}{2}} d\lambda$$

The sum of these squares can be written

$$\sum_{i=1}^{v} \left(\frac{d\theta}{d\lambda}\right)^T (\nabla f^{[i]})\,(\nabla f^{[i]})^T \frac{d\theta}{d\lambda}$$

where the summation sign acts on the gradients only, thus giving a sum of v matrices with dimension $p \times p$, which according to (2.4.12) equals M. Consequently the arc length between $\theta(\lambda_1)$ and $\theta(\lambda_2)$ is given by

$$s = \int_{\lambda_1}^{\lambda_2} \left\{\left(\frac{d\theta(\lambda)}{d\lambda}\right)^T M(\lambda)\, \frac{d\theta(\lambda)}{d\lambda}\right\}^{\frac{1}{2}} d\lambda$$

Using (2.4.8) the structure of this integral can be made lucid viz.

$$s = \int_{\lambda_1}^{\lambda_2} \| J(\lambda) \frac{d\theta(\lambda)}{d\lambda} \| d\lambda \tag{2.7.3}$$

which again is a linear combination of the direction vectors at λ.

If the parameter function is linear and strict in an arbitrary linear direction of search s, (2.7.3) reduces to

$$s = \int_{\lambda_1}^{\lambda_2} \| J(\lambda) s \| \, d\lambda \tag{2.7.4}$$

Then, if the Jacobian is independent of λ, which is the case in the tangent plane to the fitting surface, (2.7.3) becomes

$$s = \| Js \| (\lambda_2 - \lambda_1) \tag{2.7.5}$$

which intuitively is also clear from Fig. 6. Equation (2.7.5) then expresses the fact that the arc length is equal to the step factor times the length of the total tangent when s is taken equal to d.

If, finally, all parameters are kept constant, except the kth, by taking $s = I_k$, (2.7.4) becomes

$$s = \int_{\lambda_1}^{\lambda_2} \| f_k(\lambda) \| \, d\lambda \tag{2.7.6}$$

which is in agreement with (2.7.2), the arc length along θ_k on the fitting surface.

2.7.3 *Example*

The example, for which the formulas and data and calculations are given in Appendix 2.1, can be used to demonstrate the application of (2.7.6). Fig. 7 shows that, starting at $\theta^{(0)} = 1$, the length of the total tangent Jd in the tangent plane equals 7.28 in units of the metric I for the observation space E^2. The arc length along the only existing parametric curve on the fitting surface is given by

$$\int_{\theta^{(0)}}^{\theta^{(1)}} \| f_1 \| d\theta = e^{\theta} \Big|_{\theta^{(0)}}^{\theta^{(1)}} = 7.28$$

Since the initial value is $\theta^{(0)} = 1$, it is readily found that $\theta^{(1)} = 2.30$ which is the solution θ^*. Application of differential corrections produces $\theta^{(1)} = \theta^{(0)} + d = 3.68$ and overshooting on the fitting surface takes place (Fig. 7).

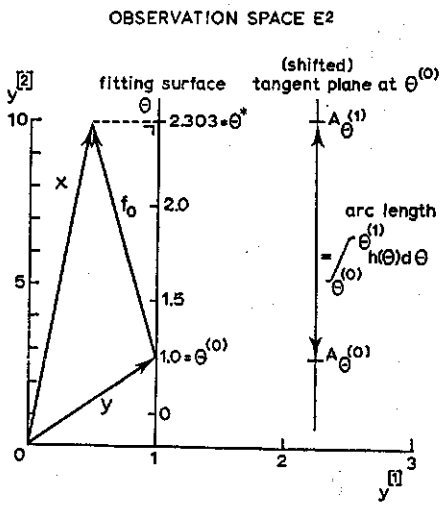


Fig. 7. Use of arc length in the observation space E^2 to determine the optimal step length on the fitting surface which in this example is degenerated into a straight line. The scale factor h is the length of the direction vector f_1 which in this particular case coincides with the fitting surface and emanates from $\theta = 1.0$ with length 2.72 (after Stol, 1962a).

2.8 Curvature

Formulas for the curvature of curves are derived in differential geometry. The principles of the theory will be discussed in short. Reference is made to Struik (1961). The derivation of formulas is for the kth parameter, leaving it understood that for the other parameters the results are identical.

If $f = f(\theta_k)$ represents the kth parametric curve, then an arc element ds is given by

$$ds = [(df^{[1]})^2 + \ldots + (df^{[v]})^2]^{\frac{1}{2}} = \|df\|$$

From this it follows that the derivative with respect to θ_k reads

$$\frac{ds}{d\theta_k} = \frac{\|df\|}{d\theta_k} = \left(\frac{df}{d\theta_k}, \frac{df}{d\theta_k}\right)^{\frac{1}{2}} = \|f_k\| = h_k \tag{2.8.1}$$

This result gives the possibility to use the more convenient arc length s as a parameter along the kth parametric curve. Consequently

$$\frac{df(s)}{ds} =: f_s = \frac{df}{d\theta_k}\left(\frac{ds}{d\theta_k}\right)^{-1} = \frac{f_k}{h_k} \tag{2.8.2}$$

Equation (2.8.2) represents a tangent vector to the kth parametric curve. The length of this vector does not depend on s because it is equal to 1. So we have

$$\frac{d}{ds} f_s^T f_s = 2 f_{ss}^T f_s = 0$$

hence $f_{ss} \perp f_s$. The vector f_{ss} gives the rate of change of the tangent when we proceed along the curve. For this reason f_{ss} is called the curvature vector whose length is the

curvature. The reciprocal of the curvature is called the radius of curvature. The curvature is denoted by

$$\kappa := \|f_{ss}\|$$

and can be determined as follows. By virtue of (2.8.2) we have

$$\frac{d}{ds} f_s = \frac{d}{ds} \frac{f_k}{h_k} = \left(\frac{d}{d\theta_k} \frac{f_k}{h_k}\right) \left(\frac{ds}{d\theta_k}\right)^{-1}$$

hence, using (2.7.1) and (2.8.1)

$$f_{ss} = \frac{f_{kk} h_k - f_k (f_k, f_{kk}) (h_k)^{-1}}{(h_k)^3} \tag{2.8.3}$$

The length of this vector can be written with a determinant, using $h_k^2 = (f_k, f_k)$

$$\kappa = \frac{1}{h_k^3} \begin{vmatrix} (f_k, f_k) & (f_k, f_{kk}) \\ (f_k, f_{kk}) & (f_{kk}, f_{kk}) \end{vmatrix}^{\frac{1}{2}} \tag{2.8.4}$$

which is the curvature at θ of the kth parametric curve. A zero second derivative produces zero curvature. The elements of the determinant can be found on the main diagonal of the matrices M, M_{12} and M_{22} (Table 1).

The curvature κ is a characteristic for space curves. For curves on surfaces, as parametric curves used in the present sence, another characteristic is of importance. This is the component of the curvature vector f_{ss} in the direction of the tangent plane, the geodesic curvature. This component is the ordinary curvature vector of the projection of the curve on the tangent plane to the surface at θ (cf. Struik, 1961).

The geodesic curvature vector is obtained as follows. The orthogonal projection of a vector on the tangent plane can be expressed by a linear combination of the basis vectors which are in the Jacobian J. According to (2.4.9) and (2.4.13) the components of this linear combination are given by $M^{-1} J^T f_{ss}$ and so the geodesic curvature vector itself reads

$$(f_{ss})_g = J M^{-1} J^T f_{ss}$$

and the magnitude of the curvature is obtained from

$$\kappa_g^2 := \|(f_{ss})_g\|^2 = f_{ss}^T J M^{-1} J^T f_{ss} \tag{2.8.5}$$

which can be considered the square of the length of f_{ss} under the metric $JM^{-1}J^T$.

A further component of the curvature vector is now obtained by

$$f_{ss} = (f_{ss})_g + (f_{ss})_r$$

The second vector at the right-hand side is a remainder curvature in a $(v - p)$-dimensional subspace of the observation space which is orthogonal to the fitting surface in θ.

The corresponding curvatures are connected by

$$\kappa = (\kappa_g^2 + \kappa_r^2)^{\frac{1}{2}}$$

The geodesic curvature in units of $\sqrt{S}$ will be called the relative curvature and it is given by $\kappa_g \sqrt{S}$.

Thus far attention was paid to curvature in the direction of a parametric curve. In the general case of curvature in an arbitrary direction of search s the formulas are to be generalized. Let an arbitrary curve on the fitting surface be given by $y(\lambda) = f(\theta(\lambda))$ where $\theta(\lambda) = \theta^{(0)} + \lambda s$. Then, analogous to (2.8.1) we have, making use of the chain rule in (2.5.4)

$$\frac{ds}{d\lambda} = \left\| \frac{df}{d\lambda} \right\| = \left\| J_\theta^f \frac{d\theta}{d\lambda} \right\| = \|Js\|$$

and analogous to (2.8.2)

$$f_s = Js \|Js\|^{-1}$$

This is a vector of unit length in the direction of the given linear combination of direction vectors.

To determine df_s/ds we make use of (2.5.8) to obtain

$$\frac{d}{d\lambda} J_k^{[i]}(\theta(\lambda)) = (\nabla^T f_k^{[i]}, s) =: K_k^{[i]}$$

hence, by this definition

$$\frac{d}{d\lambda} J(\theta(\lambda))s = Ks$$

and obviously $Ks = (J_\theta^{f^1} s, \ldots, J_\theta^{f^p} s)s$. The derivative of the length of Js reads

$$\frac{d}{d\lambda} [s^T J^T Js]^{\frac{1}{2}} = \frac{s^T K^T Js}{\|Js\|} = \frac{(Js, Ks)}{\|Js\|}$$

giving finally for $(df_s/d\lambda)\,(ds/d\lambda)^{-1}$

$$f_{ss} = \frac{Ks\|Js\|^2 - Js(Js, Ks)}{\|Js\|^4}$$

with length

$$\kappa = \frac{1}{\|Js\|^3} \begin{vmatrix} (Js, Js) & (Js, Ks) \\ (Js, Ks) & (Ks, Ks) \end{vmatrix}^{\frac{1}{2}}$$

Taking $s = (0,\ldots,0,\theta_k,0,\ldots,0)^T$ these formulas reduce to (2.8.3) and (2.8.4) obtained for the kth parametric curve.

2.9 Relationship between spaces

The spaces distinguished merit further comment as regards their relationship. It is not adequate to use only the parameter space to illustrate the progress of search. The tangent space must be used as well.

First we consider the observation space (Fig. 8). Vectors in the tangent plane are linear combinations of the direction vectors of J. The metric in the tangent plane is M because for any $p \times 1$ vector u we have $\|Ju\|^2 = u^T M u$. Vectors u can be represented in the tangent space with a $p \times p$ metric I. This is done in Fig. 9 for $u = d$ and $u = b$, respectively.

The image of the path of search on the fitting surface is a curved line obtained by

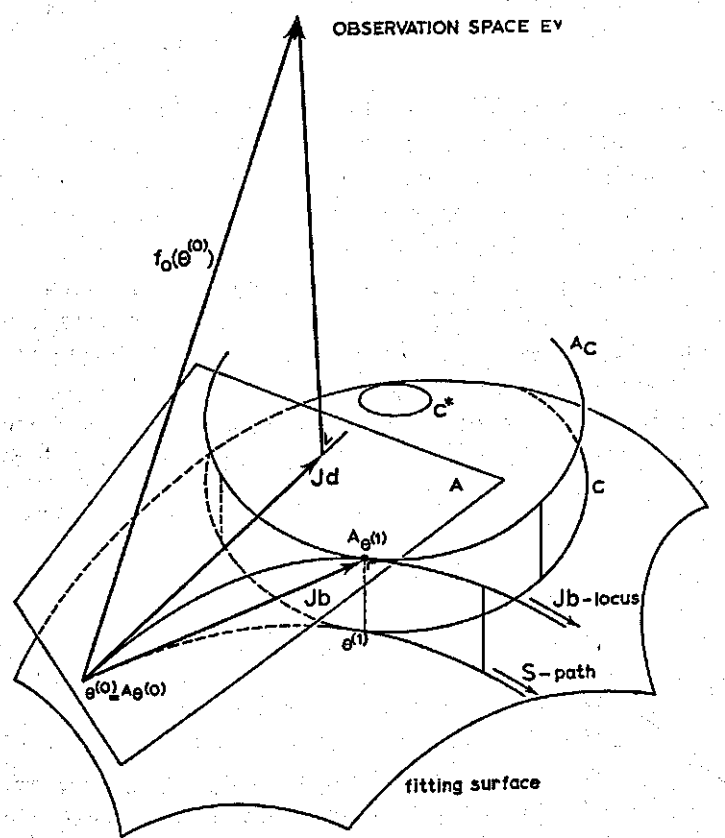


Fig. 8. Observation space with application of differential corrections. On the fitting surface the path of search generated by s emanates from $\theta^{(0)}$ in the direction of Jd. The orthogonal projection of this s-path on the tangent plane A is denoted by Jb-locus. Contour curves for S = constant are denoted by C, the neighbourhood of the solution is indicated by C^*. The metric of the tangent plane is $J^T J = M$.

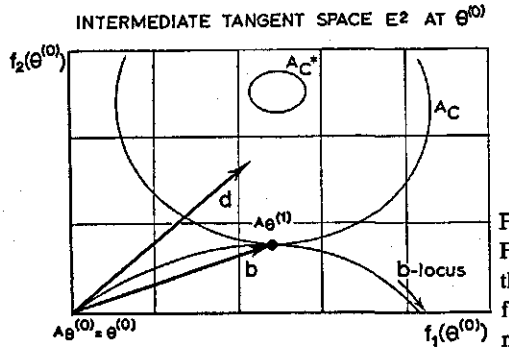


Fig. 9. Intermediate tangent space at $\theta^{(0)}$ in Fig. 8 with differential vector $\boldsymbol{d}$ and image of the path of search, the $\boldsymbol{b}$-locus. Contour curves for S = constant are denoted by ${}^{A}C$. The metric of the tangent space is $\boldsymbol{I}$.

orthogonal projection of this path on the tangent plane. In the Figs. 8 and 9 it is denoted by $\boldsymbol{b}$-locus. Although the application of differential corrections is based on the fact that a direction $\boldsymbol{d}$ to a neighbourhood of the minimum (C^*) can be found, the image of the actual path of search shows that only in ${}^{A}\theta^{(0)}$ it is in the direction of $\boldsymbol{d}$ (Fig. 9), where the left superscript A refers to quantities with respect to the tangent plane.

Next we consider the parameter space (Fig. 10). This space also has the $p \times p$ metric $\boldsymbol{I}$ and consequently it has the same coordinates as the tangent space. However, there are differences that are essential. Contour curves for $S(\theta)$ = constant form a fixed pattern in the parameter space. These contour curves change shape in a sequence of intermediate tangent spaces, they tend toward those in the parameter space when ${}^{A}\theta^{(0)} \rightarrow {}^{A}\theta^{*}$. The path of search on the fitting surface is curved but it is the locus of points $\theta = \theta^{(0)} + \lambda s$, where $s = d$, which plots a straight line in the parameter space.

From the representation in the intermediate tangent space it becomes clear that the curvature of the parametric curves on the fitting surface causes divergent shooting rather than overshooting. For this reason the method of back projection was developed in Chapter 6 to find paths that proceed towards the end point of the image of $\boldsymbol{Jd}$ in the parameter space (Fig. 10).

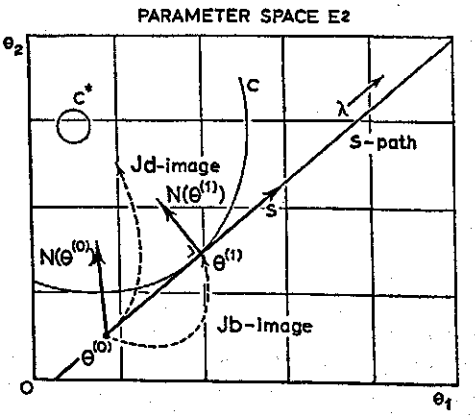


Fig. 10. Parameter space with path of search $\theta(\lambda) = \theta^{(0)} + \lambda s$, where $s = d$ the vector of differential corrections which emanates from $\theta^{(0)}$. Along the s-path the normal N changes direction. Images of the vector $\boldsymbol{Jd}$ and $\boldsymbol{Jb}$ in the observation space are indicated as well. The metric of the parameter space is $\boldsymbol{I}$.

3 Features of optimization

3.1 General

Methods that are available for unconstrained optimization have been reviewed by many authors (Brooks, 1959; Spang III, 1962; Fletcher, 1965; Box, 1966; Box, Davies & Swann, 1969; Fletcher, 1969b; Murray, 1972a; Powell, 1972a, Dixon, 1974). In general the optimization problem is to minimize an objective function $F(x)$ subject to the constraints $c_j(x) \geqq 0, j = 1(1)m$, where a vector $x \in E^v$ that satisfies the constraints is called feasible. It is assumed that $F(x)$ can numerically be evaluated for all feasible x.

Here the objective function $S(\theta)$ is a sum of squared functions. This means that parameter optimization is a special case of mathematical programming techniques. In general S is not quadratic in θ and then its optimization is not a special case of quadratic programming.

3.2 Constraints

In the present case no constraints on the values of the parameters are involved. However, one or more parameters may be bounded to ensure numerical evaluation of the condition function F in Chapter 2. In case of overshooting during intermediate iteration steps, the vector θ can enter the unfeasible region. Instead of keeping the violating parameter constant at its boundary value (Spendly, 1969), subprogram MIN iterates the parameter vector in the feasible region as close to the bound as predefined by program options. When oscillating may occur it should be tested in the next fitting cycle whether the same parameter violates the constraint again before to break down the parameter vector (Chapters 8 and 9).

3.3 Sequential and unsequential search

Unsequential search is employed in situations where function evaluation is expensive or time consuming. Factorial methods or random experimentation can be used, although they are less efficient (Brooks, 1959). Since function values in mathematical models can be obtained by numerical evaluation, most methods for nonlinear optimization of objective functions employ sequential search methods. This means that new parameter values are obtained from previous values by a fixed set of operations to update θ. For instance for any algorithm that produces a direction of search $s = s(\theta)$

1. $\theta(\lambda) = \theta^{(n)} + \lambda s^{(n)}$

2/3. $\lambda^{(n)} = \lambda$ obtained by applied algorithm

4. $\theta^{(n+1)} = \theta(\lambda^{(n)})$ (3.3.1)

5. $\theta^{(n)} := \theta^{(n+1)}$

6. $s^{(n)} := s(\theta^{(n)})$, repeat from step 1

This type of iteration is most common for various choices of s (Cauchy, 1847; Curry, 1944; Crocket & Chernoff, 1955; Hartley, 1961; Spang III, 1962).

The value of $\lambda^{(n)}$ can be held constant, $\lambda^{(1)}$ say, giving in each cycle the iteration

$$\theta^{(n+1)} = \theta^{(n)} + \lambda^{(1)} s^{(n)}, \qquad n = 1(1)t \tag{3.3.2}$$

In this case $\lambda^{(1)}$ is sometimes called the damping factor (cf. Kowalik, 1967). When $\lambda^{(1)} = 1$ in each cycle, the method is a full step method (cf. Hayes, 1974).

3.4 Direct and analytical search

Direct search methods use evaluation of functions only. As a matter of fact they are heuristic methods or methods of empirical optimization, developed in the parameter space. A general introduction to these methods is given by Hooke and Jeeves (1961). The most simple direct search method is that of alternating parameters, each at a time. Other more complicated methods as Partan algorithms (Wilde, 1965), Simplex algorithms (Nelder & Mead, 1965) as also Rosenbrocks method (Rosenbrock, 1960) explore the pattern produced by contours of S = constant in the parameter space and are called methods of pattern search. Reviews of these methods are given by Spang III (1962), Fletcher (1969a), Fletcher (1969b), Powell (1972a), Swann (1972). Direct search methods are used in cases where no derivatives of the condition function are available. This means a loss of efficiency. It will be shown that derivatives can be found for least squares methods even for complicated models (Chapter 4).

Analytical search methods make use of information about first and second derivatives of the objective function with respect to the parameters to be optimized. Thus, these methods – also called gradient methods – use information of the direction toward the minimum.

One can distinguish between methods in which the derivatives are available and those in which they are approximated. In the last case they are obtained by difference techniques, evaluating the condition function for two different values of θ_k to approximate f_k.

The first derivative of the objective function results in the gradient g, giving a linear direction of search toward a minimum with $s = -g$. This method was proposed first by Cauchy (1847) and is called the method of steepest descent. Modifications have been proposed to improve the rate of convergence and to accelerate the numerical

process. Curry (1944) did show that convergence depends on the metric, B, used, when $s = -B^{-1}g$. When the contours of the objective function are spherical in the parameter space, the steepest descent converges in one step. In this case $B = I$. When the contours are elliptical, so with a quadratic objective function, conjugate gradient methods can be applied (Fletcher & Reeves, 1964). These type of methods recently have been reviewed by McCormick & Pearson (1969), Fletcher (1972a) and Dixon (1974).

If Taylor expansion of the objective function about the current approximation is employed, the first three terms of the series are used and the optimization method is a second order method called the Generalized Newton or Newton-Raphson method. When the objective function is quadratic in the parameters one step convergence is obtained by the Newton-Raphson algorithm using the Hessian as metric, so $d = -G^{-1}g$.

Linearization of a nonquadratic objective function results in methods where the metric G is applied and updated in each fitting cycle. This is called a variable metric method. If a minimum is determined in the current direction of search in each cycle, the method is an optimal gradient method. Otherwise the method is a fixed step factor method, that can be used when the decrease of values of the objective function for a number of fitting cycles according to (3.3.2) is pertinent.

Algorithms where G^{-1} is approximated by a matrix H, say, and updating of H occurs each cycle (Davidon, 1959), are called Quasi Newton methods. The matrix H is taken positive definite to ensure reduction of the response S. Several of these methods have been compared and discussed by Fletcher (1969b).

Mixed methods make use of both direct and analytical search as the polyalgorithm for global nonlinear least squares problems designed by Aird (1973). A further example is the simulation of convergence of parameter values as developed in Chapter 10. After several cycles of analytical search, values obtained thus far for each parameter separately are extrapolated in an empirical way to accelerate convergence.

3.5 Dimension of search, optimal search

Nonlinear parameter optimization is employed by reducing the number of parameters in each fitting cycle using a $q \times 1$ algorithm parameter vector λ, $q < p$, in which case the search is q-dimensional. In fitting practice the number of algorithm parameters is small, two at most, when applying techniques for finding an optimal value of λ that gives a subminimum of S. The condition to achieve optimal progress is given by (2.5.14). For one-dimensional search, that still can be either curved or linear, this condition reads

$$g^T \frac{d\theta(\lambda)}{d\lambda} = 0$$

which for linear and strict parameter functions according to (2.1.9), reduces to $(g,s) = 0$.

In one-dimensional linear optimal search it is the minimum in the direction of search that is used to furnish a new starting value for the next fitting cycle to guarantee convergence (e.g. Kowalik & Osborne, 1968). The update of θ for an algorithm that produces a subminimum in the direction of search $s = s(\theta)$ reads

$$
\begin{aligned}
&1.\ \theta(\lambda) = \theta^{(n)} + \lambda s^{(n)} \\
&2.\ S(\theta(\lambda^{*(n)})) = \min_{\lambda} S(\theta(\lambda)) \\
&3.\ \lambda^{(n)} = \lambda^{*(n)} \\
&4.\ \theta^{(n+1)} = \theta(\lambda^{(n)}) \\
&5.\ \theta^{(n)} := \theta^{(n+1)} \\
&6.\ s^{(n)} := s(\theta^{(n)}), \qquad \text{repeat from step 1}
\end{aligned}
\tag{3.5.1}
$$

Since the minimization in step 2 is nonlinear, λ is to be solved by iterative methods (see Chapter 9).

One-dimensional nonlinear search can be carried out in several ways. Application of weights to the components of the search vector s as developed in Chapter 5 gives in step 1 in (3.5.1)

$$\theta(\lambda) = \theta^{(n)} + \lambda w^{D}(\lambda) s^{(n)} \tag{3.5.2}$$

where the weights in the diagonal matrix w^{D} depend on the step factor λ.

One-dimensional nonlinear search can also be obtained by treating two directions of search with interpolation techniques. The Levenberg method (Levenberg, 1944), generalized by the Marquardt method (Marquardt, 1963) is an example of such a procedure. It can be shown that the search with these methods is curved (spiral) and is determined by the gradient and the differential correction vector (Jones, 1970).

One-dimensional circular search is a special case which depends on two vectors say u and v with properties $\|u\| = \|v\|$ and $u \perp v$, then step 1 in (3.5.1) would read

$$\theta(\alpha) = \theta^{(n)} + u^{(n)} \cos\alpha + v^{(n)} \sin\alpha \tag{3.5.3}$$

where now the angle α is the algorithm parameter to be optimized.

Extension to a two-dimensional circular search is obvious. Optimizing the direction of search and the step length in the new direction, the update of θ is obtained with

$$\theta(\alpha,\beta) = \theta^{(n)} + \beta(u^{(n)} \cos\alpha + v^{(n)} \sin\alpha) \tag{3.5.4}$$

where the algorithm parameter vector $\lambda = (\alpha,\beta)^{T}$ is subject to optimization.

The back projection method developed in Chapter 6 is an example of the practical use of (3.5.3) and (3.5.4).

3.6 Convergence of gradient methods

The main problem in optimization is the choice of an efficient new direction of search and the application of a metric that guarantees fast convergence.

Proofs of convergence for gradient methods are given in literature for various modifications and for specific functions. Curry (1944) gives a proof for the sequential optimal steepest descent method. If there is only one stationary point in a region C the gradient method starting in C will converge to it. The process will anyway terminate in a stationary point that is in C. Crockett & Chernoff (1955) give a proof for

$$s = -B^{-1}g \tag{3.6.1}$$

using eigenvalues of the matrix $B^{-1}G$, where G is the Hessian defined in (2.5.11). Hartley (1961) proves convergence for the sequential optimal differential correction method.

The essentials of these proofs are that $S(\theta^*) > 0$ and bounded. If in a convex region in the parameter space with boundary θ_c a value $\theta^{(1)}$ can be found such that $S(\theta^{(1)}) < S(\theta_c)$ for any boundary value θ_c, and if a direction of search s can be found that gives $S(\theta^{(2)}) = S(\theta^{(1)} + \lambda^{(1)}s^{(1)}) < S(\theta^{(1)})$ the process will converge to a stationary point. Obviously a sufficient requirement to λ that the mentioned inequality be satisfied (Spang III, 1962) reads $\left.\frac{\partial S}{\partial \lambda}\right|_{\lambda = 0} < 0$. Such solutions exist for gradient methods as the method of steepest descent, the Gauss-Newton and the Levenberg method (Tornheim, 1963). Weight and metric of several gradient methods are listed in Table 2.

While for gradient methods $S(\theta^{(n+1)}) < S(\theta^{(n)})$ is obtained by optimizing the step length, the direct search methods achieve this by inspection.

Table 2. Conspectus* of gradient search directions based on the general formula $s = -W(A + aI)^{-1}g$.

Method	Weight W	Metric		Search
		matrix A	scalar a	
1. steepest descent	I	I	0	linear
2. general gradient	I	B	0	linear
3. differential correction	I	M	0	linear
4. scale factor weight	w^D	M	0	curved
5. damped differential correction	I	M, C	α	curved
6. second order, approximation	I	H^{-1}	0	linear
7. second order	I	G	0	linear

* The matrix C is defined in Chapter 2, matrix w^D will be defined in Chapter 5, B denotes an arbitrary metric matrix. For least squares $M = J^TJ$ and $G = 2(M - N_{02})$. The matrix H stands for an approximation to G^{-1} according to Davidon's algorithm. Values for α and H are chosen to ensure the metric being positive definite (Davidon, 1959; Marquardt, 1963).

3.7 Local and global minima

Optimization algorithms are designed to find the solution $g^* = 0$, they thus find at least local minima. A necessary condition for a local minimum (cf. Fiacco & McCormick, 1968) is that $g^* = 0$ and G^* a positive semi-definite matrix. A sufficient condition is that $g^* = 0$ and G^* a positive definite matrix. For a global minimum it is required that for all $\theta \in E^p$ we have a θ^* that satisfies

$$S(\theta^*) \leqq S(\theta)$$

There is only little known about algorithms that furnish information on the type of minimum obtained for arbitrary functions (cf. Spang III, 1962; Murray, 1972b). Hartley (1961) suggests a grid search of the parameter space. One can also use different starting approximations and observe whether they converge on the same stationary point. The entire feasible region should be investigated in this way. Such a type of numerical analysis is a laborious method, however, even for a small number of parameters (Powell, 1972a). For least squares problems, paths of search can be found that avoid heavy oscillation of intermediate results. For the same starting point and the same algorithm different paths on the fitting surface can then be followed. A method to achieve this kind of exploration will be paid attention to in Chapter 11.

3.8 Least squares

Objective functions with the form of a sum of squares have special properties. While the general problem of optimization is in the parameter space, minimization of a sum of squares S can be studied in the observation space, where the problem is the determination of the foot of a perpendicular to the fitting surface. For this case Taylor expansion produces the normal equations (Sections 2.3 and 2.4).

When the condition function is linear in the parameters, the objective function is quadratic giving $G = 2J^TJ$ for the Hessian. In nonlinear cases the Hessian reads $G = 2J^TJ - 2N_{02}$ as derived in (2.5.12). The Gauss-Newton method for least squares uses a metric where the matrix N_{02} is considered to be absorbed in the remainder $o(\lambda^3)$ of the Taylor expansion in (2.6.2) (Powell, 1972a). This means that only the first derivatives of the condition function need to be evaluated.

Since in most practical fitting problems $M = J^TJ$ is positive definite, its use guarantees a descent in the response space emanating from the starting approximation. To obtain convergence it is necessary to determine the next point in the parameter space according to the conditions discussed in Section 3.6. This means that determining a minimum in the direction of search producing a lower value for the sum of squares is efficient. This optimal step factor method is called the modified Gauss-Newton method which is given in (3.5.1) with $s = d$.

Methods have been developed that improve the condition of M by adding a number to the diagonal terms of this matrix (Levenberg, 1944 and Marquardt, 1963), but Davies & Whitting (1972) report the rate of convergence as being slow compared to the

Gauss-Newton algorithm because of the damping of the normal equations.

A review of methods, which are mostly based on the efficient treatment of the Jacobian and on interpolation techniques between the gradient and the differential corrections, can be found in Jacoby et al., (1972).

3.9 Evaluation of methods

There is no general classification of optimization methods with respect to their efficiency for arbitrary condition functions. The applicability of an algorithm depends on the special properties of the functions to be minimized and the extent to which assumptions made, hold. The choice of an algorithm therefore depends on several considerations. Algorithms that ensure convergence may work so slow that application without any modification is a waste of computer time. Therefore some algorithms are designed in such a manner that updating of matrices is not carried out in each cycle but only after a predetermined number of them.

The slow convergence of the method of steepest descent has been emphasized by many authors (Spang III, 1962; Marquardt, 1963; Fletcher, 1969b), although the first few steps may give an appreciable decrease of the response the rate of convergence is in general considered unpredictable. Steepest descent convergence will occur even from a poor initial approximation in the first cycle (Powell, 1972a), but the direction toward the minimum according to the Gauss-Newton algorithm is often found to be about perpendicular to the direction of steepest descent for the first fitting cycles (Marquardt, 1963; Davidon, 1969; Powell, 1972b).

The Gauss-Newton method is independent of scaling and although there is no device to force convergence from a poor approximation (Powell, 1972a) it converges rapidly when near a solution.

Fletcher (1972b) points out that in his experience first derivatives are usually extremely valuable and that second derivatives do not furnish the same order of improvement. In least squares methods updating of the matrix $2\boldsymbol{J}^T\boldsymbol{J}$ will usually be quite satisfactory. It will be shown in Chapter 5 that second derivatives are useful when applying scale factor differentials as weighting system for the differential correction vector.

Methods which use a nonlinear search might have advantages over others. They may follow the natural valleys better, as the Levenberg method does (Curry, 1944). However, a practical drawback can be the numerical complexity of the method. In the Levenberg method the algorithm parameter occurs implicit in the metric (Table 2) which to find the parameter value has to be inverted for each step in the iterative process. Davies & Whitting (1972) derive a single prediction formula for this parameter to simplify arithmetics.

Algorithms that use function values only, have the advantage that no further formulas need be evaluated. The most simple of these algorithms, the alternating direction or univariate method, usually fails to give convergence in a reasonable number of fitting cycles (Fletcher, 1969b), especially in situations where contours of the response

Table 3. Classification of optimization methods in descending efficiency.

Sequential methods	Nonsequential methods
Second order methods	Factorial methods
First order methods	Random methods
Gauss-Newton	
Quasi Newton	
Conjugate gradient	
Zero order methods	
Conjugate directions	
Direct search	

surface in the parameter space are inclined at about equal angles to the coordinate axes (Spang III, 1962). When using weak stopping criteria the obtained parameter values will deviate unpredictably from the required solution.

Some authors, possibly in accordance with their experience, give a classification of methods in nonlinear optimization as regards their assumed efficiency. A combination of classifications given by Brooks (1959), Hartley (1961) and Murray (1972d) could read as given in Table 3.

Classification of various modifications of the above mentioned methods according to 'standard' problems in unconstrained optimization are given by Kowalik & Osborne (1968), Sargent & Sebastian (1972), Himmelblau (1972), Davies & Whitting (1972), Aird (1973) and Dixon (1974) among others.

3.10 Stopping criteria

Stopping criteria are used to terminate the execution of an optimization problem for several practical reasons.

To avoid waste of computer time tests are used in computer programs to stop the process after a predetermined number of cycles has been performed, so even if no convergence takes place the number of fitting cycles remains limited. For the same reasons the use of control cards in the input stream to define the maximum computer time in a single run, can be advocated. When convergence is apparent one can terminate the numerical process when a required accuracy is obtained.

Main criteria mostly used for the termination of the optimization process are

- magnitude of the relative change of the response in consecutive cycles,
- magnitude of the relative change of the parameter values in consecutive cycles,
- length of the gradient.

Stopping criteria for least squares methods will be discussed in more detail in Section 7.4.2.

4 Extension of the condition function

4.1 General

Functions that occur in fitting practice have to be put in a suitable form before the theory discussed in Chapter 2 can be applied. In cases where the derivation and programming of second derivatives gives complications it here is assumed that algorithms based on first derivatives still can be applied. It is then at least necessary to calculate the components of the vectors f_0 and f_k, $k = 1(1)p$, numerically. These vectors are essential when using gradient or differential correction methods. However, the determination of both vectors is not always achieved by simple numerical evaluation; sometimes an iterative solution of implicit functions and the solution of derivatives from simultaneous equations is required.

To distinguish between functions of the same variables and parameters but of different form a left-superscript is introduced, e.g. ${}^1F(x,\theta)$ and ${}^2F(x,\theta)$. Particular functions will be defined in each section separately. Letter subscripts denote differentiation with respect to the indicated parameter according to (2.1.11) and (2.1.12).

4.2 Functions of different form

A vector of values defined by (2.1.5) reads

$$F(y_j,\theta) = \begin{bmatrix} F(x_1^{[1]},\ldots,x_{j-1}^{[1]},y_j^{[1]},x_{j+1}^{[1]},\ldots,x_m^{[1]},\theta) \\ \cdot \\ \cdot \\ \cdot \\ F(x_1^{[\nu]},\ldots,x_{j-1}^{[\nu]},y_j^{[\nu]},x_{j+1}^{[\nu]},\ldots,x_m^{[\nu]},\theta) \end{bmatrix} = 0 \qquad (4.2.1)$$

It is not necessary that all functions F in (4.2.1) have the same form. Generally, equations (4.2.1) are called the condition equations, where the function F represents the condition function, hence the observations are subject to ν conditions which may be of different form (Deming, 1948). In this case (4.2.1) must be written

$$F(y_j,\theta) = \begin{bmatrix} {}^1F(x_1^{[1]},\ldots,x_{j-1}^{[1]},y_j^{[1]},x_{j+1}^{[1]},\ldots,x_m^{[1]},\theta) \\ \cdot \\ \cdot \\ \cdot \\ {}^\nu F(x_1^{[\nu]},\ldots,x_{j-1}^{[\nu]},y_j^{[\nu]},x_{j+1}^{[\nu]},\ldots,x_m^{[\nu]},\theta) \end{bmatrix} = 0 \qquad (4.2.2)$$

to distinguish between v different condition functions. According to (2.1.8) this can be written

$$F(y_j,\theta) = [{}^1F^{[1]}(y_j^{[1]},\theta),\ldots,{}^vF^{[v]}(y_j^{[v]},\theta)]^T = 0$$

which expresses that actually it is the adjusted value $y_j^{[i]}$, $i = 1(1)v$, that is subject to v conditions. The vector of deviations from observed values $x_j^{[i]}$, $i = 1(1)v$, is obtained by

$$f_{0j} = x_j - y_j$$

according to (2.1.10).

If the equations (4.2.2) can be solved, giving $y_j = f_j(\theta)$, the Jacobian reads

$$J_\theta^{f_j} = [\nabla^T({}^1f_j^{[1]}),\ldots,\nabla^T({}^vf_j^{[v]})]^T$$

With these expressions the theory developed in Chapter 2 can be applied. The subscript j will be omitted in further sections unless newly defined in the text.

4.3 Implicit functions

Derivatives required to evaluate the Jacobian can also be obtained from implicit functions $F(y,\theta) = 0$. See (2.1.8).

The ith component differentiated with respect to the kth parameter gives

$$\frac{\partial f^{[i]}(\theta)}{\partial \theta_k} = -\frac{\partial F^{[i]}}{\partial \theta_k}\left(\frac{\partial F^{[i]}}{\partial y}\right)^{-1} =: -\frac{F_k^{[i]}}{F_y^{[i]}}, \qquad F_y^{[i]} \neq 0 \tag{4.3.1}$$

Evaluated for all observations and all parameters this can be written

$$J_\theta^f = -\begin{bmatrix} F_y^{[1]} & & & 0 \\ & \ddots & & \\ & & F_y^{[i]} & \\ & & & \ddots \\ 0 & & & F_y^{[v]} \end{bmatrix}^{-1} \begin{bmatrix} F_1^{[1]},\ldots,F_k^{[1]},\ldots,F_p^{[1]} \\ \vdots \\ F_1^{[i]},\ldots,F_k^{[i]},\ldots,F_p^{[i]} \\ \vdots \\ F_1^{[v]},\ldots,F_k^{[v]},\ldots,F_p^{[v]} \end{bmatrix} \tag{4.3.2}$$

The relationship between the Jacobian matrices for implicit and explicit functions can be expressed by means of Definition 2.2.1 as

$$J_\theta^f = -F_y^d J_\theta^F \tag{4.3.3}$$

4.4 Nested implicit functions

In this section we are concerned with condition functions of the following type

$$F(u,w,\theta) = 0 \tag{4.4.1}$$

where $u = u(\theta)$ and $w = w(\theta)$. Let calculated values that meet the condition (4.4.1) be u and let the vector of observed function values be x_1, then we have to determine the components of the difference vector

$$f_0 = x_1 - u \tag{4.4.2}$$

and of the direction vectors

$$f_k = \frac{\partial u}{\partial \theta_k}, \qquad k = 1(1)p$$

We assume that u can be solved from (4.4.1) iteratively which gives (4.4.2). To solve f_k use is made of the total differential of (4.4.1) with respect to each parameter which reads for the kth,

$$dF(u,w,\theta) = \frac{\partial F}{\partial u}\frac{\partial u}{\partial \theta_k}\,d\theta_k + \frac{\partial F}{\partial w}\frac{\partial w}{\partial \theta_k}\,d\theta_k + \frac{\partial F}{\partial \theta_k}\,d\theta_k = 0$$

from which we obtain with a 1×2 and a 2×1 dimensional vector of variables in the numerator, the solution

$$u_k = -\frac{(1, F_w)\,(F_k, w_k)^T}{F_u}, \qquad k = 1(1)p, \qquad F_u \neq 0 \tag{4.4.3}$$

which has to be evaluated for all i. This equation is a generalization of (4.3.1). An arbitrary element of the Jacobian thus reads

$$J_k^{[i]} = -(F_u^{[i]})^{-1}(1, F_w^{[i]})(F_k^{[i]}, w_k^{[i]})^T$$

which is a generalization of (4.3.2).

4.5 Simultaneous nested implicit functions

A further extension of the use of implicit functions is given by the following simultaneous system of condition functions

$$F(u,v,w,\theta) = 0 \tag{4.5.1}$$

$$G(u,v,z,\theta) = 0$$

where u,v,w, and z are functions of θ and the solution is with respect to u and v. The total differentials $dF = 0$ and $dG = 0$ applied to the kth parameter yield after dividing

them by $d\theta_k$

$$\frac{\partial F}{\partial u}\frac{\partial u}{\partial \theta_k} + \frac{\partial F}{\partial v}\frac{\partial v}{\partial \theta_k} + \frac{\partial F}{\partial w}\frac{\partial w}{\partial \theta_k} + \frac{\partial F}{\partial \theta_k} = 0$$

$$\frac{\partial G}{\partial u}\frac{\partial u}{\partial \theta_k} + \frac{\partial G}{\partial v}\frac{\partial v}{\partial \theta_k} + \frac{\partial F}{\partial z}\frac{\partial z}{\partial \theta_k} + \frac{\partial G}{\partial \theta_k} = 0$$

from which the partial derivatives u_k and v_k have to be solved. The solution is obtained from

$$\begin{bmatrix} F_u & F_v \\ G_u & G_v \end{bmatrix} \begin{bmatrix} u_k \\ v_k \end{bmatrix} = \begin{bmatrix} -F_w w_k - F_k \\ -G_z z_k - G_k \end{bmatrix} \tag{4.5.2}$$

and can be written with a 2×2 dimensional matrix and a 2×1 dimensional vector of variables at the right-hand side

$$\begin{bmatrix} u_k \\ \\ v_k \end{bmatrix} = \frac{-1}{F_u G_v - G_u F_v} \begin{bmatrix} G_v & -F_v \\ \\ -G_u & F_u \end{bmatrix} \begin{bmatrix} F_k + F_w w_k \\ \\ G_k + G_z z_k \end{bmatrix} \tag{4.5.3}$$

which is a generalization of (4.4.3).

Next we assume that (4.5.1) can be written in a special form to obtain mutual solvable equations, viz.

$$\begin{aligned} F &\equiv u - f(v,w,\theta) = 0 \\ G &\equiv v - g(u,z,\theta) = 0 \end{aligned} \tag{4.5.4}$$

then (4.5.3) takes the form

$$\begin{bmatrix} u_k \\ \\ v_k \end{bmatrix} = \frac{1}{1 - g_u f_v} \begin{bmatrix} 1 & f_v \\ \\ g_u & 1 \end{bmatrix} \begin{bmatrix} f_k + f_w w_k \\ \\ g_k + g_z z_k \end{bmatrix}$$

4.6 Sequential functions

Sequential functions can be considered a series of multiple nested implicit functions of different form with respect to the argument. The vector of calculated function values of such a sequence can be written, given a starting value $y^{[0]}$, in the following way

$$y = \begin{bmatrix} y^{[1]} \\ y^{[2]} \\ \cdot \\ \cdot \\ \cdot \\ y^{[i]} \\ \cdot \\ \cdot \\ \cdot \\ y^{[v]} \end{bmatrix} = \begin{bmatrix} f^{[1]}(y^{[0]}, \theta) \\ f^{[2]}(y^{[1]}, \theta) \\ \cdot \\ \cdot \\ \cdot \\ f^{[i]}(y^{[i-1]}, \theta) \\ \cdot \\ \cdot \\ \cdot \\ f^{[v]}(y^{[v-1]}, \theta) \end{bmatrix} = \begin{bmatrix} {}^1f^{[1]}(\theta) \\ {}^2f^{[2]}(\theta) \\ \cdot \\ \cdot \\ \cdot \\ {}^if^{[i]}(\theta) \\ \cdot \\ \cdot \\ \cdot \\ {}^vf^{[v]}(\theta) \end{bmatrix} \tag{4.6.1}$$

We have to determine f_0 and f_k, $k = 1(1)p$. Values of $y^{[i]}$ can be obtained by simple evaluation, the starting point for instance being $y^{[0]} = x_1^{[0]}$, which enables us to determine $f_0^{[i]} = x_1^{[i]} - y^{[i]}$, $i = 1(1)v$. The first, second,..., ith component of an arbitrary vector f_k read

$$\frac{\partial y^{[1]}}{\partial \theta_k} = \frac{\partial f^{[1]}}{\partial \theta_k}$$

$$\frac{\partial y^{[2]}}{\partial \theta_k} = \frac{\partial f^{[2]}}{\partial y}\frac{\partial y^{[1]}}{\partial \theta_k} + \frac{\partial f^{[2]}}{\partial \theta_k}$$

$$\cdot$$
$$\cdot$$
$$\cdot$$

$$\frac{\partial y^{[i]}}{\partial \theta_k} = \frac{\partial f^{[i]}}{\partial y}\frac{\partial y^{[i-1]}}{\partial \theta_k} + \frac{\partial f^{[i]}}{\partial \theta_k}, \qquad i = 3(1)v \tag{4.6.2}$$

Sequential functions given by (4.6.1) can easily be reduced to functions defined in Section 2.1. We only need to replace the variable $y^{[i]}$ in the argument of f by observed values, say $x_2^{[i]}$, so producing in the derivatives in (4.6.2) $\dfrac{\partial f^{[i]}}{\partial y} = 0$, $i = 1(1)v$ and so $\dfrac{\partial y^{[i]}}{\partial \theta_k} = f_k^{[i]}$. The observation vector x_2 is in this case defined by

$$x_2 := (x_1^{[0]}, x_1^{[1]}, \ldots, x_1^{[v-1]})^T$$

The starting value for the sequence, viz. $y^{[0]} = x_1^{[0]}$, will generally not be an optimal choice. For this reason we can define $y^{[0]} =: \theta_{p+1}$ where $\theta_{p+1}^{(0)}$ can be taken equal to $x_1^{[0]}$. This new parameter can be used to extend the vector of parameters. To optimize θ_{p+1}, partial derivatives with respect to this parameter have to be determined.

In a computer program both possibilities can be built in, their choice being governed by a system parameter r. Using an auxiliary variable z, and renumbering the parameters such that now θ_p defines $y^{[0]}$, the series (4.6.1) can be set up as follows

$$
\begin{array}{llll}
y^{[1]} & = f^{[1]} & (z^{[1]} & ,\theta) \\
y^{[2]} & = f^{[2]} & (z^{[2]} & ,\theta) \\
\cdot & & & \\
\cdot & & & \\
\cdot & & & \\
y^{[i]} & = f^{[i]} & (z^{[i]} & ,\theta) \\
\cdot & & & \\
\cdot & & & \\
\cdot & & & \\
y^{[v-1]} & = f^{[v-1]} & (z^{[v-1]} & ,\theta) \\
y^{[v]} & = f^{[v]} & (z^{[v]} & ,\theta)
\end{array}
\qquad
\begin{array}{lll}
z^{[1]} & = r\theta_p & + (1-r)x_1^{[0]} \\
z^{[2]} & = ry^{[1]} & + (1-r)x_1^{[1]} \\
z^{[3]} & = ry^{[2]} & + (1-r)x_1^{[2]} \\
\cdot & & \\
\cdot & & \\
\cdot & & \\
z^{[i+1]} & = ry^{[i]} & + (1-r)x_1{}^{[i]} \\
\cdot & & \\
\cdot & & \\
\cdot & & \\
z^{[v]} & = ry^{[v-1]} & + (1-r)x_1^{[v-1]}
\end{array}
\tag{4.6.3}
$$

where $r = 0$ means nonsequential treatment and $r = 1$ means sequential treatment.

The derivatives of the auxiliary function for $i = 1$ are

$$
\frac{\partial z^{[1]}}{\partial \theta_k} = 0, \qquad k = 1(1)(p-1)
\tag{4.6.4}
$$

$$
\frac{\partial z^{[1]}}{\partial \theta_p} = r
$$

The derivatives of the condition function for $i = 1(1)v$ are obtained from

$$
\frac{\partial y^{[i]}}{\partial \theta_k} = \frac{\partial f^{[i]}}{\partial z}\frac{\partial z^{[i]}}{\partial \theta_k} + \frac{\partial f^{[i]}}{\partial \theta_k}, \qquad k = 1(1)p
\tag{4.6.5}
$$

The next component for the auxiliary function can now be prepared

$$
\frac{\partial z^{[i+1]}}{\partial \theta_k} = r\frac{\partial y^{[i]}}{\partial \theta_k}, \qquad k = 1(1)p, \qquad i = 1(1)(v-1)
$$

where the partial derivatives $\partial z/\partial\theta_k$ cancel out if the system parameter $r = 0$. As the functions z always occur in the same way in the functions f, the derivatives can be programmed efficiently on basis of the following scheme, [denoting DO-loops

$$
\begin{array}{ll}
1 & \left[\; i = 1(1)v \right. \\
2 & \left[k = 1(1)p \right. \\
3 & \left. \left. \dfrac{\partial y^{[i]}}{\partial \theta_k} = \dfrac{\partial f^{[i]}}{\partial z} \dfrac{\partial z^{[i]}}{\partial \theta_k} \right] \right. \\
4 & \left. \dfrac{\partial y^{[i]}}{\partial \theta_k} = \dfrac{\partial y^{[i]}}{\partial \theta_k} + \dfrac{\partial f^{[i]}}{\partial \theta_k}, \qquad p \text{ equations} \right. \\
5 & \left[k = 1(1)p \right. \\
6 & \left. \left. \dfrac{\partial z^{[i+1]}}{\partial \theta_k} = r \dfrac{\partial y^{[i]}}{\partial \theta_k} \right] \right.
\end{array}
$$

where the particular partial derivatives f_k for all parameters have to be programmed separately in step 4 only.

In fitting practice it will be helpful to choose $r = 0$ in the first few fitting cycles to obtain an improved starting value $\theta^{(0)}$ before applying sequential functions with $r = 1$.

If the new parameter that represents the starting value $y^{[0]}$ of the series has to be excluded from the fitting procedure, this can be done by means of the parameter index cards in the main program to be discussed in Chapter 8.

In case of time series it is custom to plot $y^{[i]}$, $i = 0(1)v$, against i = time, connecting the points by polygons thus producing a broken line whose first derivative with respect to time is not continuous. Partial derivatives of the sequential condition function with respect to the parameters, however, are continuous since also in this case the conditions laid down in Section 1.3 hold.

4.7 Alternative functions

A further generalization of the condition function occurs when the structure depends on results obtained by function evaluation. In that case it is not known what the structure of the entire function is until all alternatives have been chosen. This type of function belongs to the class of nested implicit functions. An example is given in Fig. 11 where the functions ${}^1f$ and ${}^2f$ are supposed to have different forms.

We define the row vector of variables $x := (x_2, \ldots, x_m)$ where we assume that it is x_1 which in this model has to be compared with calculated values y to find the sum of squares S. Values for $f_0^{[i]} = x_1^{[i]} - y^{[i]}$ can be obtained by numerical evaluation.

The derivatives in which ${}^jf_k$, $j = 1,2$, is involved read respectively

if $u^{[i]} < 0$

$$
\frac{\partial y^{[i]}}{\partial \theta_k} = \frac{\partial ({}^1f^{[i]})}{\partial u} \frac{\partial u^{[i]}}{\partial \theta_k} + \frac{\partial ({}^1f^{[i]})}{\partial v} \frac{\partial v^{[i]}}{\partial \theta_k} + \frac{\partial ({}^1f^{[i]})}{\partial \theta_k}
$$

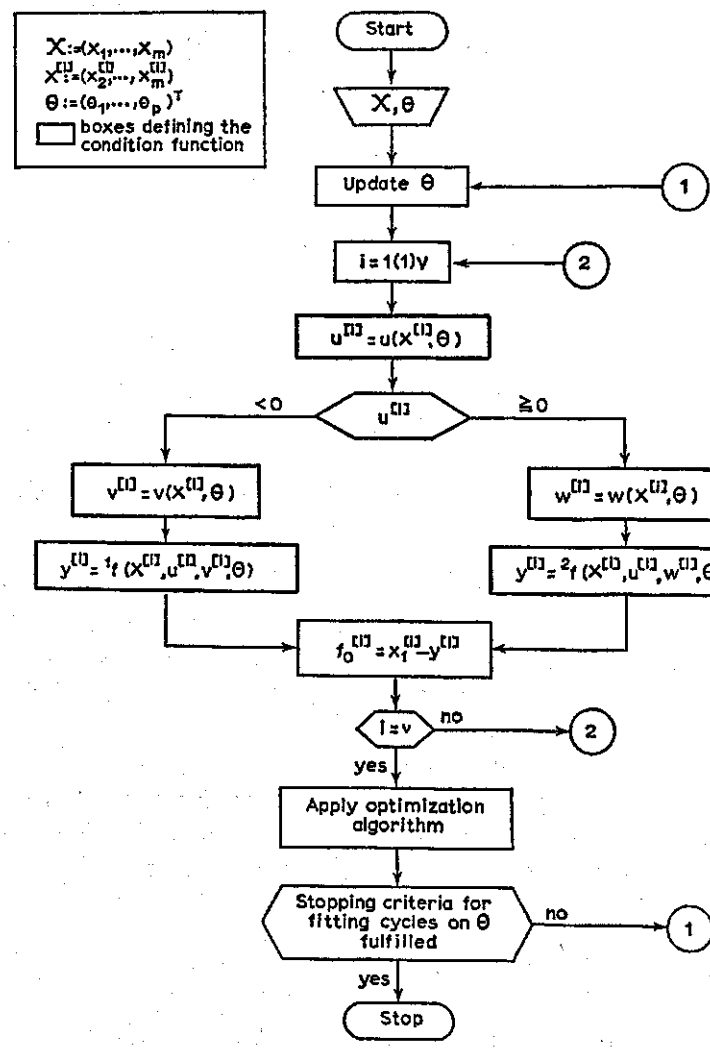


Fig. 11. Flowchart for parameter optimization detailed for the calculation of the difference vector f_0 of a composite alternative condition function. Particularization of ${}^{j}f^{[i]}$; $j = 1, 2$; $i = 1(1)v$, is caused by the value of $u^{[i]}$ which can change for the same i at varying values of θ in consecutive fitting cycles.

if $u^{[i]} \geqq 0$

$$\frac{\partial y^{[i]}}{\partial \theta_k} = \frac{\partial({}^2f^{[i]})}{\partial u}\frac{\partial u^{[i]}}{\partial \theta_k} + \frac{\partial({}^2f^{[i]})}{\partial w}\frac{\partial w^{[i]}}{\partial \theta_k} + \frac{\partial({}^2f^{[i]})}{\partial \theta_k}$$

In the computer program the derivatives can be included in the subprogram for function evaluation. The advantage is that the program can be shortened since use can be made of auxiliary variables in function as well as derivative function statements. When the derivatives are written in their own subprogram, an array of integers must

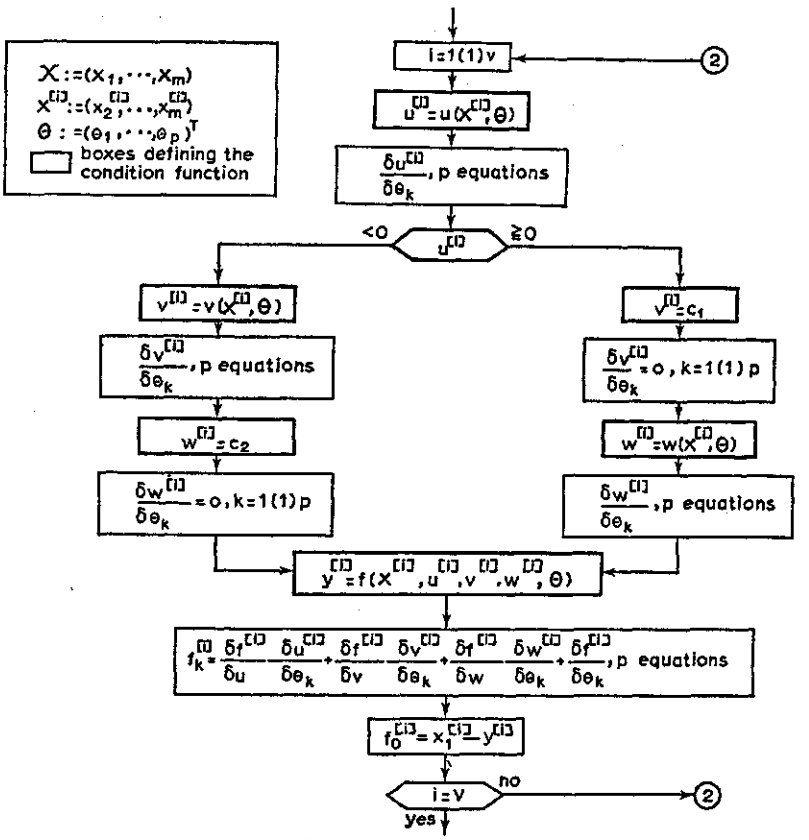


Fig. 12. Part of flowchart for the calculation of the difference vector f_0 and of the first derivatives of a composite alternative condition function. Values of first derivatives are controlled by values of $u^{[i]}$. See Fig. 11 for completion.

be produced that contains information on the step sequence at the $u^{[i]}$-decision for each i.

If f is a function of both, v and w, these functions have to be defined for both choices. An example is given in Fig. 12 where c_1 and c_2 denote constants. The calculation of the derivatives is indicated in the flowchart.

The functions u, v, w and f need not be functions of all components of θ. This means that for u, v and w only those derivatives have to be programmed whose parameters occur in the functions mentioned. The function f, which is a function of the parameters occurring in u, v, w and in its own argument, can be treated as follows. Let the function u depend on $p_u \leqq p$ parameters taken from θ and suppose when treating u that they are ordered by $k_u = 1(1)p_u$, and all this analogous for v, w and f. Numerical calculation of f_k is then carried out by the following scheme

$$
\begin{array}{lll}
i = 1(1)v & & \\
u_{k_u}^{[i]} = \dfrac{\partial u^{[i]}}{\partial \theta_{k_u}}, & p_u \text{ equations} & \\
v_{k_v}^{[i]} = \dfrac{\partial v^{[i]}}{\partial \theta_{k_v}}, & p_v \text{ equations} & \\
w_{k_w}^{[i]} = \dfrac{\partial w^{[i]}}{\partial \theta_{k_w}}, & p_w \text{ equations} & \\
f_u^{[i]} = \dfrac{\partial f^{[i]}}{\partial u}; f_v^{[i]} = \dfrac{\partial f^{[i]}}{\partial v}; f_w^{[i]} = \dfrac{\partial f^{[i]}}{\partial w} & & \\
\quad k = 1(1)p & & \\
\quad f_k^{[i]} = 0 & & \\
f_{k_u}^{[i]} = f_u^{[i]} u_{k_u}^{[i]}, & k_u = 1(1)p_u, & p_u \leqq p \\
f_{k_v}^{[i]} = f_{k_v}^{[i]} + f_v^{[i]} v_{k_v}^{[i]}, & k_v = 1(1)p_v, & p_v \leqq p \\
f_{k_w}^{[i]} = f_{k_w}^{[i]} + f_w^{[i]} w_{k_w}^{[i]}, & k_w = 1(1)p_w, & p_w \leqq p \\
f_{k_f}^{[i]} = f_{k_f}^{[i]} + \dfrac{\partial f^{[i]}}{\partial \theta_{k_f}}, & p_f \text{ equations}, & p_f \leqq p
\end{array}
$$

In Fig. 11 and in Fig. 12 it is assumed that the step sequence depends on the value of $u^{[i]}$. The foregoing procedure can also be used when the function choice is made to depend on the observation order number i. If for a first group of data, say v_1, the right-hand route is to be chosen, the argument in the decision statement on $u^{[i]}$ should read $v_1 - i$.

4.8 Combinations of functions

The functions discussed in Section 4.7 can be sequential functions. A flowchart for the application of such a combination is given in Fig. 13.

Observed values of y and u are components of the $(v + 1) \times 1$ vectors

$$x_3 := (x_1^{[0]}, x_1^{[1]}, \ldots, x_1^{[v-1]}, x_1^{[v]})^T$$

$$x_4 := (x_2^{[0]}, x_2^{[1]}, \ldots, x_2^{[v-1]}, x_2^{[v]})^T$$

It is assumed that values of u are obtained by sequential functions too and that observations on u are components of the vector x_4. We define the row vector of variables $x =: (x_5, \ldots, x_m)$ and use x_1 for the determination of $S = \sum_i (x_1^{[i]} - y^{[i]})^2$. The system can be set up with system parameters r_1 and r_2 as follows

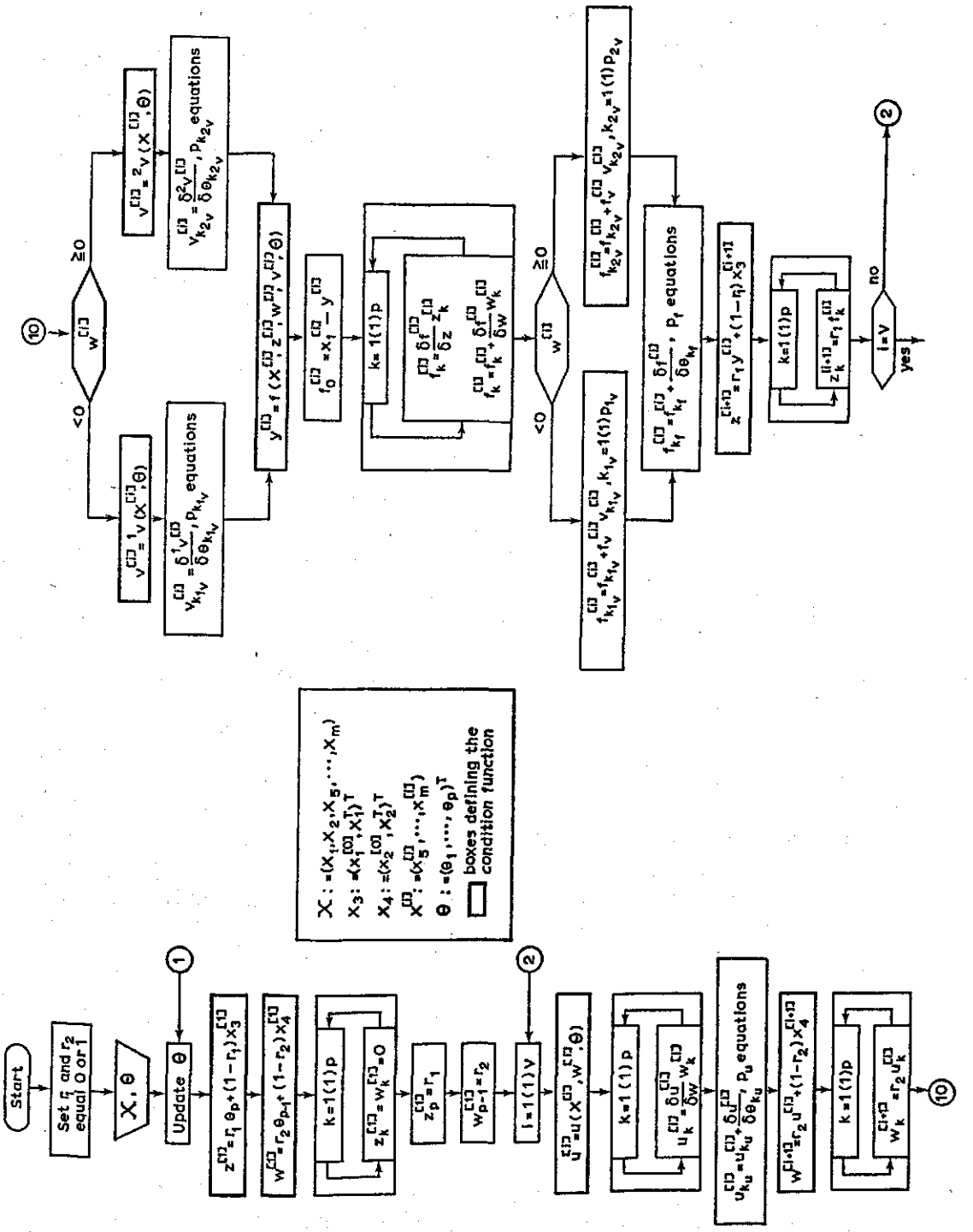


Fig. 13. Part of flowchart for the calculation of the difference vector f_0 and of the first derivatives (indicated by letter subscripts) of a sequential alternative condition function. See Fig. 11 for completion.

$r_1 = 0$, nonsequential treatment of y
$r_1 = 1$, sequential treatment of y
$r_2 = 0$, nonsequential treatment of u
$r_2 = 1$, sequential treatment of u

Auxiliary functions as used in Fig. 13 are

$$z^{[i]} = r_1 y^{[i-1]} + (1 - r_1)x_3^{[i]}, \qquad i = 1(1)(\nu + 1)$$

$$w^{[i]} = r_2 u^{[i-1]} + (1 - r_2)x_4^{[i]}, \qquad i = 1(1)(\nu + 1)$$

where $y^{[0]}$ and $u^{[0]}$ are starting values for function evaluation. The parameter vector is defined such that

$$y^{[0]} =: \theta_p \text{ and } u^{[0]} =: \theta_{p-1}$$

Observed values $x_1^{[0]}$ and $x_2^{[0]}$ are assumed to be available and are to be used as starting values for parameter optimization.

Two functions are distinguished, viz. ${}^1v$ and ${}^2v$, to obtain values for v. Each function has its own subset of parameters. The appropriate derivatives for sequential functions are obtained automatically by the choice of r_1 and r_2 respectively. The starting value $z^{[1]}$ depends on the parameter that represents the starting value $x_1^{[0]}$. However, from $i = 2$ onward, the variable $z^{[i]}$ depends on all parameters because z is a function of y if $r_1 = 1$. Analogous arguments hold for w and u.

Permutation and partitioning of the components of the parameter vector furnishes the possibility of excluding either θ_{p-1} or θ_p, or both, from the fitting procedure (See Chapter 8).

4.9 Special properties of the fitting surface

Application of sequential and alternative condition functions discussed in this chapter result in special properties of the fitting surface. This surface depends on both the form of the condition equations and the observed values assigned to the variables. For $x := (x_3,\ldots,x_m)$ and $X := (x_3,\ldots,x_m)$ let

$$y(\theta) = f(x_2, X, \theta) \tag{4.9.1}$$

be a strict function of the vectors of variables, then y is the position vector to the fitting surface. Fitting without sequential functions takes place on the fixed surface given by (4.9.1).

For sequential functions we define the second observation vector to be used in the $(n + 1)$th fitting cycle as follows (see 4.6.3)

$$x_2^{(n)}(\theta) := [\theta_p^{(n)}, y^{[1]}(\theta),\ldots,y^{[\nu-1]}(\theta)]^T$$

where $y^{[i]}$, $i = 1(1)(\nu-1)$, is found in the nth cycle, then

$$y(\theta) = f(x_2^{(n)}(\theta), X, \theta) \tag{4.9.2}$$

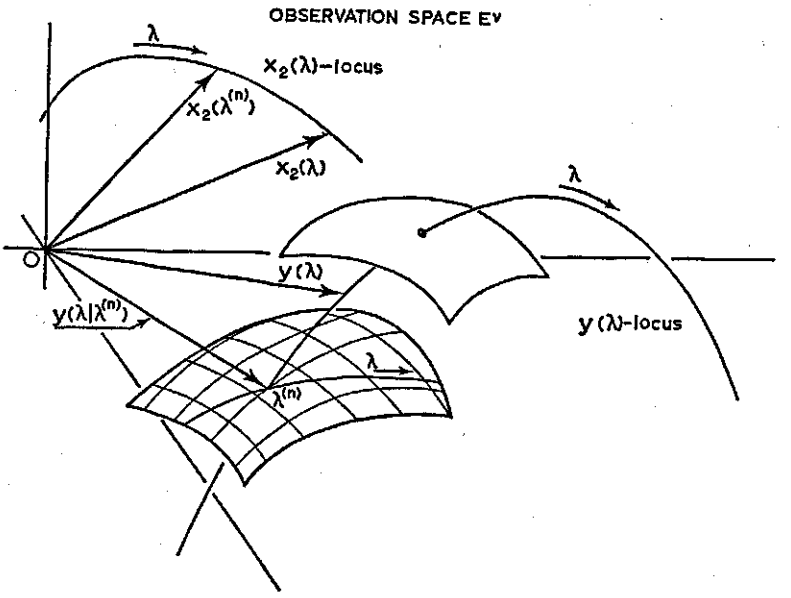


Fig. 14. Two members of a family of fitting surfaces for sequential condition functions. The lower surface is explored by keeping the vector of variables $x_2(\lambda)$ constant at $\lambda = \lambda^{(n)}$.

Now $y(\theta)$ is the position vector to another fitting surface (Fig. 14). Since $x_2^{(n)}$ is a function of the parameters, fitting with sequential functions takes place on a family of surfaces; each iteration within a cycle producing a new member of it. Fitting along a (linear) direction of search in the parameter space can be regarded as fitting along loci which satisfy

$$\begin{aligned} \theta(\lambda) &= \theta^{(n)} + \lambda s^{(n)} \\ x_2^{(n)}(\lambda) &= [\theta_p(\lambda), y^{[1]}(\lambda), \ldots, y^{[\nu-1]}(\lambda)]^T \\ y(\lambda) &= f(x_2^{(n)}(\lambda), X, \theta(\lambda)) \end{aligned} \tag{4.9.3}$$

The nth intermediate fitting surface can be explored by putting $\lambda = \lambda^{(n)} =$ constant in the expression for x_2 in (4.9.3) giving the position vector in Fig. 14,

$$y(\lambda|\lambda^{(n)}) := f(x_2^{(n)}(\lambda^{(n)}), X, \theta(\lambda))$$

The program option to control this is the choice of the value of the system parameter $r = 0$ in the $(n+1)$th fitting cycle. An adequate value for λ is in this case the optimal step factor $\lambda = \lambda^{*(n)}$, see (3.5.1).

For alternative condition functions (see Fig. 11) the fitting surface does not change as long as for all i the same choice is made in the decision statement on $u^{[i]}$. A particular order of the functions ${}^1f$ and ${}^2f$ in the nth fitting cycle can be stored in a $\nu \times 1$ choice vector $j^{(n)}$. Components of this vector are $j^{[i]} = -1$ if $u^{[i]} < 0$, $j^{[i]} = +1$

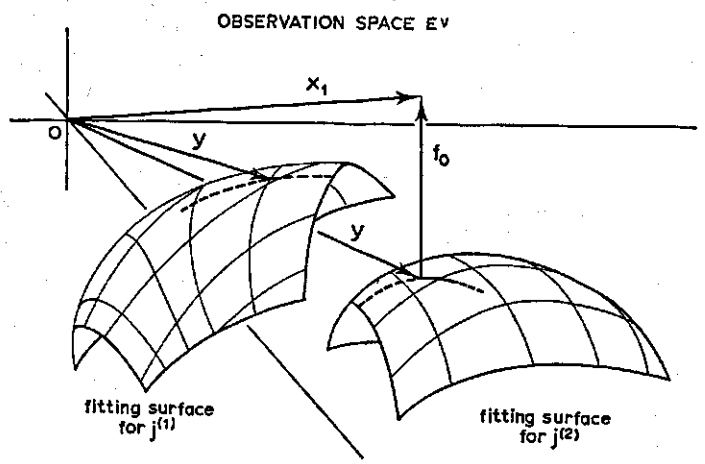


Fig. 15. Schematical illustration of a discrete fitting surface. Along the dotted line the value of one or more of the components of the integer choice vector j suddenly changes, causing a jump to a further fitting surface.

otherwise. Assume a one-dimensional search to be carried out with step factor λ then $j = j(\lambda)$. This vector will not vary continuously with λ. It changes value when jumps to other fitting surfaces occur.

Complications can arise as to the choice of a terminal point $\theta^{(s)}$ since this can be situated on a surface producing a greater value of the response S than that obtained in preceding cycles. Such a situation is sketched in Fig. 15.

5 Use of scale factors for accelerating convergence

5.1 General

Convergence in the fitting procedure can be accelerated by choosing directions in the parameter space that produce paths closer to the minimum of $S(\theta)$.

Stol (1962b) developed a method where use was made of ratios of scale factors to obtain weights to be applied to differential corrections d (Section 2.4) to find these better directions. Determination of weights that depend on the course of scale factors by varying the position vector y, involves evaluation of change of scale factor values when proceeding in the direction of search. Now it depends on the structure of the condition function whether differentials of the scale factors can easily be used to describe the scale factor course with second derivatives. Otherwise, when this structure is complicated, differences have to be applied. However, in that case numerical information on two points on the fitting surface must be available.

The problem of finding weighted corrections to accelerate convergence can be considered part of the problem of finding corrections for curvature of the fitting surface.

5.2 Role of scale factors

Scale factors were introduced in (2.2.7) and discussed in Section 2.7. Their role in parameter optimization will now be dealt with.

Consider two points on the fitting surface, given by $\theta^{(1)}$ and $\theta^{(2)}$, with scale factors $h^{(1)} := h(\theta^{(1)})$ and $h^{(2)} := h(\theta^{(2)})$ respectively.

Assume $\theta^{(2)}$ to be in the direction of differential corrections obtained from (2.4.13) so $\theta^{(2)} = \theta^{(1)} + \lambda d^{(1)}$. It is noted that $d^{(1)}$ is found by projection on the tangent plane to the fitting surface (Fig. 16). This means that the order of magnitude of the components of $d^{(1)}$ is based on units along the parametric curves at $\theta^{(1)}$, being $h_k(\theta^{(1)})$, $k = 1(1)p$, (Stol, 1962a and for a one-dimensional example Draper & Smith, 1967).

In the nonlinear case this results in too rapid a progress in a divergent direction or in slow convergence. Fig. 16 and Fig. 17 show an instance of the first case on a two parameter fitting surface. The differential correction vector in this example is supposed to be $d^{(1)} = (3,3)^T$ expressed in units of h_1 and h_2 at $\theta^{(1)}$ in the tangent plane.

In the figure the scale factors are assumed to increase, each in its own way, along the path from $f(\theta^{(1)})$ to $f(\theta^{(2)})$. This means that the length of this path on the fitting surface is considerably longer than the length of the image of this path in the tangent plane, viz. the length of the total tangent Jd. To overcome the consequences of this

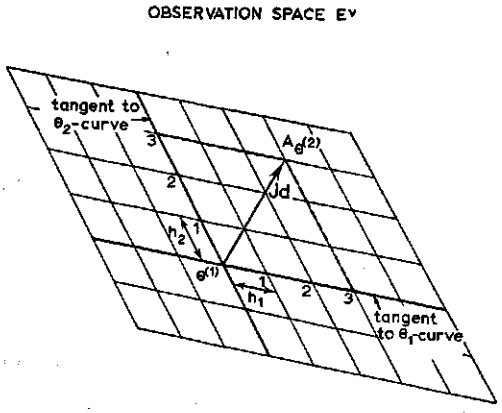


Fig. 16. Detail of tangent plane to fitting surface at $\theta^{(1)}$ with differential correction vector $d = (3, 3)^T$ measured in units of the scale factors h_1 and h_2. The terminal point of Jd is obtained by orthogonal projection as sketched in Fig. 6.

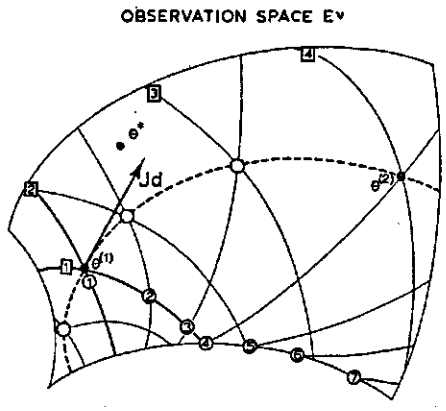


Fig. 17. Detail of fitting surface where the differential correction vector $d = (3, 3)^T$ is applied to $\theta^{(1)} = (1, 1)^T$ giving a linear path of search in the parameter space. The terminal point for step factor $\lambda = 1$ is $\theta^{(2)} = (4, 4)^T$. The course of the scale factors induces what mostly is called overshooting.

situation we introduce weights w to act on d to produce paths closer to the solution θ^*.

Obviously the weights have to be chosen inversely proportional to the scale factors in newly obtained points $\theta^{(2)}$ on the fitting surface along the direction of Jd, therefore we define the kth component of the $p \times 1$ vector w by the ratio

$$w_k^{(1)}(\lambda) := \frac{h_k(\theta^{(1)})}{h_k(\theta^{(1)} + \lambda d^{(1)})}, \qquad k = 1(1)p \tag{5.2.1}$$

hence, denoting the step factor in the weighted direction by λ'

$$\theta^{(2)\prime} := \theta^{(1)} + \lambda'[w^D(\lambda)]^{(1)} d^{(1)} \tag{5.2.2}$$

giving new paths on the fitting surface (see Fig. 18 path 2 and 3, and Fig. 1).

In (5.2.2) values for λ' and λ have to be determined to make optimal progress. Solutions can be obtained by taking differentials or differences of scale factors depending on the availability of second derivatives of the condition function.

5.3 Differentials of scale factors

5.3.1 *Application*

Consider the kth coordinate of the vector in the denominator of (5.2.1), viz.

$$h_k(\theta^{(2)}(\lambda)) := h_k(\theta^{(1)} + \lambda d^{(1)}) \tag{5.3.1}$$

which can be expanded in a Taylor series with respect to the increment λd

$$h_k^{(2)}(\lambda) = h_k^{(1)} + \lambda(d^{(1)})^T \nabla h_k(\theta^{(1)}) + o(\lambda^2) \tag{5.3.2}$$

The scalar product can be written

$$(d_1,\ldots,d_p)\left(\frac{\partial h_k}{\partial \theta_1},\ldots,\frac{\partial h_k}{\partial \theta_p}\right)^T$$

in which the elaboration of the arbitrary lth term of the second vector is given by (2.7.1)

$$\partial_l h_k = \frac{(f_k, f_{kl})}{h_k}, \qquad l = 1(1)k$$

Using the matrix (2.5.10) the results can be collected as follows, for λ sufficiently small

$$\boldsymbol{h}^{(2)}(\lambda) = \boldsymbol{h}^{(1)} + \lambda \boldsymbol{h}^{d(1)} \boldsymbol{M}_{12}^{(1)} \boldsymbol{d}^{(1)}$$

Finally the weights (5.2.1) can be given by the vector

$$\boldsymbol{w}(\lambda) = (\boldsymbol{h} + \lambda \boldsymbol{h}^d \boldsymbol{M}_{12} \boldsymbol{d})^d \boldsymbol{h} \tag{5.3.3}$$

where vectors and matrices on the right-hand side have to be evaluated at $\boldsymbol{\theta} = \boldsymbol{\theta}^{(1)}$. Relative to $\boldsymbol{h}$ the expression becomes

$$\boldsymbol{w}(\lambda) = (\boldsymbol{1} + \lambda \boldsymbol{h}^d \boldsymbol{h}^d \boldsymbol{M}_{12} \boldsymbol{d})^d \boldsymbol{1} \tag{5.3.4}$$

5.3.2 *Minimum response*

The response S in case of weighted corrections taking $\lambda = \lambda'$ and dropping the prime can be obtained from

$$S(\lambda) = [f_0(\theta^{(1)} + \lambda w^D(\lambda) d)]^T f_0(\theta^{(1)} + \lambda w^D(\lambda) d) \tag{5.3.5}$$

to be evaluated at $\theta^{(1)}$. The general formula for the optimization condition is given in equation (2.5.14) and for one-dimensional search in (2.5.15). The derivative of the argument in (5.3.5) with respect to λ becomes

$$\frac{d\theta^{(2)\prime}(\lambda)}{d\lambda} = w^D(\lambda) d + \lambda \frac{dw^D(\lambda)}{d\lambda} d$$

The kth component gives, using (5.3.3) and $\{.k.\}$ to denote the kth component of $\boldsymbol{M}_{12}\boldsymbol{d}$

$$\frac{d\theta_k^{(2)\prime}(\lambda)}{d\lambda} = \frac{h_k^2}{h_k^2 + \lambda\{.k.\}} d_k + \lambda \frac{d}{d\lambda}\left(\frac{h_k^2}{h_k^2 + \lambda\{.k.\}} d_k\right)$$

which gives

$$\frac{d\theta_k^{(2)\prime}(\lambda)}{d\lambda} = \left(\frac{h_k^2}{h_k^2 + \lambda\{.k.\}}\right)^2 d_k \tag{5.3.6}$$

The optimization condition for a subminimum along (5.2.2) reads, differentiating (5.3.5) using (2.5.13), (2.4.12) and (2.5.15)

$$\frac{dS(\lambda)}{d\lambda} = \sum_{k=1}^{p} [f_0(\lambda), f_k(\lambda)] \left(\frac{h_k^2}{h_k^2 + \lambda\{.k.\}} \right)^2 d_k = 0$$

from which λ has to be solved. This can be done by means of iterative methods as are discussed in Chapter 9. Since $d\theta(\lambda)/d\lambda$ depends on λ – see (5.3.6) – the path along which exploration of the parameter space takes place is curved, giving a one-dimensional nonlinear search in the parameter space. The derivative of $\theta(\lambda)$ has to be evaluated for each iteration step in searching the subminimum.

It will be noted that $\left. \frac{d\theta^{(2)\prime}(\lambda)}{d\lambda} \right| \lambda = 0 =: \theta_\lambda(0)$ has the direction of d because of (5.3.6).

The properties of the weighting system for $\lambda = 0$ and $\lambda \to \infty$ are, referring to (5.3.3),

$$w^D(0) = I, \qquad \theta^{(2)\prime}(0) = \theta^{(1)}, \qquad \theta_\lambda(0) = d$$

$$w^D(\infty) = 0, \qquad \theta_k^{(2)\prime}(\infty) \to \theta_k^{(1)} + \frac{h_k^2}{\{.k.\}} d_k, \qquad k = 1(1)p, \qquad \theta_\lambda(\infty) = 0$$

so $\theta_k^{(2)\prime}$, $k = 1(1)p$, is bounded for $M_{12} \neq 0$. If $M_{12}d = 0$, for instance with a linear condition function, (5.3.3) gives $w = I$ and the weights cancel out.

Changing values of λ will not change the sign of $\theta_\lambda(\lambda)$ because of the square in (5.3.6). This means that, depending on the sign of d_k the values of $\theta_k^{(2)\prime}$, $k = 1(1)p$, increase or decrease monotone although the search is nonlinear. In Fig. 18 three paths of search are sketched. Path 1 is valid for (3.5.1) with $s = d$, path 2 for (5.2.2) with $\lambda =$ constant and path 3 for (5.2.2) with $\lambda = \lambda'$.

Fig. 1 gives an example of paths obtained in this way on an actual fitting surface.

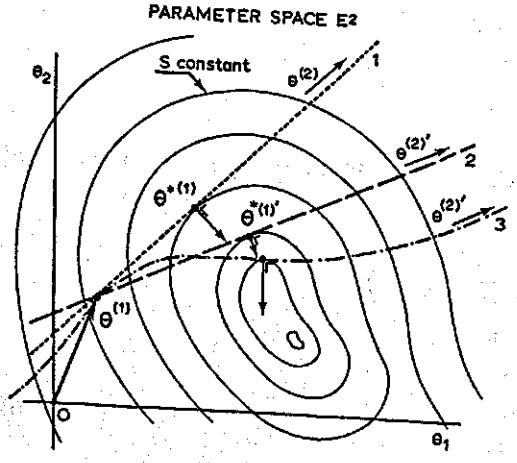


Fig. 18. Differential corrections applied in three algorithms. In path 1 given by $\theta^{(2)} = \theta^{(0)} + \lambda d^{(1)}$ according to the modified Gauss-Newton algorithm. In path 2 given by $\theta^{(2)\prime} = \theta^{(1)} + \lambda'[w^D(\lambda^{*(1)})]^{(1)} d^{(1)}$ by weighting differential corrections with constant values of scale factor differentials. In path 3 given by $\theta^{(2)\prime} = \theta^{(1)} + \lambda'[w^D(\lambda')]^{(1)} d^{(1)}$ where the weights depend on the step factor λ'.

5.4 Differences of scale factors

5.4.1 *Application*

To avoid the determination of second derivatives of the condition function a difference method can be applied. We assume that a second point is known numerically which means that (5.2.1) can be calculated. Suppose this second point be optimal in the direction $d^{(1)}$ then, with $h = h(\lambda)$

$$w^{(1)}(\lambda^{*(1)}) = [h^{(1)}(\lambda^{*(1)})]^d h^{(1)}(0) \tag{5.4.1}$$

which can be inserted in (5.2.2) giving, again dropping the prime for λ,

$$\theta^{(2)\prime}(\lambda) = \theta^{(1)} + \lambda[w^D(\lambda^{*(1)})]^{(1)} d^{(1)} \tag{5.4.2}$$

from which the optimal value of λ is to be determined.

Analogous to (5.3.4) we can write

$$w = [I + h^{d(1)}(h^{(2)} - h^{(1)})]^d I \tag{5.4.3}$$

where $h^{(2)}$ is supposed to be evaluated at $\theta^{*(1)}$ (Fig. 18). In this case the scale factor at an optimal point in the weighted direction does not agree with that at $\theta^{*(1)}$. To maintain correspondence between both, an alternative expression for (5.4.2) would read generalizing (5.4.1)

$$\theta^{(2)\prime}(\lambda) = \theta^{(1)} + \lambda[w^D(\lambda)]^{(1)} d^{(1)}$$

again giving a nonlinear direction of search.

5.4.2 *Minimum response*

In case of differences of scale factors the parameter function (5.4.2) is again a linear and strict function of d. This means that the direction along the path of search is given by

$$\frac{d\theta^{(2)\prime}(\lambda)}{d\lambda} = [w^D(\lambda^{*(1)})]^{(1)} d^{(1)}$$

independent of the step factor λ. Progress is made along a straight line in the parameter space. Optimal values of λ can be obtained by methods given in Chapter 9.

5.5 Example

For the condition function and data given in Appendix 2.3 three methods have been compared (Fig. 19). The three methods converge on the same terminal point. The starting value gives $S = 976.40$. The modified Gauss-Newton algorithm gives convergence in 25 cycles under default accuracy options (Chapter 13). When using weights according to differences of scale factors in (5.4.3) with optimal first step length the

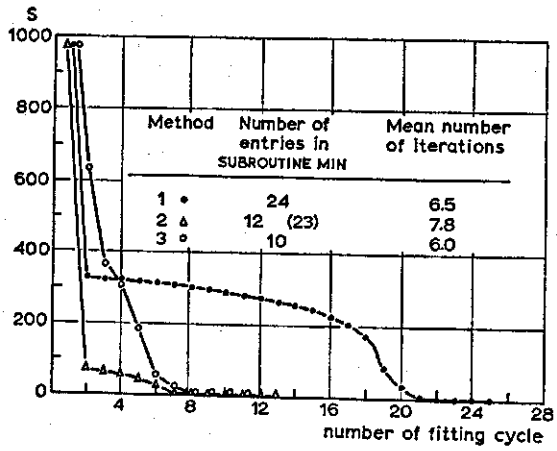


Fig. 19. Decrease of sum of squares for the problem in Appendix 2.3 according to three algorithms based on the differential correction vector $\boldsymbol{d}$. 1 modified Gauss-Newton; 2 and 3 differential corrections weighted with the ratio of scale factors; 2 with use of differences, the number between parentheses includes full optimal search to obtain $h(\lambda^{*(n)})$ in each first direction; 3 with use of differentials.

response decreases in the first cycle to $S = 71.59$. Under the same options the number of cycles necessary for the same accuracy is 13. Differentials of scale factors produce weights by means of (5.3.4) that cause convergence in 11 cycles.

As regards the approach to the terminal point, Fig. 19 shows that the modified Gauss-Newton algorithm gives a response $S < 100$ after 18 cycles. Weights based on differences of scale factors achieve this in the first cycle but it will be remembered that scale factors from two points must be available and so in each cycle two subminima have to be calculated, the first to obtain $\theta^{*(n)}$, the second to obtain $\theta^{*(n)\prime}$. With weights based on differentials of scale factors 6 cycles are needed to obtain $S < 100$. Now only one subminimum has to be determined in each fitting cycle. The least sum of squares appears to be $S(\theta^{(t)}) = 1.83$, which is found with all three methods.

The results, depicted in Fig. 19, were obtained by the default main program NLV, the Modification 8.1 and the Modification 9.2, respectively. (See Appendix 1.5).

6 Correction for curvature by back projection

6.1 General

In Chapter 5 a method was developed to accelerate convergence. Use was made of the scale factors of the parametric curves. Because of their curvature the locus of the terminal point of the position vector, which proceeds on the fitting surface along a linear direction of search in the parameter space, can deviate in undesired directions.

In the following sections a method is developed that measures and corrects for the departure from the desired direction of correction. This method consists of orthogonal projection of the path of search in the fitting surface on the tangent plane at the current starting point. Because of the fact that the direction of search, which in the present case is determined by differential corrections, is found with the aid of the same tangent plane, the method is called the back projection method.

In this chapter the situation at $\theta^{(n)}$ is considered and consequently the matrices J and M are to be evaluated at this point.

Optimization iterations pertaining to back projection will be denoted by n'.

6.2 Mathematical description

Since the tangent plane is spanned by the column vectors of J, the projection of the vector f_0 on the tangent plane is Jd (Fig. 20), where d is obtained by (2.4.13).

An arbitrary point in the direction d in the parameter space is denoted by $\theta^{(n+1)}(\lambda) = \theta^{(n)} + \lambda d$. In fitting practice $\theta^{(n+1)}$ will be chosen to represent $\theta^{*(n)}$, obtained with $\lambda = \lambda^{*(n)}$, and the orthogonal projection ${}^{A}\theta^{(n+1)}$ of this point on the tangent plane at $\theta^{(n)}$ will be considered. The normal for this case reads according to (2.4.9)

$$N = J^T[f(\theta^{(n+1)}) - f(\theta^{(n)})] \tag{6.2.1}$$

The projection of this difference vector on the tangent plane can be written as Jb, see Fig. 20. The solution of b is obtained from the inverse matrix M^{-1} (earlier stored in the computer memory) by

$$b = M^{-1}N \tag{6.2.2}$$

which is the vector of back projection.

Now the situation is as follows. Emanating from $\theta^{(n)}$, the direction of search on the fitting surface is along Jd in the tangent plane. On the fitting surface the path is curved and it terminates in the (optimal) point $\theta^{(n+1)}$. Back projection furnishes the informa-

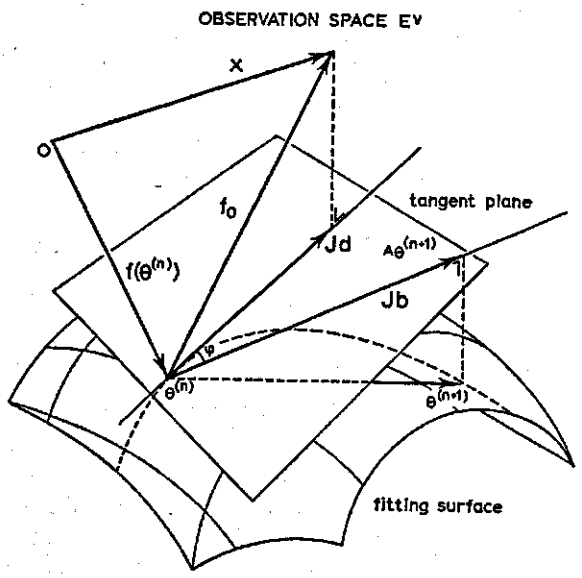


Fig. 20. Illustration of the principle of the back projection method. The point $\theta^{(n+1)}$ on the fitting surface is projected on the tangent plane giving ${}^{A}\theta^{(n+1)}$ indicating that the vector Jb makes an angle ϕ with the direction obtained with differential corrections.

tion that the subminimum is found in the direction Jb in the tangent plane. Since the new direction is produced by the disturbing influence of the curvature of the parametric curves, corrections have to be applied to find a back projection that is closer to the vector Jd in the tangent plane.

6.3 Two-dimensional linear search

6.3.1 *Correction for curvature*

In Fig. 20 the situation is sketched where back projection gives a direction Jb too far to the right of Jd. Correction for this can be made by starting in a direction to the left of the vector Jd for which we take Js (Fig. 21). The magnitude of the correction can be taken equal to the angle ϕ between the vectors Jd and Jb. The new direction in the two-dimensional tangent subplane spanned by Jd and Jb can be written as the linear combination $Js = \alpha Jd + \beta Jb$.

In this tangent subplane we have

$$\cos\phi = \frac{d^T J^T Jb}{\|Jd\| \cdot \|Jb\|} = \frac{d^T Mb}{(d^T Md)^{\frac{1}{2}}(b^T Mb)^{\frac{1}{2}}} \tag{6.3.1}$$

From Fig. 22 we can derive the values for α and β to obtain the desired correction, namely

$$Js = 2\cos\phi\, Jd - \frac{\|Jd\|}{\|Jb\|} Jb \tag{6.3.2}$$

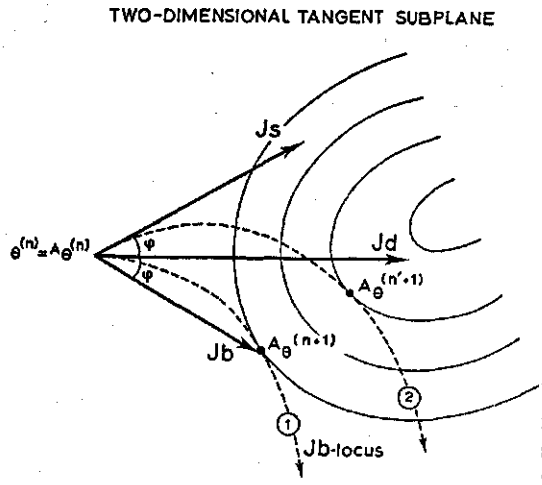


Fig. 21. Starting in the direction ***Jd*** in the tangent plane a path of search on the fitting surface is produced whose image is denoted by curve 1. The tangent subplane is spanned by the vectors ***Jd*** and ***Jb***. Curve 2 is the image of a path of search on the fitting surface when starting in the direction ***Js*** which is a linear combination of ***Jd*** and ***Jb***.

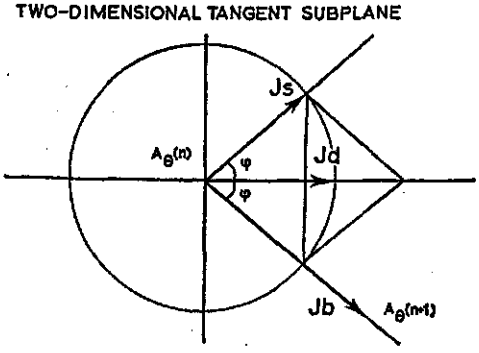


Fig. 22. Construction of the vector ***Js*** in the two-dimensional tangent subplane as suggested by Fig. 21.

The length of this vector is obtained from

$$\|\boldsymbol{Js}\|^2 = 4\cos^2\phi\ \|\boldsymbol{Jd}\|^2 - 4\cos\phi\,\frac{\|\boldsymbol{Jd}\|}{\|\boldsymbol{Jb}\|}\,\boldsymbol{d}^T\boldsymbol{M}\boldsymbol{b} + \|\boldsymbol{Jd}\|^2$$

In virtue of (6.3.1) the first two terms at the right-hand side vanish and so $\|\boldsymbol{Js}\| = \|\boldsymbol{Jd}\|$.

The solution of $\boldsymbol{s}$ can be obtained from (6.3.2) and reads

$$\boldsymbol{s} = 2\cos\phi\ \boldsymbol{d} - \frac{\|\boldsymbol{Jd}\|}{\|\boldsymbol{Jb}\|}\,\boldsymbol{b} \tag{6.3.3}$$

To avoid matrix calculation this solution can be approximated by one that would be obtained in the two-dimensional tangent subspace with $\boldsymbol{M} = \boldsymbol{I}$ which gives

$$\boldsymbol{s} = 2\cos\phi\ \boldsymbol{d} - \frac{\|\boldsymbol{d}\|}{\|\boldsymbol{b}\|}\,\boldsymbol{b} \tag{6.3.4}$$

where $\cos \phi = (\boldsymbol{d},\boldsymbol{b})/(\|\boldsymbol{d}\| \, . \, \|\boldsymbol{b}\|)$.

The search according to (6.3.4) – and analogous for (6.3.3) – can be done by

$$\boldsymbol{\theta}^{(n'+1)}(\lambda) = \boldsymbol{\theta}^{(n)} + \lambda \boldsymbol{s} \tag{6.3.5}$$

where a first approximation to the step factor can be taken equal to $\lambda^{*(n)}$.

Apart from the iteration to find $\lambda = \lambda^{*(n')}$ in (6.3.5), the method of back projection itself can be considered an iterative process. Optimization can be employed for ϕ and λ both, considering them a pair of algorithm parameters. Defining $c := 2 \cos \phi$ the algorithm parameter vector becomes $\boldsymbol{\lambda} := (\lambda,c)^T$ and (6.3.5) then reads

$$\boldsymbol{\theta}^{(n'+1)}(\boldsymbol{\lambda}) = \boldsymbol{\theta}^{(n)} + \lambda \boldsymbol{s}(c) \tag{6.3.6}$$

finally giving for the sum of squares the expression

$$S(\boldsymbol{\theta}(\boldsymbol{\lambda})) = S\left(\boldsymbol{\theta}^{(n)} + \lambda\left(c\boldsymbol{d} - \frac{\|\boldsymbol{d}\|}{\|\boldsymbol{b}\|}\boldsymbol{b}\right)\right) \tag{6.3.7}$$

6.3.2 *Directions of search*

The conditions for a minimum are given by (2.5.14) and read in the present case

$$\frac{\partial S}{\partial \lambda} = [\boldsymbol{g}(\boldsymbol{\lambda})]^T\left(c\boldsymbol{d} - \frac{\|\boldsymbol{d}\|}{\|\boldsymbol{b}\|}\boldsymbol{b}\right) = 0 \tag{6.3.8}$$

$$\frac{\partial S}{\partial c} = [\boldsymbol{g}(\boldsymbol{\lambda})]^T(\lambda \boldsymbol{d}) = 0 \tag{6.3.9}$$

to be solved for λ and c.

The tangent subspace spanned by the vectors $\boldsymbol{d}$ and $\boldsymbol{b}$ is considered. The directions of search are given by differentiating the parameter function (6.3.6) and (6.3.4)

$$\frac{\partial \boldsymbol{\theta}}{\partial \lambda} = \boldsymbol{s}(c) = c\boldsymbol{d} - \frac{\|\boldsymbol{d}\|}{\|\boldsymbol{b}\|}\boldsymbol{b} \tag{6.3.10}$$

$$\frac{\partial \boldsymbol{\theta}}{\partial c} = \lambda \boldsymbol{d} \tag{6.3.11}$$

which is also clear from (6.3.8) and (6.3.9).

For a fixed value of c, $c^{(n')}$ say, the vector (6.3.10) emanates from $\boldsymbol{\theta}^{(n)}$ as sketched in Fig. 23. This means that the gradient in (6.3.8) can be used to solve λ iteratively, starting with $\lambda = \lambda^{*(n)}$, producing the new optimal value $\lambda^{*(n')}$. Keeping this value constant, c can be solved iteratively from the gradient in (6.3.9) to obtain $c^{*(n')}$. This is performed along $\lambda^{*(n')}\boldsymbol{d}$ emanating from the point

$$\boldsymbol{\theta}^{(n)} - \lambda^{*(n')}\frac{\|\boldsymbol{d}\|}{\|\boldsymbol{b}\|}\boldsymbol{b} := {}^{A}\boldsymbol{\theta}^{(n')}$$

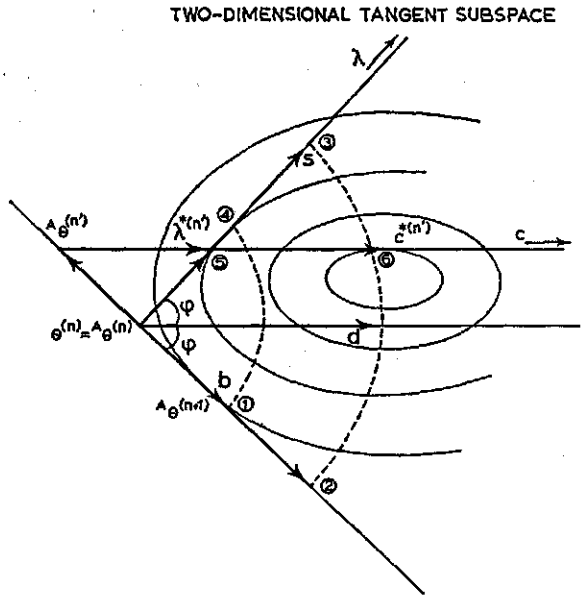


Fig. 23. Relationship between directions of search as used in two-dimensional search by back projection. The steps are: 1 back projection of $\theta^{(n+1)}$ producing the vector b in the tangent subspace; 2 determination of a vector of length $\|d\|$ in the direction of b; 3 reflection of this vector in d to produce s; 4 initial step along s with a vector of length $\|b\|$; 5 determination of the optimal value of the step factor λ to obtain $\lambda^{*(n')}$; 6 optimization of the angle ϕ by means of the parameter c starting at ${}^A\theta^{(n')}$ to obtain $c^{*(n')}$.

with initial step factor $c^{(n')}$ as illustrated by point 5 in Fig. 23.

The optimization in the directions distinguished can be carried out with methods described in Chapter 9.

6.3.3 *Properties of the algorithm parametric curves*

In contrast with parametric curves for the condition function, parametric curves for $\lambda = (\lambda,\phi)^T$ tend toward orthogonality when $n \to \infty$.

Consider the two-dimensional tangent plane spanned by Jd and Jb. The condition function reads

$$f(\theta(\lambda)) = f\left(\theta^{(n)} + \lambda\left(cd - \frac{\|Jd\|}{\|Jb\|}b\right)\right), \qquad c = 2\cos\phi$$

The direction vectors to the curvilinear coordinates of the (λ,ϕ)-system are given by

$$f_\lambda = J\left(cd - \frac{\|Jd\|}{\|Jb\|}b\right)$$

$$f_\phi = -2\lambda \sin\phi\, Jd$$

where use was made of (2.5.4) with $J_\lambda^f = (f_\lambda, f_\phi)$ and of (6.3.6) and (6.3.4).

The scalar product becomes

$$f_\lambda^T f_\phi = -2\lambda \sin\phi \left(cd - \frac{\|Jd\|}{\|Jb\|}b\right)^T Md$$

which tends to zero for $n \to \infty$ because in the Gauss-Newton algorithm $d^{(n)} \to 0$.

6.4 Two-dimensional circular search

The method described in Section 6.3.2 gives linear search in two directions. The reflection of the vector Jb in the straight line given by the vector Jd in Fig. 21 can be replaced by a circular search when moving from point 2 to point 3 in Fig. 23, following the circular path that connects the terminal point of the vectors $(\|d\|/\|b\|)b$ and s.

6.4.1 *Correction for curvature*

Correction formulas are derived for the metric $M = J^TJ$. An orthogonal base for the two-dimensional tangent subplane is determined (see Fig. 24) by taking a vector perpendicular to Jd and of the same length

$$Jz := \frac{J}{|\sin\phi|}\left(d\cos\phi - \frac{\|Jd\|}{\|Jb\|}b\right) \tag{6.4.1}$$

where
$$\cos\phi = \frac{d^TMb}{(d^TMd)^{\frac{1}{2}}(b^TMb)^{\frac{1}{2}}} \tag{6.4.2}$$

and
$$z^TMd = 0 \tag{6.4.3}$$
$$\|Jz\| = \|Jd\|$$

The relationship between z, d and b is obtained from (6.4.1).

The metric can be taken unity ($M = I$) when applying the formulas (6.4.1) through (6.4.3) in the two-dimensional tangent subspace analogous to the conversion of (6.3.3) into (6.3.4). This is done in the remainder of this section.

An arbitrary vector that is a linear combination of the base vectors z and d with properties analogous to (6.4.3), namely $(z,d) = 0$ and $\|z\| = \|d\|$, is given by

$$s(\psi) = z\sin\psi + d\cos\psi \tag{6.4.4}$$

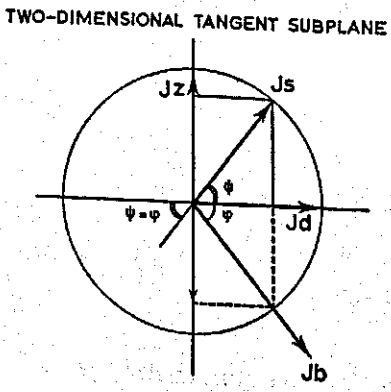


Fig. 24. Construction of the vector Js in the two-dimensional tangent subplane as a linear combination of the orthogonal vectors Jd and Jz. The particular case $\psi = \phi$ is illustrated.

which has a length equal to that of d. Taking $\psi = \phi$ equation (6.4.4) reduces to (6.3.4) using, see (6.4.1),

$$z = \frac{1}{|\sin \phi|}\left(d \cos \phi - \frac{\|d\|}{\|b\|} b\right)$$

With (6.4.4) a new parameter vector for two-dimensional circular search is, defining $\lambda := (\lambda,\psi)^T$,

$$\theta^{(n'+1)}(\lambda) = \theta^{(n)} + \lambda s(\psi) \tag{6.4.5}$$

giving for the sum of squares

$$S(\theta(\lambda)) = S(\theta^{(n)} + \lambda z \sin \psi + \lambda d \cos \psi)$$

6.4.2 *Directions of search*

The directions of search in each point of the two-dimensional (λ,ψ)-coordinate system is given by formulas analogous to (6.3.10) and (6.3.11). In the present case they are

$$\frac{\partial \theta}{\partial \lambda} = z \sin \psi + d \cos \psi = s(\psi) \tag{6.4.6}$$

$$\frac{\partial \theta}{\partial \psi} = \lambda (z \cos \psi - d \sin \psi) \tag{6.4.7}$$

The optimization conditions are again

$$[g(\lambda)]^T\left(\frac{\partial \theta}{\partial \lambda}\right) = 0 \text{ and } [g(\lambda)]^T\left(\frac{\partial \theta}{\partial \psi}\right) = 0$$

The value of ψ can be determined from the second condition, keeping λ constant. Then the first condition is used to solve λ, keeping ψ constant. Optimization of each of the algorithm parameters can be achieved by methods described in Chapter 9.

6.4.3 *Properties of the algorithm parametric curves*

The directions of search in (6.4.6) and (6.4.7) are orthogonal since the scalar product of the vectors $\partial\theta/\partial\lambda$ and $\partial\theta/\partial\psi$ is

$$\lambda z^T z \sin \psi \cos \psi - \lambda z^T d(\sin^2 \psi - \cos^2 \psi) - \lambda d^T d \sin \psi \cos \psi = 0$$

This holds because z is perpendicular to d with equal length.

6.5 Example

Results of optimization using back projection techniques are given for the condition function and data mentioned in Appendix 2.3.

In Fig. 25 the decrease of the sum of squares is plotted against cycle number for

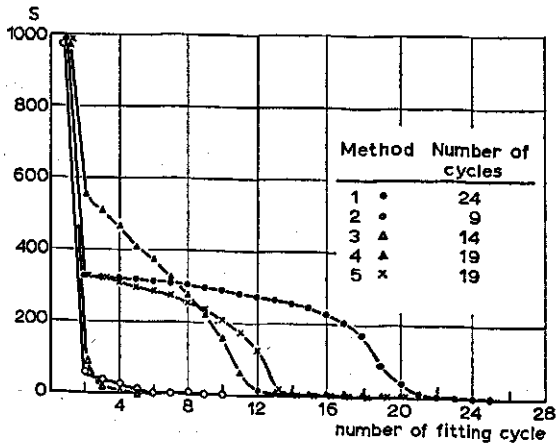


Fig. 25. Decrease of sum of squares for the problem in Appendix 2.3 according to five algorithms. 1 modified Gauss-Newton; 2 through 5 methods of back projection; 2 noncircular search with metric I; 3 circular search with metric I, after five cycles transfer was made to modified Gauss-Newton; 4 noncircular search with metric M; 5 circular search with metric M. The course of the partial cosines for algorithm 3 is plotted in Fig. 32.

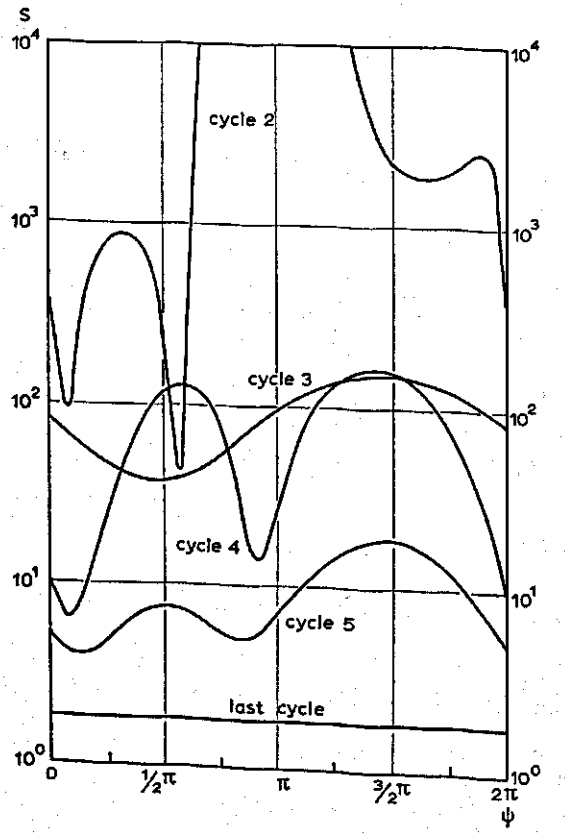


Fig. 26. Relationship between sum of squares S and angle ψ for the direction of circular search with metric I for various fitting cycles. After five cycles transfer was made to modified Gauss-Newton.

four methods of back projection. In this example the use of the metric I appears to be more efficient than that of the metric M. The curves in this figure can be compared with those given in Fig. 19, taking the unweighted modified Gauss-Newton algorithm as reference.

A large area of the fitting surface is explored by back projection techniques as can be seen from Figures 26 and 27 for circular search. They give the relationship between the sum of squares obtained by rotating the vector s respectively Js about the starting point (see Fig. 24 and equation (6.4.4)). To avoid an extension of the search in directions that produce too large values for the sum of squares, a reduction to the angle ϕ in (6.3.2) and (6.4.1) is applied as a starting value when $\cos \phi$ is negative. In fitting practice it then appears to be efficient to change the sign of the vector of back projection b. However, in some of such cases the sign of the direction of search that is applied in subprogram MIN must also be changed to obtain a negative slope at the initial point, (cycle 2 and cycle 13 in Fig. 27). Since in the present case the search is nonlinear, the program statement that would change the sign was temporarily deleted, giving an optimal search for λ alone at $\psi = 0$. Otherwise execution would have been terminated after the second time a positive slope at the initial point was found in the same cycle.

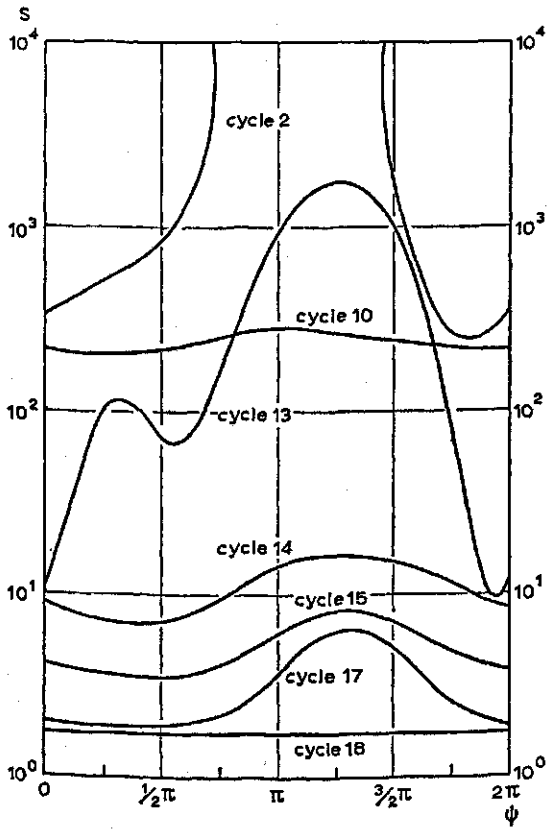


Fig. 27. Relationship between sum of squares S and angle ψ for the direction of circular search with metric M, for various fitting cycles.

The convergence to the final solution is slow after the sum of squares has decreased to values below $S = 5$, say. For circular search with metric $\boldsymbol{I}$ the algorithm was transferred to the Gauss-Newton method with optimal step length after the angle ϕ decreased below 0.02 radians or $\cos \phi < 0.0008$.

The best result for the given example was obtained with a non-circular search with metric $\boldsymbol{I}$. The number of cycles necessary to obtain default accuracy decreased from 24 to 9 (Fig. 25).

II Special procedures

7 Structure of the program

7.1 General

The computer program is written in FORTRAN and was run on a CDC 6600 computer using Scope 3.3 UPDATE features (Appendix 1.1).

A program written for the application of different algorithms has to be compiled of subprograms that contain the statements of the subroutines of the entire procedure. Particular algorithms then can be employed by choosing the proper sequence of subroutines in the subprograms by calling them from the appropriately modified main program. Use is made of SUBROUTINE subprograms only.

The program is divided into three parts. The first part consists of subprograms that have to be updated for each particular condition function that is to be optimized. The second and third part consist of subprograms that contain algorithm statements and that need no updating except for the dimensions of the arrays, if desired. Subprograms are called by the main program NLV (Appendix 1.2) which can be modified to suit various algorithms (Appendix 1.5).

Variables that occur in different subprograms are linked by the appropriate arguments or by COMMON statements. Variables that occur in COMMON statements are subdivided into two blocks. Those whose dimensions of arrays are condition function dependent are collected in *COMDECK DVAR, those whose dimensions are algorithm dependent are collected in *COMDECK DFIX together with nonsubscripted variables. Condition function dependent subscripted variables are updated with dimension comdecks as described in Appendix 1.1. See also example given in Appendix 1.4.1.

To use the program for investigation either of the condition function and its properties or the convergence process, options are included that are not part of the algorithm. Options can also be used to modify the main program NLV (Chapter 12 and Appendix 1.5).

7.2 Condition function statements

Subprograms written as a framework for new condition functions are contained in the first part of the entire program. They serve two purposes.

Administration The SUBROUTINE INITL was written to produce the specific headings for problem identification, to initialize variables and counters and to govern

Table 4. Update *COMDECKs called by SUBPROGRAMs mentioned. See also Appendix 1.3 and Appendix 1.4.1.

Contents of the comdecks	Comdeck name	Called by
Headings	XINITL	INITL
Dimensions	NDIMRD	READ
READ instructions	RDDATA	READ
WRITE instructions	WTDATA	READ
Dimensions	NDIMFN	FNCTN
Function evaluation	NFNCTN	FNCTN
WRITE instruction	WFNCTN	FNCTN
Dimensions	NDIMDF	DFDA
First derivatives	NDFDA	DFDA
Dimensions	NDIMD2	D2FDA
Second derivatives	ND2FDA	D2FDA
Data	DATA	
Data to be copied to file MYDATA, see Section 7.6.		

the desired output.

Function evaluation It is evident that for each new problem to be optimized, the condition function and its derivatives with respect to the parameters is to be programmed. To be able to apply methods at the level of information available, separate subprograms were written for the evaluation of the condition function (SUBROUTINE FNCTN), of the first derivatives (SUBROUTINE DFDA) and of the second derivatives (SUBROUTINE D2FDA).

A new function and its derivatives can be inserted in the framework of the program by means of update decks that are called by the subprograms during an update run before they are written to the compile file. Update decks to be inserted in the different subprograms are mentioned in Table 4.

7.3 Algorithm statements

Subroutines of gradient methods were written in separate subprograms which can be linked to obtain modifications of standard methods (Chapter 12). These subprograms form the second part of the entire program. The third part consists of special procedure subprograms. They need not necessarily be loaded when using the default deck structure of the main program NLV.

7.3.1 *Gradient subroutines*

Algorithm partitioning The level at which an algorithm can be applied furnishes the criterion for partitioning in subprograms. Fig. 28 gives a conspectus of the compo-

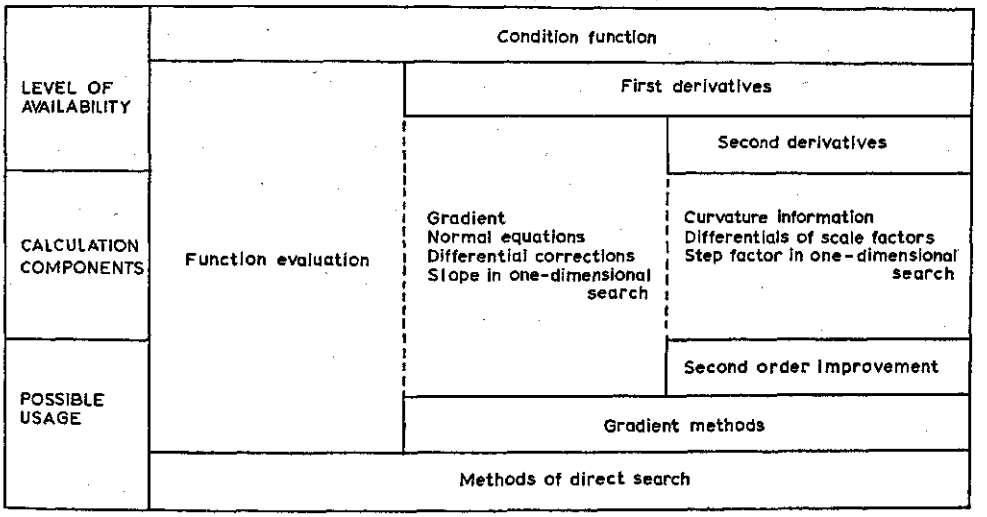


Fig. 28. Conspectus of components of calculations and their possible usage following from the level of availability.

nents of the calculations that are needed for further elaborations. Apart of the subprograms mentioned in Table 4, subprograms were written for the determination of the gradient and the composition of the normal equations (NRMEQ), for their solution to produce differential corrections (SOLVE) and for one-dimensional optimal search (MIN). The determination of curvature and of differentials of scale factors are spread through some appropriate subprograms.

Process continuation The convergence process is to be followed carefully in order to detect whether convergence occurs and to decide whether execution can be terminated. Decisions based on stopping criteria are taken in two steps. The first step is to investigate by means of subprogram HOWA the situation on the fitting surface at the initial point, point A say. When the decision is made to proceed with a fitting cycle at a new point B, the second step is to start the next cycle with the replacement of point A by point B which is carried out in the subprogram AISB. After this the first step is repeated, and so on, till the stoppig criterion is satisfied.

Administration The output contains the main results of the calculations with respect to optimal parameter values and corresponding function values, as well as intermediate results concerning the applied algorithm and its properties. Subprogram LISTING controls the output and the listing of tables produced by the first and second parts of the program, according to the demands of the user.

7.3.2 *Particular subroutines*

Exclusion of parameters Parameters in condition functions can be kept constant, and thus excluded from the optimization process, to particularize or simplify the function. Examples were given in Section 4.8, where a choice was made between sequential

and nonsequential use. Another possibility is a partitioning of the parameters into two groups, the first consisting of those that occur nonlinearly, the second of those that occur linearly in the condition function. Grouping and selective use of parameters once can be applied in the default deck structure of the main program. Grouping and partitioning can be saved, however, by means of the subprogram BLOCK that can contain 5 different groupings and governs their subsequential use. Details of grouping are given in Chapter 8.

Additional subprograms The method of back projection developed in Chapter 6 is programmed in subprogram BACK. Two further subprograms are included in the third part of the program. A method of finding controlled paths on the fitting surface, given in Chapter 11, is made operational in subprogram TRACK. Possibilities to extrapolate intermediate parameter values, as developed in Chapter 10, are available in subprogram LIHYPEX.

7.4 Convergence characteristics

7.4.1 *Type of convergence*

From personal experience it was found that in many least squares problems both the sum of squares with respect to the fitting surface S and the sum of squares with respect to the tangent plane ${}^{A}S$ decrease. To distinguish between this situation and the one where $S^{(n+1)} < S^{(n)}$ while at the same time ${}^{A}S^{(n+1)} > {}^{A}S^{(n)}$, the latter situation will be called Type I convergence and the former Type II convergence.

In Fig. 29 the principle of both types is sketched. An example is given in Section 7.4.3.

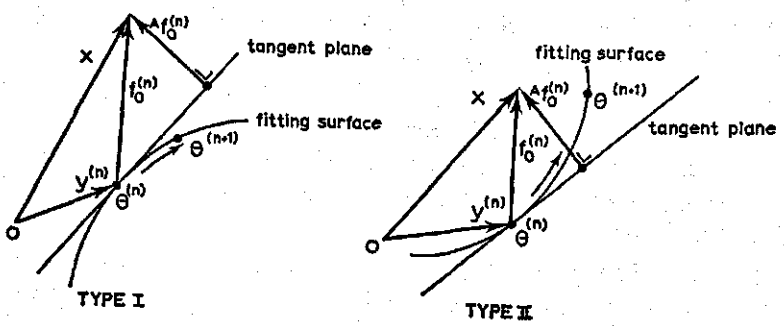


Fig. 29. Principle of the two types of convergence distinguished. Proceeding on the fitting surface in the direction of the arrow, the properties of Type I convergence are $S^{(n+1)} = \|f_0^{(n+1)}\|^2 < S^{(n)} = \|f_0^{(n)}\|^2$ and ${}^{A}S^{(n+1)} = \|{}^{A}f_0^{(n+1)}\|^2 > {}^{A}S^{(n)} = \|{}^{A}f_0^{(n)}\|^2$. Type II convergence is characterized by $S^{(n+1)} < S^{(n)}$ and ${}^{A}S^{(n+1)} < {}^{A}S^{(n)}$.

7.4.2 *Stopping criteria*

Stopping criteria are used for economic reasons. They can be based on several quantities that give information on the state of the convergence process in consecutive fitting cycles. Quantities that converge to an unknown value, as the sum of squares and the parameter values, give this information in a relative sense. Quantities based on the orthogonality of the vector f_0 on the fitting surface in the final solution, give this information in an absolute sense.

A sequence of parameter values $\theta^{(n)}$, $n \to \infty$, is assumed to converge to θ^*. Several criteria can be put forward to establish $\theta^{(t)}$ to be a terminal point of the sequence, where $t < \infty$. The criterion δ_S for the sum of squares reads

$$\begin{aligned} &\text{if} \quad |S^{(n)} - S^{(n+1)}| < \delta_S, \ \delta_S > 0 \\ &\text{then} \quad \theta^{(t)} := \theta^{(n+1)} \end{aligned} \tag{7.4.1}$$

For the parameters this becomes

$$\begin{aligned} &\text{if} \quad |\theta_k^{(n)} - \theta_k^{(n+1)}| < \delta_k, \ \delta_k > 0 \\ &\text{then} \quad \theta_k^{(t)} := \theta_k^{(n+1)} \\ &\text{where} \quad k = 1(1)p \end{aligned} \tag{7.4.2}$$

Both criteria can be made relative to the order of magnitude of the values of S and θ_k respectively by dividing the expressions by the value obtained in the nth cycle, thus expressing δ in fractions. These criteria do not make use of first derivatives of the condition function, so the rules do not guarantee that a neighbourhood of θ^* has been reached because (7.4.1) and (7.4.2) do not depend on the condition for a minimum.

With the aid of first derivatives stronger criteria can be put forward. They depend on the consequences of $\nabla S = 0$. For nonsingular M^{-1}, equation (2.4.13) results in $d = 0$ for $N(\theta^*) = 0$. Application of (2.2.11), where $J^T f_0 = N$, produces a further criterion in that the components of the vector of partial cosines c have to be zero. From Fig. 6 it is obvious that at θ^* the vector f_0 and the vector ${}^A f_0$ coincide, hence giving zero difference in length when the iteration process terminates.

The foregoing characteristics can also be expressed by the cosine of β (Fig. 6), requiring that the total cosine tends toward zero in consecutive fitting cycles, so

$$\cos \beta = \frac{f_0^T J d}{\|f_0\| \cdot \|Jd\|} \to 0 \tag{7.4.3}$$

In fitting practice the criterion (7.4.1) has often been found too weak; the criterion $\|N\| < \delta_N$, $\delta_N > 0$, appears to be one of the strongest. Experience learned that the above given criteria probably can be ordered according to the following sequence from weak to strong, see also (2.4.16),

$$\left\{\frac{|S^{(n)} - S^{(n+1)}|}{S^{(n)}}\right\}_\delta, \{|\sqrt{S} - \sqrt{{}^A S}|\}_\delta, \{c\}_\delta, \{\cos \beta\}_\delta, \{\|d\|\}_\delta, \{\|N\|\}_\delta \tag{7.4.4}$$

where $\{u\}_\delta$ stands for $\underset{n\to t}{u^{(n)}} < \delta$, $\delta > 0$, $\delta = \delta^T I_1$ and $\delta_k = \delta_l$, $k = 1(1)p$, $l = 1(1)p$.

It proved that the partial cosines provide the most efficient criterion even if their absolute values may show an increase in the early stages of the convergence process although the sum of squares is monotone decreasing. The partial cosines give information on each parameter separately. They also give a direct geometrical interpretation in the observation space with known final values $c = 0$. This value does not depend on the step length in one-dimensional search, contrary to (7.4.2) which gives for linear and strict parameter functions with $\theta_k^{(n+1)} = \theta_k^{(n)} + \lambda s_k$

$$|\theta_k^{(n)} - \theta_k^{(n+1)}| = \lambda |s_k|$$

which will satisfy the criterion when

$$\lambda < \delta_k / |s_k|$$

The total cosine is related to the partial cosines since for an arbitrary direction s equation (7.4.3) can be written, with (2.2.11) in the form of the row vector

$$c^T = f_0^T J h^D \| f_0 \|^{-1}$$

as

$$\cos \beta = \frac{c^T h^D s}{\| Js \|}$$

giving for $s = s_k I_k$ the equality $\cos \beta = c_k$. For nonsingular M^{-1} the total cosine tends toward zero when $c \to 0$ using the direction of differential corrections $s = d = M^{-1}N$.

In the computer program cosine criteria are set for the partial cosines. The default value $\delta = 0.001$, in FORTRAN defined by COSCRIT = 3, requires that at least three leading zero decimals appear in the results for the partial cosines. On basis of this criterion it is decided in subprogram HOWA whether or not further fitting cycles are to be produced. The same criterion is used on the total cosine in an informative sense.

7.4.3 *Example*

For the condition function and the data mentioned in Appendix 2.3 results on the use of convergence characteristics are depicted in Figs 30 and 31. Fig. 30 illustrates that in this example the process has Type II convergence during the first 14 cycles of the modified Gauss-Newton algorithm. It has been found that in general this type of convergence is slow. During the cycles 14 to 19 the convergence is of Type I. Fig. 31 shows the course of the total and the partial cosines. They are not monotone decreasing although the sum of squares, depicted in Fig. 30, does. Only during the last few cycles the partial cosines decrease simultaneously. A typical example of the behaviour of partial cosines is shown in Fig. 32, which gives cosines for the fitting process that was discussed in Section 6.5 (see Fig. 25, curve 3).

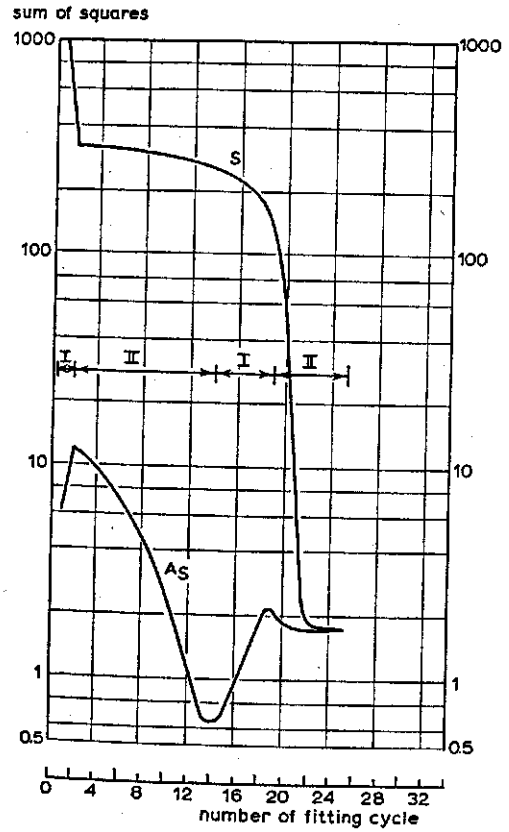


Fig. 30. Decrease of sum of squares S with respect to the fitting surface and of sum of squares ${}^{A}S$ with respect to the tangent plane, in subsequent fitting cycles. When S decreases and ${}^{A}S$ increases the convergence is of Type I; when both sum of squares decrease the convergence is of Type II (see also Fig. 29).

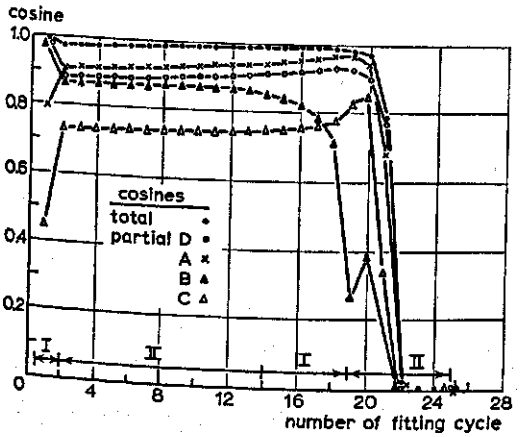


Fig. 31. Relationship between cosines, total and partial, for the parameters D, A, B and C in the condition function of Appendix 2.3, and number of fitting cycle for the modified Gauss-Newton algorithm. Although the sum of squares is monotone decreasing (Fig. 30 curve S) this is not the case for the cosines. During the last few fitting cycles they suddenly approach zero. The numbers I and II refer to the type of convergence as given in Figs. 29 and 30.

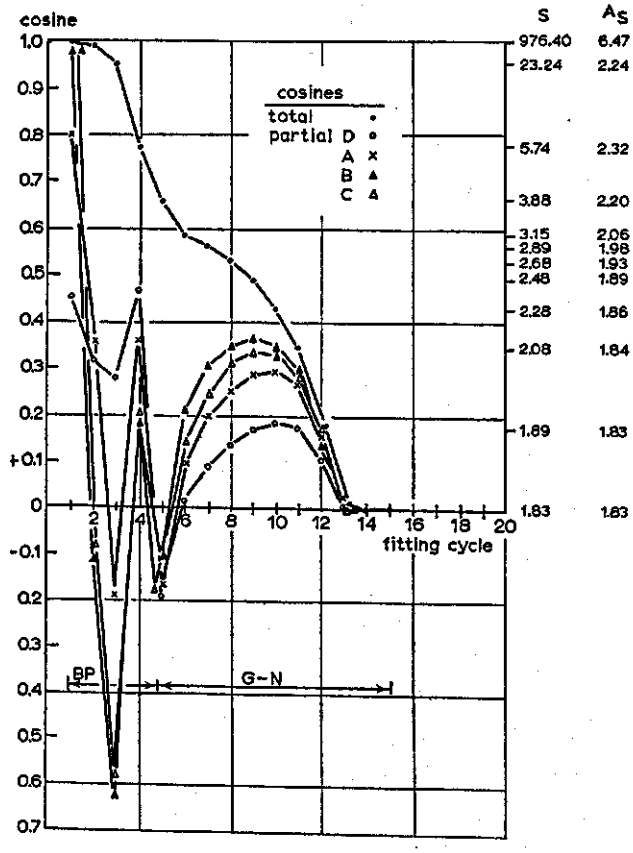


Fig. 32. Relationship between total cosine and partial cosines for the parameters D, A, B and C in the condition function of Appendix 2.3, and number of fitting cycle. During cycles indicated by BP back projection has been applied (circular search with metric I, see Fig. 25); during cycles indicated by G–N modified Gauss-Newton has been applied showing an increase of the partial cosines before they tend to zero. Sums of squares, S and ${}^4S$ respectively, for each fitting cycle are mentioned in the margin on the level of the ordinate values for the total cosine.

7.5 Subroutine control

The subprogram D2FDA for the evaluation of second derivatives is an example of a subprogram that needs not necessarily be called. Results obtained from D2FDA, however, are used in other subprograms and the corresponding statements must be by-passed in case D2FDA was not called by the main program NLV. This is governed by the subscripted variable NRINSUB, the subroutine index that counts the number of entries in each subprogram. The first statement of subprogram INITL is to set the subroutine counters to zero. The first statement of the Ith subprogram reads NRINSUB(I) = NRINSUB(I) + 1. Since the subroutine index occurs in COMMON it is possible in every subprogram to test each number of entries. If no use is made of

subprogram D2FDA, for which I = 5, its counter remains zero. Second derivative application anywhere in the program can thus be skipped if NRINSUB(5) = 0 in appropriate IF-statements.

A further advantage of the use of subroutine indices is that initial values to variables can be set to zero in the subprograms themselves. This can be done during the first entry, which is recognized by an appropriate IF-statement on the subroutine index.

7.6 Program layout

The arguments of the main program name, NLV, depend on the way the data are stored. If data have to be read from the input file, the first statement in the main pro-

Table 5. List of subprograms and their entries.

Name of subprogram	Alternative entry	Main purpose is to
Part I Function subprograms		
1. INITL		– define initial values
	DEFAULT	– define default values
	HEADING	– produce headings
2. READ		– read data
	NEWZERO	– perform univariate direct search
3. FNCTN		– evaluate the condition function
4. DFDA		– evaluate first derivatives
	ORDER	– perform optimal Gauss-Newton with parameter vectors partitioned according to ordered partial cosines
5. D2FDA		– evaluate second derivatives
Part II Gradient subprograms		
6. NRMEQ		– produce the gradient vector
		– produce normal equations
7. SOLVE		– solve normal equations
8. HOWA		– investigate the fitting result at point A, $(\theta^{(a)})$
	12. SUMRY	– summarize results
	COMBIN	– produce combinatorial search
9. MIN		– find the subminimum (point B, $\theta^{*(a)}$) in the direction of search
10. AISB		– replace A by B
11. LISTING		– produce an output list of intermediate and final results
	13. PLOT	– plot sequential functions on the line printer
	PUNCH	– punch (or print) intermediate parameter values
Part III Particular subprograms		
14. BLOCK		– store parameter groupings
15. BACK	B1, B2	– perform back projection
16. TRACK		– perform procedures of controlled approach
17. LIHYPEX		– apply parameter value extrapolation formulas

gram reads

```
PROGRAM NLV(INPUT,OUTPUT,TAPE2=INPUT,TAPE3=OUTPUT,
                                                TAPE7)
```

If data are written onto a file MYDATA the first statement reads

```
PROGRAM NLV(MYDATA,OUTPUT,TAPE2=MYDATA,TAPE3=
                                        OUTPUT,TAPE7)
```

The use of the arguments of the subprogram names are explained in Chapter 13. The subprograms are listed in Table 5 according to the subdivision into three parts as explained in Sections 7.2 and 7.3. The main purpose of each subprogram is mentioned as well. In Appendix 1.5 the modifications of the main program NLV that call subprograms from the third part, as well as those that call alternative entries, can be found.

8 Grouping of parameters

8.1 General

For several reasons it is useful to create the possibility to change the parameter vector.

The condition function can be extended for instance to a more general form. Then the research worker can decide to leave out of consideration particular parts of the function by keeping constant one or more of the parameters by giving them a trivial value, for example zero. The choice between sequential and nonsequential search (cf. Section 4.8) is another example where one might decide to keep constant one or more of the parameters in the condition function. One can also decide to partition the components of the parameter vector in two groups: the first consisting of those that are nonlinear in the condition function, the second consisting of those that are linear (Hayes, 1974).

Another reason to advocate the possibility to have the disposal of a system for solving only part of an optimization problem is, that if bounds on parameter values are exceeded, the parameter vector has to be broken down and its components have to be rearranged (Chapter 9).

Finally, in the first steps of the fitting procedure a group of parameters can be more efficient than the entire parameter vector. This means that fitting cycles then could be carried out with only $p_1 < p$ parameters and in a further stage either with the remaining $p - p_1$ parameters or the entire $p \times 1$ vector.

In these cases only that part of the solution of the normal equations that furnishes the required corrections needs attention and only those parameters that remain in the parameter vector as to be fitted parameters are subject to alterations.

It is obvious that it is not sufficient to develop a system with which the parameter vector can be truncated to a $p_1 \times 1$ vector. It is also necessary to permute the components of the parameter vector to generalize the procedure.

Permutation and truncation of the components of the parameter vector can be controlled by the user of the program. He has at his disposal parameter index cards in the main program deck, as described in Section 8.3.1. for permuting and grouping the parameter vector. Since this grouping takes place before execution of the main program, it is unconditional. Permuting and grouping also can be performed automatically during execution. This is unconditional in combinatorial search where all $2^p - 1$ combinations of the components of the parameter vector are subsequentially

used in a one-dimensional optimal search (entry COMBIN in subprogram HOWA, see Chapters 12 and 13). Conditional grouping is applied in subprogram MIN (Chapter 9) when bounds are violated. Finally, in entry ORDER in subprogram DFDA grouping is based on the values of the partial cosines in the vector c.

8.2 Partitioning the parameter vector

The back solution of the complete $p \times p$ system of normal equations gives the possibility to preserve partitioned solutions for 1, 1(1)2,..., 1(1)p_1,..., 1(1)p parameters, $p_1 < p$, in order of occurrence in the parameter vector. Solutions beyond the required first p_1 parameters can be useful. For example the length of the $p \times 1$ correction vector $\boldsymbol{d}$ tends toward zero. In case only p_1 parameters are applied this holds for the $p_1 \times 1$ vector $\boldsymbol{d}'$, say. However, one may wish to remain informed on the behaviour of the length of the $p \times 1$ vector $\boldsymbol{d}$ after each fitting cycle and so further solutions must then be available in an informative sense. Other examples are the partial cosines for those parameters that are not involved in a particular fitting cycle. As the coordinate system on the fitting surface generally is not an orthogonal one, so $\boldsymbol{J}^T\boldsymbol{J} \neq \boldsymbol{h}^D\boldsymbol{h}^D$, improvement of cosines with respect to parameters not used also may occur. It is of interest to remain informed on the values of these cosines as well.

From a point of view of efficiency this may not always be satisfactory. Suppose $p_1 << p$. In grouping, normal equations are evaluated for all p parameters, but only p_1 are employed in the following fitting cycle. When the advantage of the availability of informative parameters is not great, it is not efficient to calculate all products and cross products of the matrix $\boldsymbol{M} = \boldsymbol{J}^T\boldsymbol{J}$. To avoid time consuming unnecessary calculating and printing, it is advisable to reduce the optimization problem to a smaller

Table 6. Scheme of status in the program, denoted by x, of parameters after permutation.

Use	Status		
	to be fitted	informative	constant
Parameter vector	x	x	x
Evaluation of condition function	x	x	x
Evaluation of first derivatives	x	x	x
Evaluation of second derivatives	x	x	x
Normal equations	x	x	
Partial cosines	x	x	
Correction to parameters	x	x	
Length of normal vector	x	x	
Length of differential vector	x	x	
Application of corrections	x		
Curvature of parametric curves	x		
Differentials of scale factors	x		
Optimal step factor	x		

size with a $p_1 \times p_1$ matrix M. So parameters that are not involved in a particular fitting cycle then are taken constant for efficiency reasons.

These considerations lead to the concept of tripartitioning the parameter vector into to be fitted, informative and constant parameters, see Table 6.

Permutation and grouping are program options achieved by parameter index cards. As each parameter occurs subscripted in the condition function it must have a fixed place in the parameter vector. Hence permutation has to be carried out by indirect subscripting of the parameters. Grouping by the user is carried out by rearranging parameter index cards in the desired order in the main program deck. It then is automatically employed throughout all subroutines. Alternative groupings can be saved in subprogram BLOCK, up to a number of five. The use of this subprogram is explained in Chapter 13.

8.3 Application

8.3.1 *Main program NLV*

Permutation of parameters is a one-two-one mapping of the components of the original vector onto the components of the permuted vector. When permuting, each parameter obtaines a new subscript. The old and the new subscripts have to be kept in store to preserve the uniqueness of the permutation. This permutation is carried out by means of the parameter index cards.

After parameter values have been defined, the main program therefore continues with for example the following statements

```
K=0                                                          (0)
K=K+1      $ IP(K)=4                    $ JP(4)=K
K=K+1      $ IP(K)=2                    $ JP(2)=K
                          NPART=K                            (1)
K=K+1      $ IP(K)=1                    $ JP(1)=K
                          NPAR =K                            (2)
K=K+1      $ IP(K)=3                    $ JP(3)=K
                          MPAR =K                            (3)
```

In this case the order of the parameters θ_1, θ_2, θ_3 and θ_4 has been changed into θ_4, θ_2, θ_1, θ_3. The old subscripts are indicated by IP(K), the new subscripts by JP(k) on the parameter index cards. The maximum number of parameters p is denoted by MPAR. For to be fitted and informative parameters truncation is achieved by NPAR, this is the total number of parameters to be included in the normal equations. Finally NPART defines the first p_1 parameters that have to be fitted.

The order of cards that define the grouping must be (0), (1), (2), and (3) respectively, the first and last in fixed positions. The parameter index cards and the defining cards (1) and (2) – in that order – can be permuted, thus giving tripartitioning of the

parameter vector. The restrictions are

$$1 \leqq \text{NPART} \leqq \text{NPAR} \leqq \text{MPAR} = p$$

causing MPAR to be fixed at the value p.

8.3.2 *Subprogram FNCTN*

The subroutine subprogram that calculates the function values uses the parameters, called by the subscripted variable C in the argument, in the original order, so:

```
    DO 200 I=1,NDATA
200 YCLC(I)= function of X(I,J) and C(K)
```

Calculated values YCLC are obtained from the independent variables X(I,J) and the parameter values C(K). In the function the jth independent variable for $j = 1(1)m$, in FORTRAN J=1,NVEC, and the kth parameter for $k = 1(1)p$, or K=1, MPAR, take their own place and all parameters are needed. Therefore the components of the parameter C are not subject to permutation, neither to grouping.

Evaluation of the condition function is for all data $i = 1(1)v$, or I=1,NDATA, thus producing the vector

$$y = f(\theta)$$

according to (2.2.1).

8.3.3 *Subprogram DFDA*

In the general case the first derivatives with respect to the parameters are functions of all parameters. This means use of the unalteted parameter vector. However, in preparing the calculation of permuted normal equations, the vectors of first derivatives themselves have to be permuted. This can be done using the JP indirect subscript as follows (Appendix 1.4.1)

```
    K1=JP(1) $ K2=JP(2) $ K3=JP(3) $ K4=JP(4)
    DO 200 I=1,NDATA
    FA(I,K1)= function of X(I,J) and C(K)
    FA(I,K2)= ibid
    FA(I,K3)= ibid
200 FA(I,K4)= ibid
```

The functions at the right-hand sides denote the derivatives $f_1^{[i]}, f_2^{[i]}, f_3^{[i]}$ and $f_4^{[i]}$, respectively. The left-hand sides, evaluated for all data, produce for the example given in Section 8.3.1. the Jacobian

$$J = (f_4, f_2, f_1, f_3)$$

according to (2.2.5).

8.3.4 *Subprogram D2FDA*

The appropriate construction of permuted second derivatives is more complicated because cross derivatives occur. The total number of second derivatives equals $\binom{p+1}{2}$ or in FORTRAN MT*MPAR/2. In the example given in Section 8.3.1 this number is equal to 10. The pairs of subscripts of the unpermuted parameters are replaced by a single subscript KL, in our example according to

```
(K,L)= (1,1), (1,2), (1,3), (1,4); (2,2), (2,3), (2,4); (3,3), (3,4); (4,4)
KL   =  1  ,  2  ,  3  ,  4  ;  5  ,  6  ,  7  ;  8  ,  9  ; 10
```

The subscripts IDGL of the diagonal (K=L) are computed by

```
IDGL=(K-1)*(2*MPAR-K)/2+K for all K
```

However, when permuting, K and L have to be replaced by the new subscripts JP(K) and JP(L), for shortness denoted by KV and LV respectively. In those cases where KV > LV the subscripts are exchanged before calculating their single subscript KL. This part of the program reads (Appendix 1.3 and 1.4.1)

```
      KL=0
      DO 4 K=1,MPAR $ DO 4 L=K,MPAR $ KL=KL+1
      KV=JP(K)       $ LV=JP(L)
      IF (KV.LE.LV) GO TO 5
      KRES=KV $ KV=LV $ LV=KRES
    5 LV=LV-KV
    4 IL(KL)=(KV-1)*(2*MPAR-KV)/2+KV+LV

      K1=IL(1) $ K2=IL(2) $ K3=IL(3) $ K4=IL(4)
      K5=IL(5) $ K6=IL(6) $ K7=IL(7) $ K8=IL(8)
      K9=IL(9) $ K10=IL(10)

      DO 200 I=1,NDATA
      FAA(I,K1)= function of X(I,J) and C(K)
      FAA(I,K2)= ibid
      .          .
      .          .
      .          .
  200 FAA(I,K10)=ibid
```

The new single subscripts are stored in IL(KL). For the example given in Section 8.3.1, the relationship between pairs of permuted subscripts and the old and new single subscripts is

```
(KV,LV)= (4,4), (4,2), (4,1), (4,3); (2,2), (2,1), (2,3); (1,1), (1,3); (3,3)
KL     = 10  ,  7  ,  4  ,  9  ;  5  ,  2  ,  6  ;  1  ,  3  ;  8
IL(KL) =  1  ,  2  ,  3  ,  4  ;  5  ,  6  ,  7  ;  8  ,  9  ; 10
```

The indicated functions at the right-hand sides in the program text following the DO 200 statement denote for our example the second derivatives $f_{11}^{[i]}, f_{12}^{[i]}, \ldots, f_{44}^{[i]}$, respectively. The left-hand sides, evaluated for all data, produce the matrix J_2 as defined in (2.5.6) now with permuted column vectors.

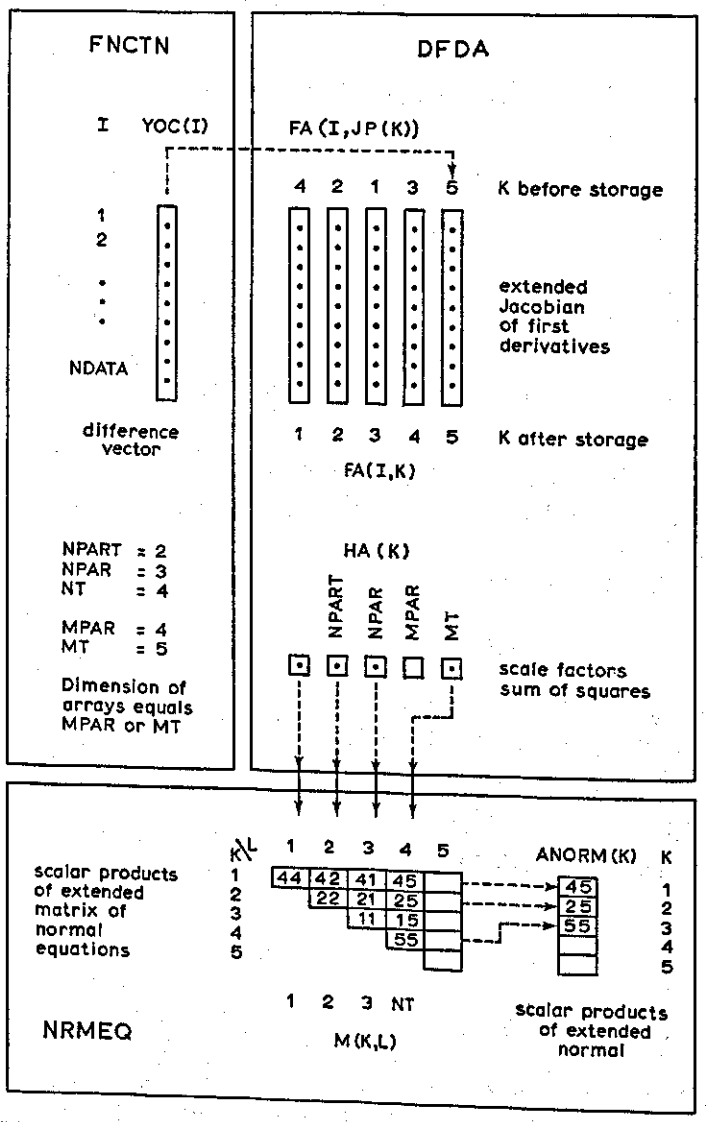


Fig. 33. Flow of first derivatives FA and of differences YOC through subroutines in the subprograms FNCTN, DFDA and NRMEQ. The Jacobian is extended with the difference vector. Scale factors are stored in HA, extended with the square root of the sum of squares stored in HA(MT). The normal and the sum of squares are stored in ANORM.

8.4 Consequences for first and second derivative flow

Permutation of the parameter vector has consequences for the flow of the derivatives through the subroutines.

The example of Section 8.3.1 is used in Fig. 33 which schematically shows the flow of the first derivatives and the difference vector through subroutines of subprograms FNCTN, DFDA and NRMEQ. The truncated normal is preserved in a vector ANORM extended with the sum of squares. The normal equations are solved for NPAR parameters of which the solution for NPART parameters is stored in the vector DELTA. The solutions for 1; 1,2; 1,2,3; ...; 1(1)NPAR parameters obtained in the subprogram SOLVE are saved in the extended matrix M together with the inverse matrix. This is illustrated in Fig. 34.

In subprogram MIN the solution stored in DELTA is applied to the relevant parameter values stored in A. In the equation for B the subscript K ranges from 1(1) NPART. The indirect subscripting causes the flow of the components of the parameter vector as given in Fig. 35. Parameters that play the role of informative parameters, in the present example θ_1, and constants, in the example θ_3, are not altered by differential corrections DELTA.

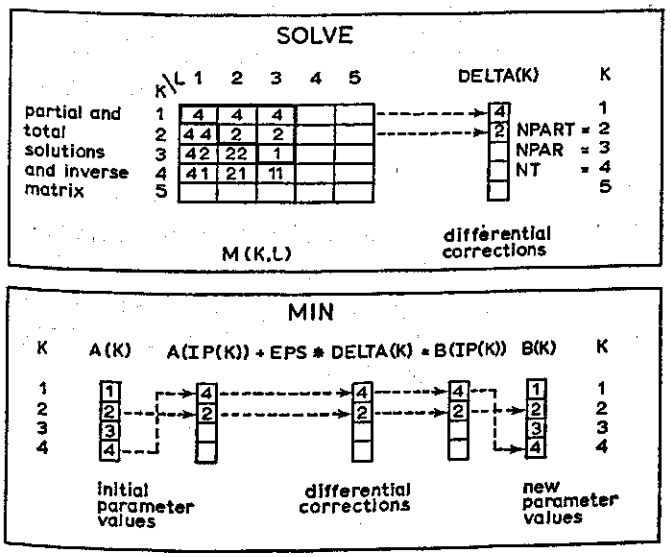


Fig. 34. Storage of solutions (upper triangle) and of the inverse matrix (lower triangle) for the first NPAR permuted parameters in the matrix M and the vector of differential corrections DELTA for the to be fitted NPART parameters.

Fig. 35. Flow of the components of the parameter vector to apply differential corrections DELTA to the parameter θ_4 and θ_2 stored in the initial parameter vector A. The new parameter vector is used in one-dimensional search. The step factor is denoted by EPS.

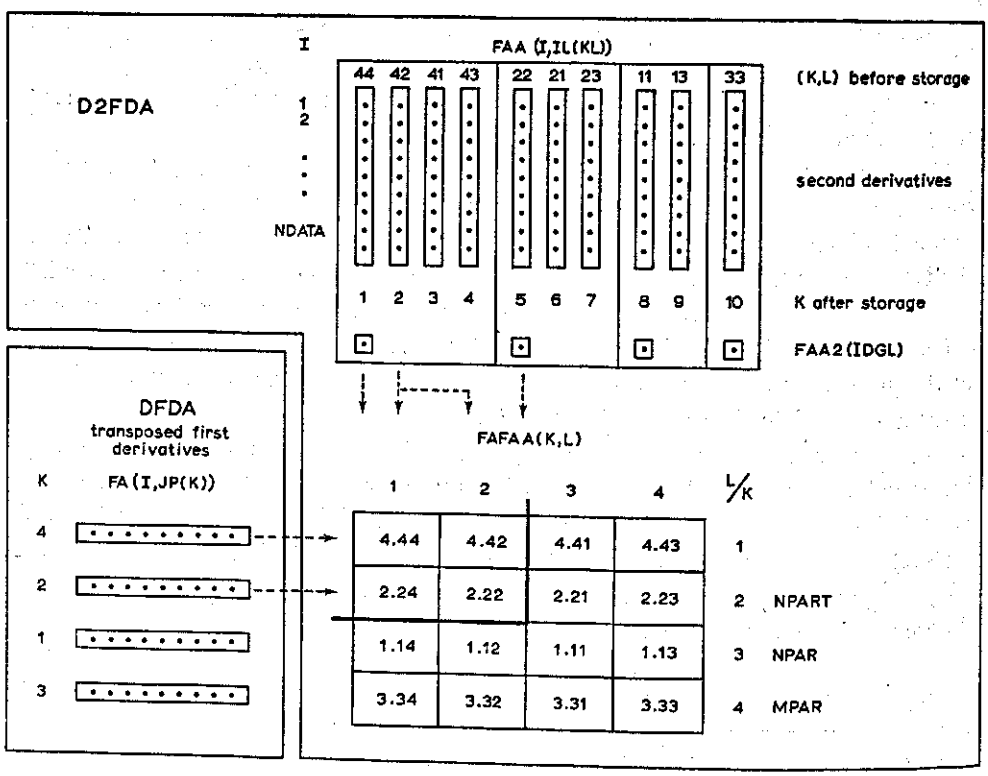


Fig. 36. Flow of first and second derivatives to obtain the matrix J_2, stored in FAA, the diagonal terms of the matrix M_{22}, stored in FAA2 and the matrix M_{12}, stored in FAFAA.

For the flow of second derivatives the cross second derivatives are stored in their new order of occurrence IL(KL) as illustrated by Fig. 36. To keep the possibility of using any number of parameters, all second derivatives are, for our example, stored in $\binom{5}{2}$ columns. From these columns scalar products are calculated to obtain the diagonal of the matrix M_{22} (equation 2.5.7) and the entire matrix M_{12} (equation 2.5.10). The latter is stored in the NPART × NPART matrix FAFAA.

9 Finding the minimum response in a given direction

9.1 General

When in a certain fitting cycle a direction of search s is adopted, the subminimum of the response S in this direction can be determined. The fitting problem thus is reduced to the determination of the optimal step length $\lambda^{*(n)}s^{(n)}$ in (3.5.1), which can be solved in the response subspace giving a subminimum (Fig. 5). The search then is called optimal search.

Methods to be applied depend to a certain extent on the progress already made during the preceding fitting procedure. In a neighbourhood of the minimum of S, where linear approximation to the fitting surface is accurate enough, the entire correction vector obtained in the Gauss-Newton algorithm can be added to the parameter vector (cf. Deming, 1948; Hayes, 1974). In this situation the efforts to achieve a better starting point by iteration procedures are superseded by the advantage of proceeding at once with further fitting cycles. Before a neighbourhood of the minimum response is reached, the optimum rate of progress in the new direction is to be determined to ensure real improvement of the actual parameter values.

In this chapter the parameter $\theta^{(n+1)}$ is considered a linear and strict function of the direction of search $s^{(n)}$.

9.2 Review of methods

In the response subspace (Fig. 5) we assume our problem solved for λ^* if

$$\left.\frac{dS(\theta(\lambda))}{d\lambda}\right|_{\lambda=\lambda^*} = 0 \tag{9.2.1}$$

which produces optimal search along $s^{(n)}$ in $\theta^{(n+1)} = \theta^{(n)} + \lambda s^{(n)}$.

In case the fitting surface is linear the response surface is quadratic (Section 2.6) and the $(\lambda, S(\lambda))$-diagram will plot a parabola in any direction s. This can be derived from (2.6.3) where for any constant vector s, $d^2S/d\lambda^2 = 2s^TJ^TJs > 0$. When the fitting surface is nonlinear but is approximated by a quadratic surface, the $(\lambda, S(\lambda))$-relationship is also represented by a parabola.

To find the subminimum, several methods have been suggested. Hartley (1961) calculates three values of the response, viz.

$$S(0),\ S(\tfrac{1}{2}),\ S(1) \tag{9.2.2}$$

The optimal value of λ where the approximating parabola through (9.2.2) attains its minimum then is given by the prediction

$$\lambda^* = \tfrac{1}{2} + \tfrac{1}{4}\frac{S(0) - S(1)}{S(0) - 2S(\tfrac{1}{2}) + S(1)} \tag{9.2.3}$$

It is to be checked whether $S(\lambda^*) < S(0)$ to be sure that an improvement is obtained.

A method where only two response values are needed was mentioned by Ruedenberg (Hartley, 1961). He uses $S(0)$, $S(1)$ and the slope $dS/d\lambda$ at $\lambda = 0$ which is given by (2.5.15) which gives $-2N^Ts$ and so the prediction is

$$\lambda^* = \frac{N^Ts}{S(1) - S(0) + 2N^Ts} \tag{9.2.4}$$

the vector N to be understood as evaluated at $\lambda = 0$. In this case a check is also required.

A method which tries to find the minimum by checking the results each time and that can be used where parabolic approximation is not allowed, is described by Booth and Peterson (1960). Their method (Fig. 37) works by halving or doubling the step factor λ. When new calculated values of the response $S(\lambda)$ no longer decrease, the last three values are used to find the minimum by approximation according to (9.2.3). The method principally is based on the following ideas. If $S(\frac{1}{2}) > S(1)$ both points are assumed to be located to the left of the minimum on the $(\lambda, S(\lambda))$-curve. If $S(\frac{1}{2}) < S(1)$ both points are assumed to be located to the right of the minimum. In case this is not true it will only be detected after many further calculations have already been made. In fitting practice the $(\lambda, S(\lambda))$-relationship is often found to be

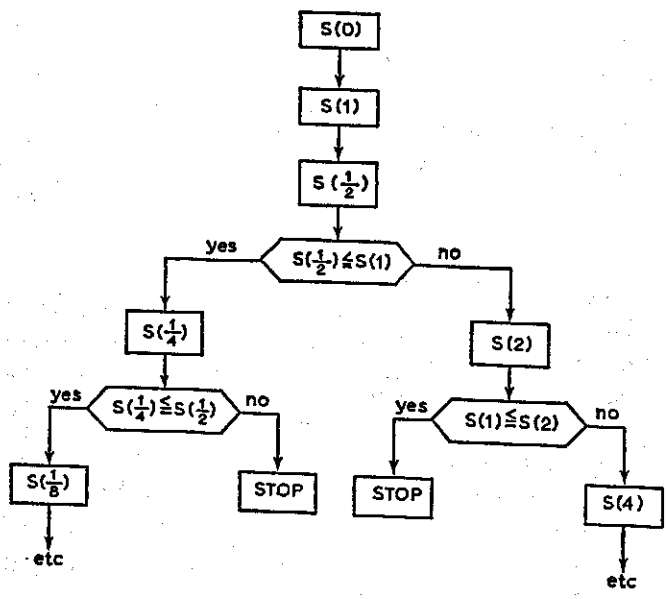


Fig. 37. Flowchart of minimization by halving or doubling the step factor (after Booth & Peterson, 1960).

highly asymmetric, so here also it is to be tested whether the value $S(\lambda^*)$ actually is smaller than any of the other values of the response found thus far.

The methods described above have no stopping criterion that depends on the condition for a subminimum. The test whether the value $S(\lambda^*)$ is smaller than any of the other values found in the current direction, does not guarantee that the best value of the step factor is found. Techniques that for this reason use first and second derivatives of the response are discussed in Sections 9.3 and 9.4.

9.3 Use of first derivatives

9.3.1 *Root finding*

In finding a subminimum of the response, use can be made of the actual function applied in (2.4.1). We transform the problem into the search for a root of the first derivative of the $(\lambda, S(\lambda))$-curve.

The calculation of the first derivative of the condition function is then necessary, but the advantage of using it is the more complete information on the process of finding the subminimum and the possibility to apply a simple stopping criterion. The slope to the $(\lambda, S(\lambda))$-curve in any arbitrary point is given by (2.5.15) and for any linear path of search with parameter λ by

$$S'(\lambda) = \frac{d}{d\lambda} S(\theta^{(n)} + \lambda s^{(n)}) = -2[N(\lambda)]^T s^{(n)} \tag{9.3.1}$$

In the general case this is a nonlinear function of λ and it is to be solved by iterative methods. The solution λ^* must meet the condition

$$-2[N(\lambda^*)]^T s^{(n)} = 0 \tag{9.3.2}$$

The component of the gradient $g(\lambda) = -2N(\lambda)$ in the direction of an arbitrary unit vector is called the directional derivative, which is the rate of change of S in the direction of this vector. Let $s/\|s\|$ be such a unit vector, then the directional derivative along this direction is given by

$$[g(\lambda)]^T \frac{s}{\|s\|} \tag{9.3.3}$$

Let $\phi(\lambda)$ be the angle between the gradient and the direction of search s, then the scalar product (9.3.3) can be written $\|g(\lambda)\| \cdot 1 \cdot \cos[\phi(\theta^{(n)} + \lambda s^{(n)})]$. This expression takes its maximum for constant λ when $\phi = 0$, so by taking s in the direction of the gradient itself. The greatest rate of change therefore is found in the direction of g and, consequently, the steepest descent takes place along N, cf. (2.4.10). The descent in the direction of s is zero if the directional derivative vanishes, so if for constant s the argument of the cosine becomes $\phi(\theta^{(n)} + \lambda^* s^{(n)}) = \pi/2$ and $g(\lambda^*)$ and $s^{(n)}$ are perpendicular (cf. Section 2.5 and equation (2.5.15)). This may be called optimal

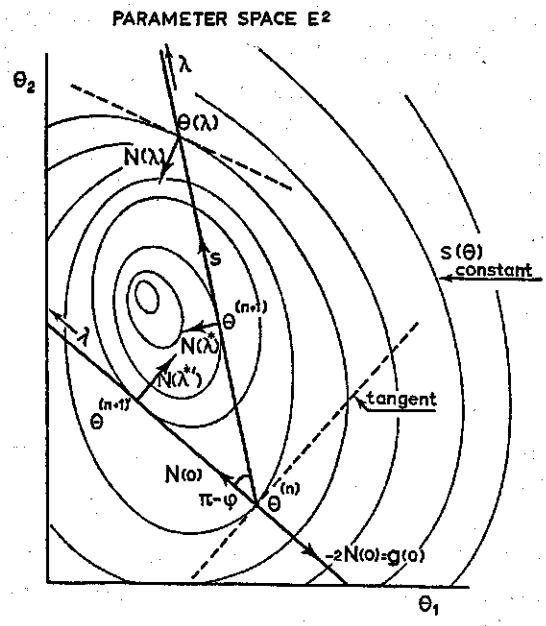


Fig. 38. Path of search in the parameter space E^2 for the subminimum of the response S in an arbitrary direction s and in the direction of the steepest descent $N(0)$. The solution is obtained when the gradient at $\theta^{(n+1)}$ is perpendicular to s or the gradient at $\theta^{(n+1)\prime}$ is perpendicular to $N(0)$.

gradient search (cf. Spang III, 1962). In Fig. 38 this is the case at $\theta^{(n+1)}$ in the direction of s, and at $\theta^{(n+1)\prime}$ in the direction of the steepest descent at $\theta^{(n)}$ taking $s = N(0)$.

Equation (9.3.1) can also be expressed as a cosine, viz.

$$\cos\phi(\lambda) = \frac{[g(\lambda)]^T}{\|[g(\lambda)]^T\|} \cdot \frac{s}{\|s\|}, \qquad -1 \leqq \cos\phi \leqq +1 \tag{9.3.4}$$

The minimum of S along s is obtained by setting (9.3.1), (9.3.3) or (9.3.4) equal to zero. These equations are spoken of as measures of slope. They all give the same solution for λ. The measures of slope are indicated by numerical, directional and cosine tangent respectively. The choice is governed by the calling program NLV, by means of the argument of the subprogram name MIN.

9.3.2 *One-dimensional linear search*

The 'Regula Falsi' (see e.g. Stanton, 1961 and Fig. 39 for factor $R = 1$) can be applied succesfully when two points are known that enclose the desired solution. To obtain a pair of points with this property, the following procedure is employed in subprogram MIN.

The first point of the $(\lambda, S(\lambda))$-curve, when $\lambda = 0$, has a negative slope if $s = N$ or $s = d$, because (9.3.1) and (2.4.13) produce $-2N^TN$ and $-2N^TM^{-1}N$ which are always negative (cf. Kowalik & Osborne, 1968). A starting value $\lambda = \lambda^{(0)}$ is applied. In case $ds/d\lambda$ at $\lambda = \lambda^{(0)}$ is positive, the location of the root is already enclosed. If $S'(\lambda^{(0)})$ is still negative a step factor $\lambda = 2\lambda^{(0)}$ is applied. The step factor is doubled until the first positive value of the $(\lambda, S'(\lambda))$-relationship is found. Now two

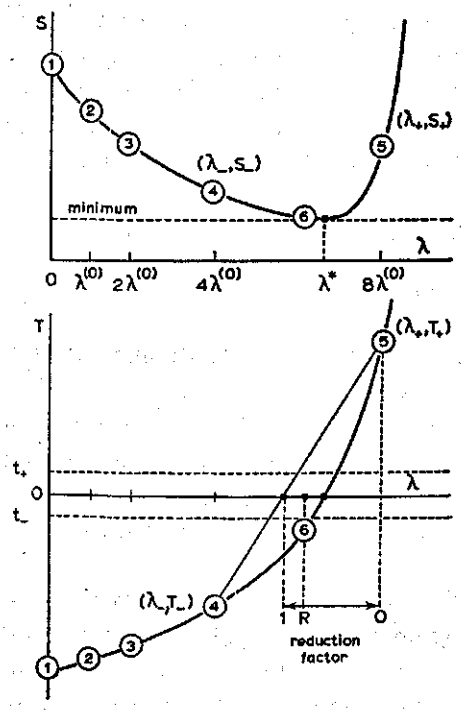


Fig. 39. Scheme of the order of calculation to enclose the minimum response S with the aid of a measure of slope T. Function evaluation is performed from point 1 through 5. The 6th point is obtained by using the factor R. Indicated by t_- and t_+ is the region for values of slope that satisfy the stopping criterion. With $R = 1$ the 'Regula Falsi' will be performed.

approximations to the root are available. In Fig. 39 this is the case in the fifth iteration step. The minimum is located somewhere in the interval (λ_-, λ_+), where λ_- = last found abscissa with $S' < 0$ (point 4) and λ_+ = first found abscissa with $S' > 0$ (point 5).

As the $(\lambda, S'(\lambda))$-curve may deviate appreciably from a straight line, a modified 'Regula Falsi' – illustrated by Fig. 39 – is an option in subprogram MIN. The arc between the points 4 and 5 is replaced by the chord. A new abscissa $\lambda \in (\lambda_-, \lambda_+)$ is found by the update formula

$$\lambda = \lambda_+ - RT_+ \frac{\lambda_+ - \lambda_-}{T_+ - T_-} \tag{9.3.5}$$

where

T= measure of slope according to (9.3.1), (9.3.3) or (9.3.4)
R= reduction factor $(0 < R \leqq 1)$
$-,+$ = subscripts denoting location with respect to the minimum at $\lambda = \lambda^*$

In further iteration steps cross partitioning of the chord is employed. If $T < 0$ equation (9.3.5) is used, if at the current point $T \geqq 0$ use is made of

$$\lambda = \lambda_- - RT_- \frac{\lambda_+ - \lambda_-}{T_+ - T_-} \tag{9.3.6}$$

Fig. 39 illustrates the advantage of using the reduction factor R in a special case. However, as the shape of the $(\lambda, S'(\lambda))$-curve may be different in each special case, it should be investigated whether it makes sense to use other values than unity for R as a default value. Because of the application of cross partitioning, values of R greater than unity can cause unpredictable results in that the iteration takes place at one side of the minimum only.

Obviously the 'Regula Falsi' works with $R = 1$.

9.3.3 *Stopping criteria*

Use of first derivatives makes it possible to introduce adequate stopping criteria. In the first place it is required that the positive value $T(\lambda_+)$ be less than $t_+ := -\rho_+ T(0)$, in the second place that the negative value $T(\lambda_-)$ be greater than $t_- := \rho_- T(0)$, see Fig. 39. Values can be assigned to ρ_+ and ρ_- in the main program NLV, the default value is set to 0.1 (10 % of the slope at $\lambda = 0$). Higher values can be efficient in the first number of fitting cycles since the $(\lambda, S(\lambda))$-curve then often is needle-shaped.

Stopping values for the step factor λ and the absolute value of the numerical tangent T given by (9.3.1) are built in in subprogram MIN. If doubling the step factor results in $\lambda > 10$, the step factor is fixed at $\lambda = 10$ and a last iteration with this value is performed. When $|T| < 0.1 \times 10^{-10}$ no further iterations are carried out. In both cases the step factor that produces the least value for the sum of squares is chosen and a return to the main program NLV is made.

The total number of iteration steps permitted is governed by the argument of SUBROUTINE MIN when greater than the default value 3.

9.3.4 *Example*

For the condition function and the data mentioned in Appendix 2.2, the number of iteration steps is determined for various values of the process parameters when applying the modified Gauss-Newton algorithm. The entries in Table 7 are the stopping criteria ρ and the reduction factor R. The number of iteration steps is listed for the cycle that gives the first subminimum and for the one that gives the second subminimum.

The increase of the number of iterations necessary to fulfill a given accuracy decreases for high values of R. Numbers obtained at values of the reduction factor equal to 0.8, 0.9 and 0.95, are of the same order of magnitude as numbers found with the 'Regula Falsi' given in Table 8. Also in this case the number of iteration steps only decreases slighty when ρ takes smaller values. Values for the measure of slope denoted by numerical tangent and cosine tangent, as derived in (9.3.1) and (9.3.4), are headed by 1 and 2 respectively. The sum of squares S calculated for these cases do not differ much. The results obtained with the default algorithm ($R = 1$, $\rho_- = \rho_+ = 0.1$) are given in bold type. See also Table 9 where the number of iteration steps in further cycles are also mentioned.

Table 7. Number of iteration steps to obtain the subminimum according to the Gauss-Newton algorithm for various values of the reduction factor R and the measure of accuracy ρ, $\rho_- = \rho_+$, for condition function and data mentioned in Appendix 2.2.

Reduction factor R	First subminimum, ρ fraction of slope						Second subminimum, ρ fraction of slope					
	0.6	0.4	0.2	0.1	0.05	0.01	0.6	0.4	0.2	0.1	0.05	0.01
0.4	14	14	16	18	22	28	7	11	13	17	19	25
0.5	14	14	16	18	20	24	7	9	11	13	15	19
0.6	14	14	16	16	18	22	5	7	7	9	11	13
0.7	14	14	16	16	18	20	5	5	7	8	9	12
0.8	14	14	16	16	17	19	5	6	6	8	8	10
0.9	14	14	16	16	16	18	6	6	6	7	7	9
0.95	14	14	16	16	16	17	7	7	7	7	8	9
0.9999	14	14	19	19	19	19	14	14	14	14	14	14

Table 8. Number of iteration steps to obtain the subminimum according to the Gauss-Newton algorithm for the 'Regula Falsi' for various values of the measure of accuracy ρ, $\rho_- = \rho_+$, and for the numerical tangent (1) and the cosine tangent (2), for condition function and data mentioned in Appendix 2.2.

ρ	First subminimum				Second subminimum			
	number of iterations		$S(\theta^{*(1)})$		number of iterations		$S(\theta^{*(2)})$	
	1	2	1	2	1	2	1	2
0.6	14	16	26.90	26.07	6	5	13.87	13.85
0.4	14	16	26.90	26.07	6	5	13.87	13.85
0.2	15	19	25.98	25.97	7	6	13.85	13.85
0.1	16	19	25.97	25.97	8	7	13.85	13.85
0.05	16	20	25.97	25.97	9	7	13.85	13.85
0.01	17	21	25.97	25.97	10	8	13.85	13.85

9.4 Use of second derivatives

9.4.1 *Root finding*

The 'Regula Falsi' convergence gives a linear approach to the root of a function. When certain conditions are met the Newton-Raphson algorithm can be used. It is faster because of a quadratic approach, since the number of correct decimals roughly doubles with each iteration step (see e.g. Stanton, 1961).

Application of the Newton-Raphson algorithm requires the evaluation of the derivative of the $(\lambda, S'(\lambda))$-relationship, so it requires the second derivative of the

response with respect to λ, given in (2.5.17) and (2.5.18). Consequently the method requires the evaluation of $(\frac{p+1}{2})$ second derivatives and so the advantage of using it for searching a subminimum is to be weighed against this disadvantage. Application can be justifiable when the number of parameters p and the number of data v is small.

If it is asked to find the solution of $f(x) = 0$, the Newton-Raphson iteration requires the evaluation of

$$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})} \tag{9.4.1}$$

where

$x^{(1)}$ = starting point of the iteration, near enough to the required root that the tangent to $f(x)$ at $(x^{(1)}, f(x^{(1)}))$ cuts the abscissa in a point nearer to the root than $x^{(1)}$,

$x^{(2)}$ = point of intersection which meets this condition.

Expressed in the symbols used in (9.3.1) and (2.5.17) the equation (9.4.1) becomes (cf. Davies & Whitting, 1972)

$$\lambda^{(2)} = \lambda^{(1)} + \frac{[N(\lambda^{(1)})]^T s}{s^T[M(\lambda^{(1)}) - N_{02}(\lambda^{(1)})]s} \tag{9.4.2}$$

As the direction of search s is chosen arbitrarily, (9.4.2) can be used for differential corrections d even if they are determined using first derivatives only, viz. $d = M^{-1}N$. The second term at the right-hand side of (9.4.2) is spoken of as the prediction formula for the step factor.

We restrict attention to a linear condition function. Then N_{02} vanishes and M does not depend on λ, so for any vector s in (2.5.17)

$$S''(\lambda) = 2s^T M s = \text{constant} \tag{9.4.3}$$

This expression is greater than zero for $s \neq 0$ (Section 2.6) which means that the $(\lambda, S(\lambda))$-curve has a minimum in the first quadrant because $S'(0) < 0$ and $S(\lambda) > 0$. Now (9.4.2) reduces to

$$\lambda^{(2)} = \lambda^{(1)} + \frac{[N(\lambda^{(1)})]^T s}{s^T M s} \tag{9.4.4}$$

Using differential corrections, so $s = d$, the denominator in (9.4.4) becomes $d^T M M^{-1} N(0)$ using (2.4.13) at $\lambda = 0$. From (9.4.3) and (9.3.1) it now follows that

$$S''(\lambda) = -S'(0) = \text{constant} \tag{9.4.5}$$

where $S'(0) < 0$. Equation (9.4.5) expresses that the slope of the straight line for the $(\lambda, S'(\lambda))$-relationship is equal to the negative value of its intercept on the ordinate. The equation of the line itself reads

$$S'(\lambda) = -(\lambda - 1)S'(0) \tag{9.4.6}$$

which has a root at $\lambda = 1$, giving a full step in the direction of search. For nonlinear condition functions this result may be used in near-linear situations so when $d^T N_{02} d$ now present in (9.4.2), is sufficiently small.

9.4.2 *Initial step factor*

Second derivatives can be used to predict the step factor in a certain fitting cycle (see Davies & Whitting, 1972; and for a discussion in detail for variable metric algorithms Dixon, 1972). This prediction value can be used as an initial value for λ to start the iteration process. Inserting $\lambda^{(1)} = 0$ in (9.4.2) gives

$$\lambda^{(2)} = \frac{[N(0)]^T s}{s^T[M(0) - N_{02}(0)]s} \tag{9.4.7}$$

The choice to use (9.4.7), only makes sense if the CALL D2FDA statement is present. Application then is possible by an appropriate modification of the main program NLV (Section 12.3).

Without using second derivatives a starting value $\lambda^{(1)}$ can be declared in the main program, the default value being 0.001. In further fitting cycles $\lambda^{(n+1)} = \lambda^{*(n)}$, $n = 1(1)t$ is automatically chosen as initial value in the next cycle.

In fitting practice the solution $\lambda^{(2)}$ obtained with second derivatives according to (9.4.2) can be compared with the solution by the iterative procedure to be described in Section 9.5.1 which gives the optimal value $\lambda^{*(1)}$. When in preliminary cycles both values differ appreciably, one should prefer $\lambda^{*(1)}$. When after several cycles it appears that $\lambda^{(n+1)}$ is a good prediction for $\lambda^{*(n)}$, one should prefer $\lambda^{(n+1)}$ without further iterations. A criterion could for example be $0.5\lambda^{*(n)} < \lambda^{(n+1)} < 1.5\lambda^{*(n)}$.

Since the routine is to enclose the root of any of the expressions (9.3.1), (9.3.3) or (9.3.4), in subprogram MIN, the employed initial value is automatically reduced. The default value for this reduction is $\lambda_r = \frac{2}{3}$. Assuming $\lambda^{*(n+1)} = \lambda^{*(n)}$, this reduction gives

$$\tfrac{2}{3}\lambda^{*(n)} < \lambda^{*(n+1)} < \tfrac{4}{3}\lambda^{*(n)} < 2\lambda^{*(n+1)} < \tfrac{8}{3}\lambda^{*(n)}$$

thus producing intervals symmetric about the assumed new optimal value and a value twice as large.

9.4.3 *Information on curvature*

Second derivatives can be used to obtain information on the curvature of the direction of search on the fitting surface at the initial point (Section 2.8). Before proceeding with the proper minimization routine, subprogram MIN starts with the determination of the geodesic curvature in the chosen direction of search.

9.4.4 *Example*

Initial values for the step factor obtained by the prediction formula (9.4.7) can be compared with optimal values obtained by the iteration process in subprogram MIN. This is done in Table 9 for the condition function and data given in Appendix 2.2 and Appendix 2.3, example 1 and 2 respectively, by consequent use of the modified Gauss-Newton algorithm. For each cycle the corresponding prediction value of the step factor was also calculated. It is observed that in example 1 from cycle 4 on, there exists a good agreement between both values. In example 2 the values deviate from cycle 17 through 21. These are the fitting cycles where the partial cosines start to approach zero as was illustrated in Fig. 31. In Table 9 the number of iterations to obtain the subminimum in the cycles mentioned was determined for the 'Regula Falsi' and default accuracy options viz. $\rho_- = \rho_+ = 0.1$. The first subminimum is obtained after 16 respectively 15 iterations, then the number of iterations is found to be 6 in almost all cycles. In example 2 this number slightly increases in cycles where the course of the partial cosines is irregular (Fig. 31).

The radius of geodesic curvature $(\kappa_g)^{-1}$ is listed in Table 10 as absolute values and values relative to $\sqrt{S}$ in each initial point of a fitting cycle. Fast progress is made in the cycles 18 through 22 where the radius of curvature is large and so the curvature in the direction of search is small. For a plot of the decrease of S in subsequent fitting

Table 9. A comparison between predicted step factors obtained with formula (9.4.7) and optimal step factors as obtained with the 'Regula Falsi' to determine the subminimum in subsequent fitting cycles for example 1 (Appendix 2.2) and example 2 (Appendix 2.3) when applying Gauss-Newton. The number of iteration steps in each cycle is also given.

Example 1				Example 2							
Cycle	Step factor		Number	Cycle	Step factor		Number	Cycle	Step factor		Number
	predicted	iterated			predicted	iterated			predicted	iterated	
1	0.0	0.0	0	1	0.0	0.0	0	13	0.0332	0.0333	6
2	0.8334	1.0753	16	2	0.3354	1.2043	15	14	0.0396	0.0397	6
3	0.9302	0.4885	8	3	0.0104	0.0103	7	15	0.0488	0.0491	5
4	0.9950	1.0094	7	4	0.0116	0.0117	6	16	0.0632	0.0641	5
5	1.0606	0.9161	6	5	0.0129	0.0130	6	17	0.0893	0.0918	6
6	0.9787	1.0482	6	6	0.0144	0.0144	6	18	0.1498	0.1615	7
7	1.0686	1.0659	6	7	0.0159	0.0160	6	19	0.3930	0.5458	8
				8	0.0177	0.0177	6	20	0.3765	0.7294	6
				9	0.0197	0.0197	6	21	0.5643	1.3269	7
				10	0.0221	0.0221	6	22	0.9255	1.0399	6
				11	0.0249	0.0250	6	23	0.9596	0.9808	6
				12	0.0285	0.0286	6	24	1.0278	1.0309	6
								25	1.1436	1.1416	6

Table 10. Values of quantities at the initial point of each fitting cycle for example 2, see Appendix 2.3.

Fitting cycle n	S	Total cosine	$\|\|N\|\|$	$\|\|d\|\|$	Radius of geodesic curvature in direction d	
					absolute	relative to $\sqrt{S}$
1	976.4	.9967	2966	6.1	27.39	0.88
2	327.2	.9808	2400	58.4	0.41	0.02
3	323.9	.9825	2267	52.3	0.45	0.03
4	320.3	.9843	2133	46.6	0.50	0.03
5	316.3	.9862	2000	41.6	0.56	0.03
6	311.8	.9882	1869	37.0	0.62	0.04
7	307.0	.9902	1742	32.8	0.69	0.04
8	301.6	.9922	1617	29.0	0.78	0.04
9	295.7	.9940	1445	25.6	0.88	0.05
10	289.3	.9956	1376	22.4	1.01	0.06
11	282.1	.9970	1261	19.5	1.17	0.07
12	274.0	.9980	1148	16.8	1.40	0.08
13	264.9	.9986	1037	14.3	1.72	0.11
14	254.4	.9988	927	11.9	2.20	0.14
15	241.9	.9986	818	9.4	3.03	0.19
16	226.4	.9979	707	6.9	4.67	0.31
17	205.6	.9966	592	4.1	9.14	0.64
18	172.9	.9944	464	2.1	25.54	1.94
19	87.5	.9869	273	4.2	16.33	1.74
20	38.1	.9751	162	0.9	29.69	4.81
21	4.9	.7926	78.5	0.12	31.21	14.08
22	1.8	.0338	0.71	0.048	0.34	0.25
23	1.8	.0028	0.13	0.0027	3.90	2.88
24	1.8	.0002	0.0026	0.0005	0.12	0.08
25	1.8	.0000	0.0007	0.0000	–	–

cycles see Fig. 25. It is observed that the steep decrease of sum of squares S from cycle 1 to cycle 2, occurs in an approximately linear direction of search with radius of curvature equal to 27.39 (Table 10).

9.5 Subprogram MIN

9.5.1 *Finding the optimal step factor*

Subprogram MIN starts with setting to zero the indices IREQN and IREQP that save the state of the measure of slope T relative to the stopping criteria t_- and t_+ (Fig. 40). An index ITO for the choice of the route through the program is set to 1. The first calculation of T is for $\lambda = 0$. A starting value $\lambda^{(0)}$ can be declared in the main program NLV, the default value being 0.001. For reasons explained in Section

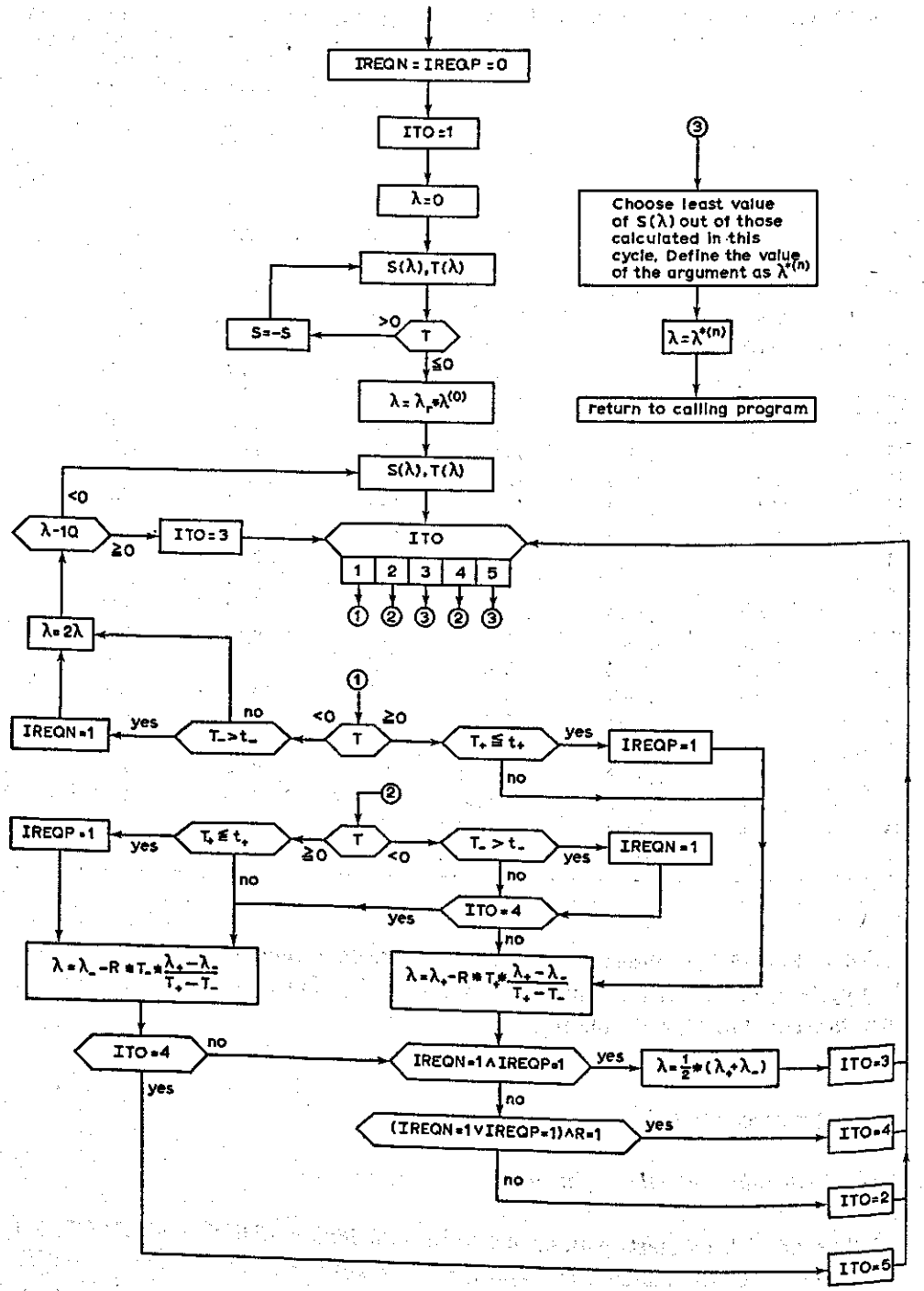


Fig. 40. Flowchart for main statements in subprogram MIN to find the subminimum of the response in the direction of search s.

9.4.2 a reduction is employed taking $\lambda = \lambda_r \lambda^{(0)}$ as the first step factor.

The iteration process consists of the following procedures (Fig. 40).

1st procedure With a given arbitrary correction vector s the sum of squares S and the desired measure of slope T to the $(\lambda, S(\lambda))$-curve at $\lambda = 0$ is calculated. If T happens to be positive, the sign of s is changed. Next, S is evaluated at $\lambda = \lambda_r \lambda^{(0)}$.

2nd procedure The step factor is doubled until a point is encountered where the slope is positive.

$R \neq 1$

3rd procedure The chord between the two points (λ_-, T_-) and (λ_+, T_+) last found is used to approximate the root of $S'(\lambda) = 0$, until both the last found negative value T_- and the last found positive value T_+ meet the required stopping criteria t_- and t_+ at λ_-^* and λ_+^* respectively.

4th procedure The mean value of λ_-^* and λ_+^* is employed as a possible value for λ^*, with which the last iteration step is performed.

$R = 1$, 'Regula Falsi'

3rd procedure The chord between the two points (λ_-, T_-) and (λ_+, T_+) last found is used to approximate the root of $S'(\lambda) = 0$, until either the last found negative value T_- or the last found positive value T_+ meets the required stopping criteria t_- and t_+ at λ_-^* and λ_+^* respectively.

4th procedure One further iteration is carried out, producing a possible value for λ^*.

$R \neq 1, R = 1$

5th procedure The value of λ which gives the least value for the response out of those calculated during the entire iteration process is finally employed as optimal step factor λ^*. The value $\lambda = \lambda^*$ is saved by means of the COMMON statements.

9.5.2 *Treatment of bounded parameters*

When doubling parameter values during the first stages of the iteration process to find the subminimum, predefined boundary values may be violated. In subprogram MIN this is detected by a separate routine. When boundary values are exceeded the routine iterates backwards in the direction of search to find a parameter vector whose terminal point is close to the boundary but inside the feasible region. The violating parameter is kept constant in further cycles, reducing the number of to be fitted parameters to $p - 1$, then grouping of parameters according to Section 8.2 automatically takes place.

When no boundary values are defined in the main program deck, this routine is bypassed.

9.5.3 *One-dimensional nonlinear search*

Subprogram MIN is suitable for linear and strict parameter functions.The corrections to parameter values can be obtained by adding the components of the vector of direction of search to the corresponding components of the parameter vector. In FORTRAN statements

```
    DO 12 K=1,NP $ IPK=IP(K)
12  B(IPK)=A(IPK)+EPS*CRCTN(K)
```

The direction of search is stored in CRCTN, and the step factor λ in EPS.

In nonlinear search formulas for the direction of search have to be derived by differentiating the parameter function with respect to the step factor as algorithm parameter, see (2.5.15). This is done in the subprograms BACK and LIHYPEX for the special methods developed in Chapter 6 and Chapter 10. In these subprograms the direction of the nonlinear search is calculated and stored in CORR, the parameter values are calculated and stored in B, both being subscripted variables that occur in COMMON. Control then bypasses the above mentioned DO-loop in subprogram MIN. CORR is assigned to CRCTN by means of the SUBROUTINE MIN argument, for further use in the subprogram.

10 Convergence of parameter values by extrapolation

10.1 General

Extrapolation of intermediate results can be employed at various stages of the fitting process. In these cases it is often the response surface or a response subsurface that is described by means of lower degree polynomials. Examples are the minimization in a direction of search by means of an approximating parabola and the approximation of the response surface by quadratic functions to assess the Hessian (Spendley, 1969; Murray, 1972b).

However, these techniques can also be used to simulate the convergence of parameter values. In many cases the path produced by the relationship between j and $\theta_k^{*(j)}$, $j = 1(1)n$, shows only little curvature. Still, linear extrapolation of the terminal points of parameter vectors obtained in preceding fitting cycles is not satisfactory, because in nonlinear optimization final values for each parameter separately will generally not be obtained with the same step factor. So other functions must be tried.

Values of $\theta_k^{*(j)}$, $j = 1(1)n$, can be plotted against the cycle number j or versus the cumulated values of $\lambda^{*(j)}$, giving ordered pairs

$$(j, \theta_k^{*(j)}) \quad \text{and} \quad (\sum_{i=1}^{j} \lambda^{*(i)}, \theta_k^{*(j)})$$

In the first case parameter values in subsequent cycles are equidistant for $j = 1(1)n$, in the second case their location is assumed to depend on the step factor.

The constants occurring in the parameter function to be used for extrapolation have to be solved from results of previous fitting cycles. Although such constants have to be determined for each condition function parameter separately they are not subscripted in this chapter, leaving it understood that they are collected in arrays ranging $1(1)p$.

For convenience sake we define in this chapter the following auxiliary quantities, omitting the subscript k for an arbitrary component of θ

$$\theta_{ij} := \theta^{*(i)} - \theta^{*(j)}, \qquad \lambda_j := \sum_{i=1}^{j} \lambda^{*(i)}, \qquad \lambda_{ij} := \lambda_i - \lambda_j \tag{10.1.1}$$

In Sections 10.3 and 10.4 it is required that

$$\theta^{*(1)} \leqq \theta^{*(2)} \leqq \theta^{*(3)} \quad \text{or} \quad \theta^{*(1)} \geqq \theta^{*(2)} \geqq \theta^{*(3)} \tag{10.1.2}$$

and $\lambda_j \geqq 0, \qquad j = 1(1)n.$

10.2 Fourth degree extrapolation

Simulation of convergence with a fourth degree polynomial was based on five equidistant fitting cycle numbers. After rescaling the parameter function becomes with $\lambda = -4(1)0$

$$\theta(\lambda) = a + b(2 + \lambda) + c(2 + \lambda)^2 + d(2 + \lambda)^3 + e(2 + \lambda)^4$$

where the solution for the constants is given by

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \frac{1}{24} \begin{bmatrix} 0 & 0 & 24 & 0 & 0 \\ 2 & -16 & 0 & 16 & -2 \\ -1 & 16 & -30 & 16 & -1 \\ -2 & 4 & 0 & -4 & 2 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix} \begin{bmatrix} \theta^{*(1)} \\ \theta^{*(2)} \\ \theta^{*(3)} \\ \theta^{*(4)} \\ \theta^{*(5)} \end{bmatrix}$$

The direction of search in the parameter space for an arbitrary component of s is given by

$$s(\lambda) = \frac{d\theta(\lambda)}{d\lambda} = b + 2c(2 + \lambda) + 3d(2 + \lambda)^2 + 4e(2 + \lambda)^3$$

producing a nonlinear direction of search. The optimization of λ can be achieved with subprogram MIN, see Section 9.5.3. A trivial starting value is the equidistant unit $\lambda = 1$, which produces a prediction for $\theta^{*(6)}$.

10.3 Hyperbolic extrapolation

In hyperbolic simulation (Fig. 41) it is assumed that $\theta^{*(J)}$ tends toward θ^* by means of a hyperbolic relationship. The general formula for each parameter then reads

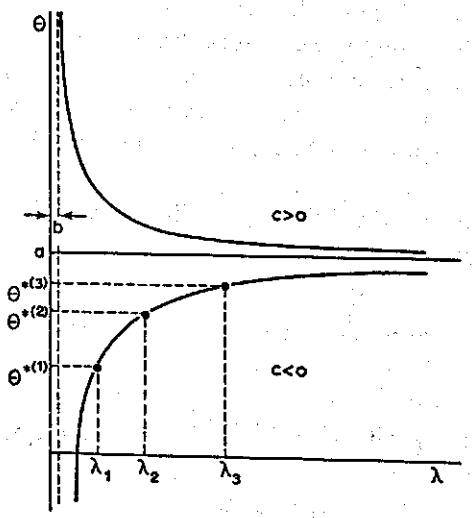


Fig. 41. Schematic illustration of using three intermediate fitting results for hyperbolic extrapolation based on parameter values and cumulative step factors.

$$\theta(\lambda) = \frac{c}{\lambda - b} + a, \qquad b < \lambda_1$$

where a is an approximation to θ^*. The three constants are solved from three not necessarily equidistant points. Simulation of convergence then is based on the points

$$(\lambda_1, \theta^{*(1)}), \qquad (\lambda_2, \theta^{*(2)}) \qquad \text{and} \qquad (\lambda_3, \theta^{*(3)}) \tag{10.3.1}$$

The first step to the solution of the constants is to eliminate a by taking the differences θ_{21} and θ_{32}. The next step is solving these differences for c. Equating gives

$$b = \frac{\lambda_{21}\theta_{32}\lambda_3 - \lambda_{32}\theta_{21}\lambda_1}{\lambda_{21}\theta_{32} - \lambda_{32}\theta_{21}}$$

so

$$c = \frac{-\theta_{32}}{\lambda_{32}}(\lambda_3 - b)(\lambda_2 - b)$$

and

$$a = \theta^{*(3)} - \frac{c}{\lambda_3 - b}$$

Finally we use, relative to the point last mentioned in (10.3.1)

$$\theta(\lambda) = a + \frac{c}{(\lambda_3 + \lambda) - b}, \qquad \lambda \geqq 0 \tag{10.3.2}$$

with initial step factor $\lambda = \lambda_3$ when entering subprogram MIN for the optimization of λ. Terminal points are $\theta(0) = \theta^{*(3)}$ and $\theta(\infty) = a$. The direction of search in the parameter space is for an arbitrary component of s given by

$$s(\lambda) = \frac{d\theta(\lambda)}{d\lambda} = \frac{-c}{(\lambda + \lambda_3 - b)^2}$$

which gives a nonlinear direction of search. The solution is restricted by a number of conditions that have to be checked for each parameter separately. Reference is made to (10.1.1) and (10.1.2).

The algorithm parameter λ is valid for the entire vector θ and so conditions on λ hold for all k. We observe that there is no solution if $\lambda_{32} = 0$. On the other hand if $\lambda_{21} = 0$ the solution reads $b = \lambda_1$, which inserted in the equation for c gives $c = 0$ and the hyperbola degenerates into its asymptotes. In this case extrapolation is to be employed by

$$a = 2\theta^{*(3)} - \theta^{*(2)}, \qquad b = 0, \qquad c = -\theta_{32}\lambda_3 \tag{10.3.3}$$

to be inserted in (10.3.2). Terminal points of search in this case are $\theta(0) = \theta^{*(3)}$ and $\theta(\infty) = \theta^{*(3)} + \theta_{32}$. All other cases have the property $\lambda_1 < \lambda_2 < \lambda_3$.

If for a condition function parameter it is found that $\theta_{32} = 0$, the solution is $b = \lambda_1$ and $c = 0$ and so $\theta(\lambda) = \theta^{*(3)}$ for any choice of λ in (10.3.2). Finally if $\theta_{21} = 0$ the solution given by (10.3.3) can be used. The particular parameter for which this solution holds is extrapolated linearly.

The conditions can be summarized with

$$|\theta_{21}/\lambda_{21}| > |\theta_{32}/\lambda_{32}| \tag{10.3.4}$$

which can easily be interpreted geometrically. It covers the situations $c > 0$ and $c < 0$ (Fig. 41).

10.4 Exponential extrapolation

In exponential simulation (Fig. 42) it is assumed that $\theta^{*(J)}$ tends toward θ^* by means of an exponential relationship. The general formula for each parameter reads

$$\theta(\lambda) = ce^{-b\lambda} + a$$

where the constants are solved from a set of points according to (10.3.1). The first steps are analogous to those described in the former section. Then we arrive at a nonlinear equation for b which reads

$$\frac{\theta_{32}}{\theta_{21}} = \frac{1 - \exp(-b\lambda_{32})}{\exp(b\lambda_{21}) - 1}$$

This can be written, introducing the auxiliary variables $t_1(b_1)$ and $t_2(b_2)$ requiring $t_1 = t_2$ for $b_1 = b_2$ and $b_1 \neq 0$

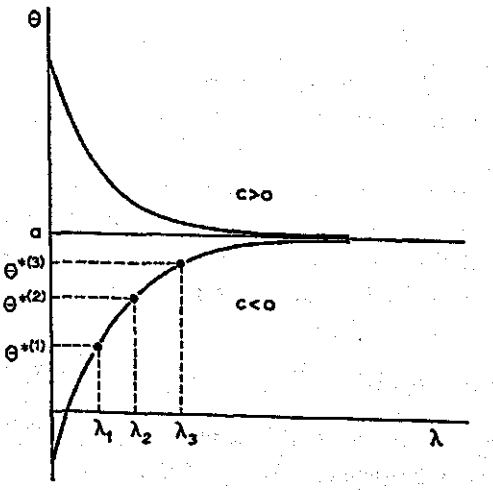


Fig. 42. Schematic illustration of using three intermediate fitting results for exponential extrapolation based on parameter values and cumulative step factors.

$$t_1(b_1) = \theta_{32}(e^{b_1\lambda_{21}} - 1), \qquad \left.\frac{dt}{db_1}\right|_{b_1=0} = \lambda_{21}\theta_{32} \tag{10.4.1}$$

$$t_2(b_2) = \theta_{21}(1 - e^{-b_2\lambda_{32}}), \qquad \left.\frac{dt}{db_2}\right|_{b_2=0} = \lambda_{32}\theta_{21} \tag{10.4.2}$$

which are monotone functions because of the conditions laid down in Section 10.1.

In (10.4.1) we have $t_1(0) = 0$, $t_1(\infty) \to \infty$; in (10.4.2) this is $t_2(0) = 0$, $t_2(\infty) = \theta_{21}$. The condition for a solution thus is $\lambda_{21}\theta_{32} < \lambda_{32}\theta_{21}$. The iteration procedure starts with the calculation of t_2 for a given value of b_2; equating $t_2 = t_1$ the equation (10.4.1) is solved for b_1 and the solution is put equal to b_2 and so on. Elimination of t gives the iteration formula

$$b_1 = \frac{\ln [\theta_{31} - \theta_{21} \exp(-b_2\lambda_{32})] - \ln \theta_{32}}{\lambda_{21}} \tag{10.4.3}$$

The first iteration step can conveniently be performed with $b_2 = \infty$. Back solution of c and a yield

$$c = \frac{\theta_{32}}{e^{-b\lambda_3} - e^{-b\lambda_2}}$$

and

$$a = \theta^{*(3)} - ce^{-b\lambda_3}$$

Parameter values relative to the point last mentioned in (10.3.1) now can be obtained from

$$\theta(\lambda) = a + ce^{-b(\lambda_3+\lambda)}, \qquad \lambda \geqq 0 \tag{10.4.4}$$

with initial step factor $\lambda = \lambda_3$ when entering subprogram MIN for the optimization of λ. Terminal points are $\theta(0) = \theta^{*(3)}$ and $\theta(\infty) = a$. The direction of search in the parameter space is for an arbitrary component of s given by

$$s(\lambda) = \frac{d\theta(\lambda)}{d\lambda} = -bce^{-b(\lambda_3+\lambda)}$$

giving a nonlinear direction of search.

This solution also is restricted by a number of conditions, analogous to those for hyperbolic extrapolation. Again we have no solution if $\lambda_{32} = 0$. Further, $\lambda_{21} = 0$ produces $b \to \infty$ and the solution preferred under this condition reads

$$a = 2\theta^{*(3)} - \theta^{*(2)}, \qquad b = 1, \qquad c = -\theta_{32}e^{\lambda_3} \tag{10.4.5}$$

to be inserted in (10.4.4). Terminal points of search are in this case $\theta(0) = \theta^{*(3)}$ and

$\theta(\infty) = \theta^{*(3)} + \theta_{32}$. For decreasing values of θ slopes in (10.4.1) and (10.4.2) are negative. The condition for a solution then is $\lambda_{21}\theta_{32} > \lambda_{32}\theta_{21}$.

If for a condition function parameter it is found that $\theta_{32} = 0$ the solution for any choice of λ in (10.4.4) gives $\theta(\lambda) = \theta^{*(3)}$. Finally $\theta_{21} = 0$ requires a solution by (10.4.5). The particular parameter for which this solution holds is extrapolated linearly.

The conditions can be summarized with (10.3.4) covering the situations $c > 0$ and $c < 0$ (Fig. 42).

10.5 Program options

The extrapolation procedures are programmed in subprogram LIHYPEX. Specific options are governed by the appropriate arguments of the subroutine and the calling programs. The statement reads

SUBROUTINE LIHYPEX(LHE,N1,N2,N3,NCRIT,LIST)

where the following choices can be made

LHE = 1, fourth degree polynomial extrapolation.
= 2, hyperbolic extrapolation.
= 3, exponential extrapolation.

N1,N2,N3, fitting cycle numbers that define the intermediate optimal points to be used. If LHE = 1 five equidistant points are chosen, the first and second being given by N1 and N2.

NCRIT, critical number of significant figures to predetermine the required accuracy of the iteration process in (10.4.3). A suitable value is NCRIT=3. The number of iterations for each parameter is confined by a default value 25 defined in the subroutine.

LIST = 1, a list of intermediate results will be produced.
= 0, listing is suppressed.

In subprogram LIHYPEX it is tested whether the number of cycles is sufficiently large to perform the extrapolation with the points required by N1, N2 and N3. The subprogram can be called by a modification of the main program NLV and then is automatically called by subprogram MIN. In the last case the statement reads CALL LIHYPEX (LPX + 10, 0, 0, 0, 0, 0) and it is used to jump to the calculation of $\theta(\lambda)$ and $s(\lambda)$ only, for optimizing λ. In subprogram MIN the DO-loop that calculates parameter values θ is bypassed, as explained in Section 9.5.3.

10.6 Example

For the condition function and data given in Appendix 2.3 exponential extrapolation was applied. Intermediate parameter vectors chosen to be used for extrapolation were $\theta^{*(2)}$, $\theta^{*(4)}$ and $\theta^{*(6)}$, obtained with the modified Gauss-Newton algorithm.

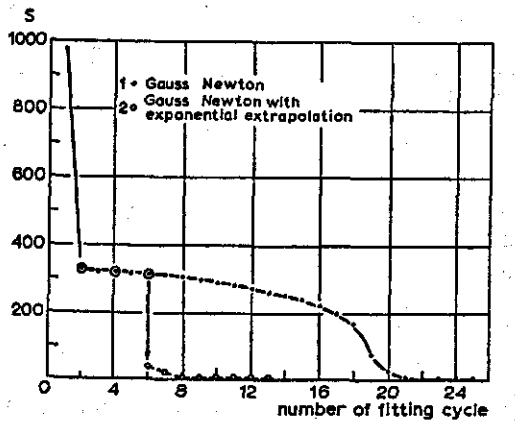


Fig. 43. Decrease of sum of squares for the problem in Appendix 2.3 according to two algorithms. 1 modified Gauss-Newton; 2 modified Gauss-Newton with exponential extrapolation of parameter values obtained in the 2nd, 4th and 6th cycle. See also Figs. 19 and 25.

For each parameter, exponential approximation was applied. The optimization of the algorithm parameter λ gives a decrease of the sum of squares from 311.8 to 40.3 in the 6th fitting cycle which is illustrated in Fig. 43. The modified Gauss-Newton algorithm needs 19 further cycles, but after exponential extrapolation only 7 further cycles were needed.

Table 11. Parameter values obtained in the fitting cycles mentioned. In the upper part for the modified Gauss-Newton algorithm, in the lower part after in the 6th cycle exponential extrapolation was employed based on intermediate results in cycles 2, 4 and 6. For condition function and data see Appendix 2.3.

Fitting cycle n	λ_n	Parameters in condition function			
		D	A	B	C
Gauss-Newton					
1	0.0	38.40	1.3100	0.2746	3.489
2	1.204	37.94	0.8067	0.2117	10.772
4	1.226	38.10	0.8187	0.2365	9.570
6	1.254	38.31	0.8345	0.2674	8.385
10		38.93	0.8847	0.3529	6.167
15		40.31	1.0508	0.5392	3.722
20		37.81	2.2101	0.6662	2.117
25		38.31	2.1277	0.5474	3.047
Gauss-Newton and exponential					
1	0.0	38.40	1.3100	0.2746	3.489
2	1.204	37.94	0.8067	0.2117	10.772
4	1.226	38.10	0.8187	0.2365	9.570
6	1.254	38.31	0.8345	0.2674	8.385
6*	1.702	40.42	1.2617	0.6144	4.468
10		38.22	2.1315	0.5509	2.952
13		38.31	2.1277	0.5474	3.047

* Exponential extrapolation.

Table 12. Values of constants for exponential extrapolation. See Table 11.

Condition function parameter	Exponential extrapolation constants		
	a	*b*	*c*
D	44.145	1.0	–20.447
A	2.018	1.0	– 4.147
B	0.925	1.675	5.371
C	4.413	9.537	0.619×10^6

The solutions converge to the same parameter values, but it is as if a number of intermediate results is skipped when using extrapolation. A summary of results is given in Table 11. The upper part of the table gives a sequence of results obtained with the modified Gauss-Newton algorithm. The lower part gives the results obtained with this algorithm, with in the 6th cycle application of exponential extrapolation.

In Table 12 the constants used in exponential extrapolation are given for the parameters of the condition function.

11 Controlled approach by intermediate observation vectors

11.1 General

In parameter optimization the path on the fitting surface along which convergence is achieved depends on

- the choice of the initial value of the condition function parameters,
- the algorithm applied,
- the strategy on step length.

Appropriate choices can be made by the user of an optimization program, but by this no conditions are set to the path of search to be followed. It might be of interest to the research worker, however, to know that the minimum response was reached without heavy oscillation of the intermediate solutions in undesirable directions (see Section 1.1).

In this chapter a method is developed that gives a controlled approach to θ^* starting at $\theta^{(0)}$ by gradually increasing the distance to the fitting surface from zero to the value at the terminal point of the observation vector. Although the procedure does not give an acceleration of the convergence process, the advantage is found in that it reduces the relative curvature of the fitting surface and the parametric curves (Section 2.8) and produces on the fitting surface a locus of points that are closest to the terminal points of intermediate observation vectors.

11.2 Procedure

11.2.1 *Reduction of the original difference vector*

Starting from the initial approximation $\theta^{(0)}$ intermediate observation vectors $x^{(i)}$ can be found with

$$x^{(0)} := f(\theta^{(0)})$$

$$x^{(i)} := x^{(i-1)} + \beta(x - x^{(i-1)}) \tag{11.2.1}$$

$$\beta = \frac{1}{N + 1 - i}, \qquad i = 1(1)N, \qquad N < \infty$$

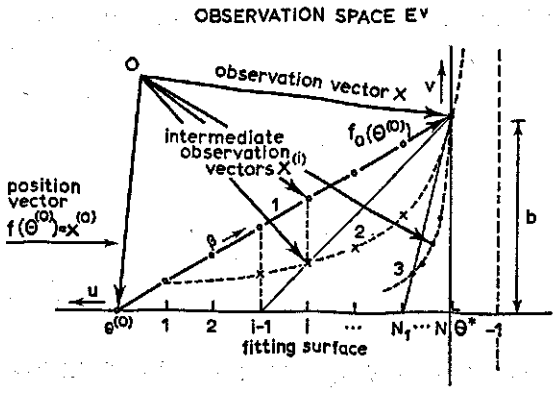


Fig. 44. Schematic illustration of methods of controlled approach by intermediate observation vectors whose terminal points describe loci in the observation space. 1 locus when using the original difference vector $f_0(\theta^{(0)})$ only; 2 locus when using subsequent difference vectors $f_0(\theta^{(i)})$, $i=1(1)N$; 3 when using $f_0(\theta^{(N1)})$ to start a new procedure 2 to proceed from N_1 in again N steps for better control.

when an approach to $x = x^{(N)}$ in N steps is employed. The locus of the terminal points of $x^{(i)}$ is illustrated in Fig. 44 by line 1. In this linear case the locus is the original difference vector $f_0(\theta^{(0)}) = x - f(\theta^{(0)})$. For the construction of these vectors $x^{(i)}$ no further points on the fitting surface are needed.

Intermediate observation vectors are obtained by adding to the position vector at $\theta^{(0)}$ the original difference vector reduced with a factor β. The orthogonal projection of their terminal points on the fitting surface are determined, giving controlled approach to θ^*.

11.2.2 *Reduction of sequential difference vectors*

The procedure can be repeated when as a result of fitting on $x^{(1)}$ the optimal parameter has be found, now using $\theta^{(1)}$ as the starting point for a new difference vector. This results in a slower progress to the terminal point of the observation vector x in the first stages (curve 2 in Fig. 44). Intermediate observation vectors in this case are obtained by

$$x^{(0)} := f(\theta^{(0)})$$

$$x^{(i)} := f(\theta^{(i-1)}) + \beta[x - f(\theta^{(i-1)})] \tag{11.2.2}$$

$$\beta = \frac{1}{N+1-i}, \qquad i = 1(1)N, \qquad N < \infty \tag{11.2.3}$$

where $\theta^{(i-1)}$ can be chosen to denote $\theta^{*(i-1)}$. If the fitting surface is approximated by a straight line (Fig. 44), the terminal points of the intermediate observation vectors are obtained by the intersection of lines, expressed in a (u,v)-reference system,

$$u = N - i$$

and

$$v = \frac{-bu}{N + 1 - i} + b$$

where $b^2 = S(\theta^*)$ and $i = 1(1)N$. Elimination of i yields

$$v(u + 1) = b$$

Thus the locus of the terminal points of $x^{(i)}$ is a hyperbola with asymptotes $u = -1$ and $v = 0$. Also here the orthogonal projection of the terminal points on the fitting surface gives controlled approach to θ^*.

11.3 Application

Since each step in the controlled approach method is subject to a fitting procedure to find the next value of θ, available optimization algorithms can be used. To obtain the intermediate results weaker stopping criteria than the default values (Section 13.4) could be employed. On the other hand, to ensure that the path on the fitting surface really is controlled, criteria that are too weak are in contradiction with the purpose of the method. The disadvantage of a relatively strong stopping criterion in the determination of intermediate results is partly compensated by increasing the number of steps on the fitting surface. For the chosen example the number of cycles in each step was found to decrease slightly in these situations (Section 11.4).

The factor β in (11.2.1) and (11.2.2) is defined such that the original observation vector x is reached in a predefined finite number of steps N. If required, from a certain step N_1 on progress to this vector can be done in more steps by replacing the original difference vector by $f_0(\theta^{(N_1)})$.
Such an approach is indicated in Fig. 44, curve 3, where at step N_1 the value of i in (11.2.3) is set to 1 again.

The method and its modifications are part of subprogram TRACK.

11.4 Example

For condition function, data and starting values mentioned in Appendix 2.2, controlled approach was applied using the method developed in Section 11.2.1. For each parameter separately the results are plotted in Fig. 45. The numbers refer to the the number of steps that the original difference vector f_0 was divided in. For $N = 1$ the full difference vector was used, the total number of fitting cycles using modified Gauss-Newton being 7. Then the value $N = 4$ was applied. The number of fitting cycles for each step then was 5, 4, 4 and 5 respectively. In each step the deviations from the ultimate trend are damped. For $N = 10$ only the terminal points $\theta^{(i)}$ per step are given. For $N = 50$ these points constitute the heavily drawn curve. In the latter case 3 fitting cycles per step were needed.

Although the intermediate parameter values, especially of the first and the last parameter, in controlled approach show the influence of the curvature of the para-

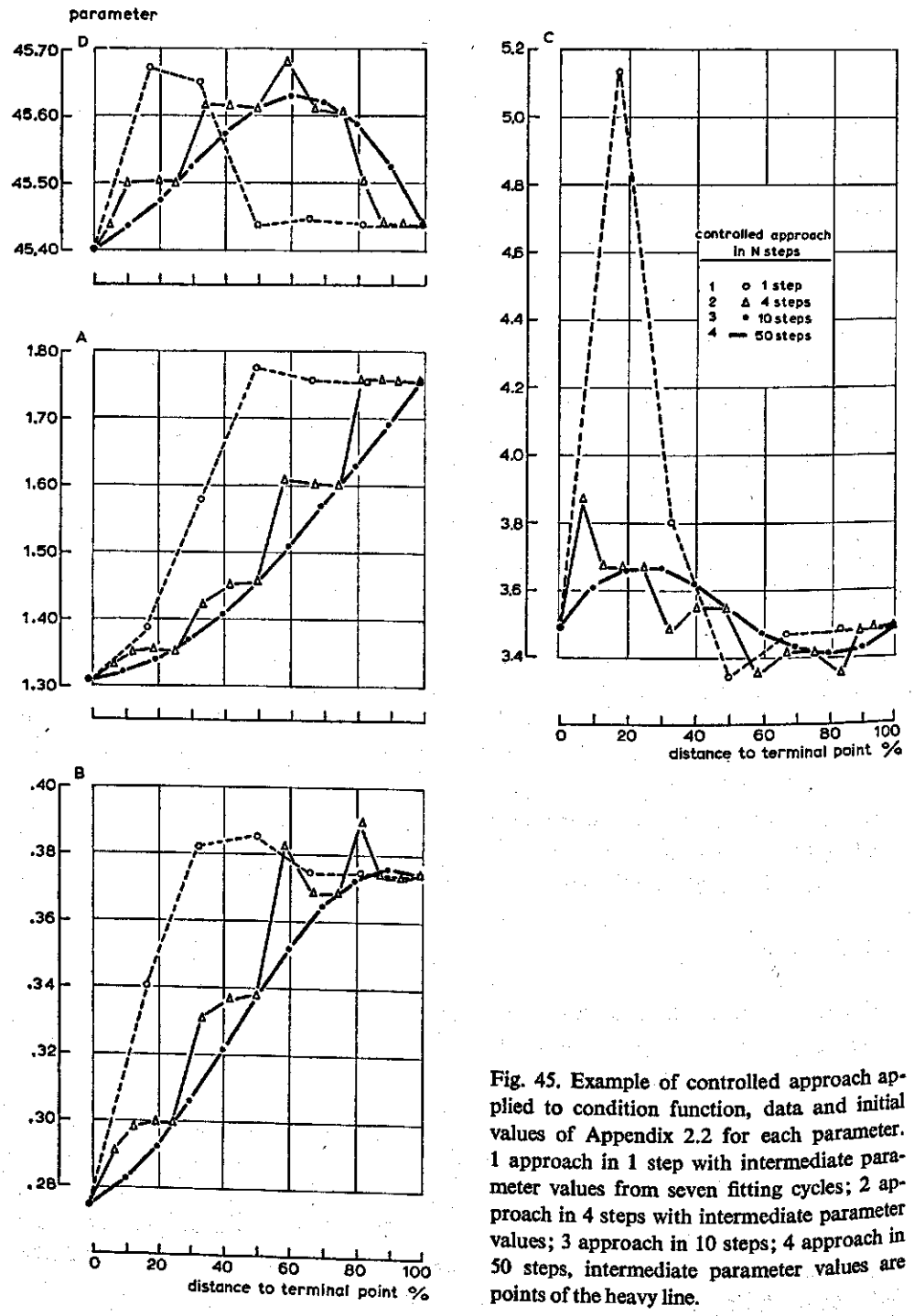


Fig. 45. Example of controlled approach applied to condition function, data and initial values of Appendix 2.2 for each parameter. 1 approach in 1 step with intermediate parameter values from seven fitting cycles; 2 approach in 4 steps with intermediate parameter values; 3 approach in 10 steps; 4 approach in 50 steps, intermediate parameter values are points of the heavy line.

metric curves, a cross section through the fitting surface gives no indication that this surface is curved along the path used in this controlled approach. This is illustrated in Fig. 46 that gives the calculated distance of the fitting surface to the intermediate points of the original difference vector for $N = 50$.

To gain a better insight a schematic view is given (Fig. 47) of parametric curves on

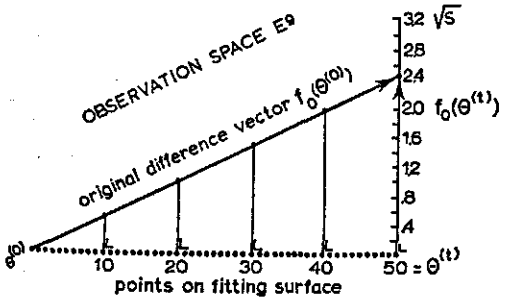


Fig. 46. Cross section through the fitting surface (heavy dots) obtained by plotting calculated distances from the original difference vector $f_0(\theta^{(0)})$ to the fitting surface, when applying equation (11.2.1) with $N = 50$ to condition function and data of Appendix 2.2. See also Fig. 45.

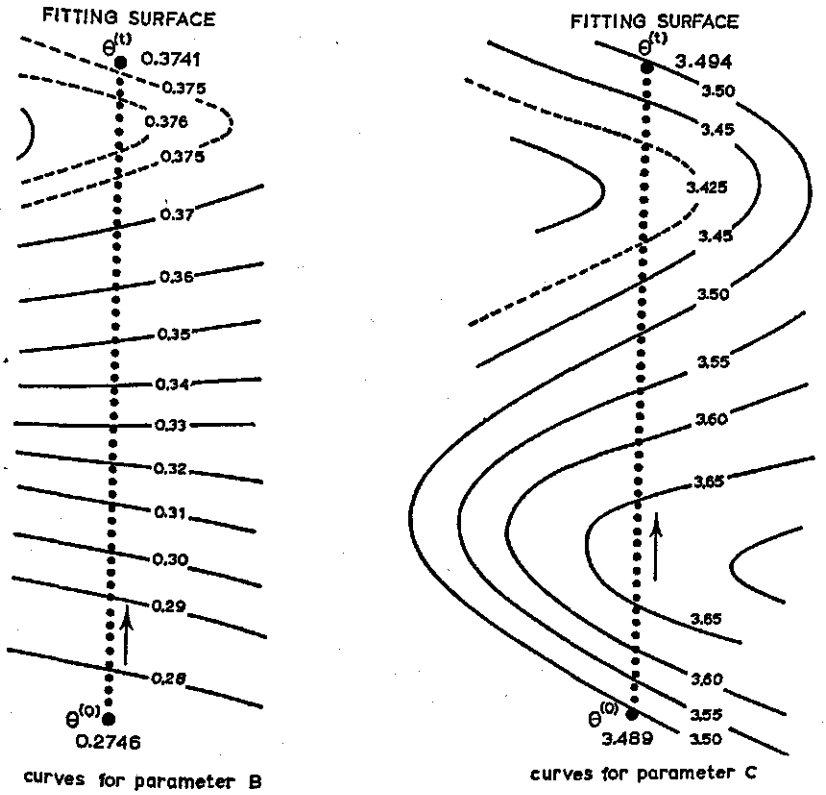


Fig. 47. Path of controlled approach on the fitting surface from $\theta^{(0)}$ to $\theta^{(t)}$, for two parameters, passing calculated intermediate parameter values as obtained by equation (11.2.1) with $N = 50$, for condition function and data of Appendix 2.2. Hypothetical parametric curves that could have produced these results are drawn. See also Figs. 45 and 46.

the fitting surface that could produce results analogous to those depicted in Figures 45 and 46. For two parameters the path from $\theta^{(0)}$ to $\theta^{(t)}$ is given when passing the parameter values that form the intermediate results for $N = 50$ in controlled approach.

III The program

12 Application

12.1 Testing the formulas for first and second derivatives

Formulas for first and second derivatives can be tested for programming errors by a comparison of calculated function values and results empirically obtained (Murray, 1972d). In order to test all derivatives the parameter index cards (Section 8.3.1) are arranged such that NPART = MPAR. The comparison proposed is valid in the neighbourhood of the solution. This can be simulated by using a vector of calculated function values, obtained with the initial parameter values, as new observation vector. To this purpose calculated values are rounded to numbers with five significant figures.

First derivatives can be tested with the aid of the $(\lambda, S(\lambda))$-relationship. The slope to the curve at $\theta^{(1)}(\lambda)$ being given by (9.3.1), which with differential corrections reads, see (2.4.13) and (2.4.9)

$$S'(\lambda) = -2[f_0(\theta^{(1)}(\lambda))]^T J(\theta^{(1)}(\lambda))\,(J^TJ)^{-1}J^Tf_0 \tag{12.1.1}$$

where vectors and matrices without argument are to be evaluated at $\theta^{(0)}$ and where $\theta^{(1)}(\lambda) = \theta^{(0)} + \lambda d^{(0)}$.

In this expression the first derivatives occur evaluated at two points on the fitting surface. Approximation to the slope is obtained by means of central differences at λ_i given by

$$S'(\lambda_i) \simeq \frac{S(\lambda_{i+1}) - S(\lambda_{i-1})}{\lambda_{i+1} - \lambda_{i-1}}$$

where in this case, according to the program option, $\lambda_{i+1} - \lambda_i = \lambda_i - \lambda_{i-1}$. Values obtained this way can be compared with those calculated with (12.1.1). They are printed on the same line in the output list by appropriate statements in subprogram MIN.

Second derivatives can be tested by means of the optimal step factor and the weights obtained with scale factors. The optimal step factor is found empirically in each fitting cycle in subprogram MIN and printed in a summary of the entire fitting process by subprogram LISTING. On the same line of the output list, values calculated with (9.4.7) are mentioned. In this last mentioned equation use of second derivatives is essential. Empirically and analytically determined values can be compared as done in Table 9.

Comparison can also be made for the course of the scale factors determined with formulas for differentials (Section 5.3) and for differences (Section 5.4). The first method requires the use of the matrix M_{12}, defined in (2.5.10), which is compiled of

Table 13. Modifications to the default deck structure of main program NLV.

No	Objective	Method	Reference	How to perform	Modification No in Appendix 1.5
1	Test of the formulas for first and second derivatives.	Comparison of numerical values obtained by using derivatives with empirically found values.	Sec. 12.1	Insert specific statements	1.1
2	Properties of condition function and starting point.	Univariate search.	Ch. 13	CALL NEWZERO	2.1
		One-step combinatorial search.	Ch. 13	CALL COMBIN	2.2
3	Selective use of parameters.	Multivariate search automatically ordered by partial cosines.	Eq. (2.2.11)	CALL ORDER	3.1
		Multivariate search defined by parameter permutation.	Sec. 8.3.1	CALL BLOCK	3.2
4	Application of algorithms.	Steepest descent.	Eq. (2.4.9)	CALL MIN (CRCTN = ANORM)	4.1
5	Selected paths.	Controlled approach.	Ch. 11	CALL TRACK (NTYPE = 1,2)	5.1
		Additional surfaces.	Sec. 4.9	CALL MIN(JUMP = 1)	–
6	Simulation of reversed type of convergence.	Reflection of the terminal point of the observation vector in			
		– each further tangent plane,	Ch. 13	CALL TRACK(NTYPE = 3)	6.1
		– the terminal fitted point.	Ch. 13	CALL TRACK(NTYPE = 4)	6.2
7	Intermediate information on the fitting process.	Listing of intermediate results.	Ch. 13	Redefine INKi options, Ch. 13	–
		Listing of $(\lambda, S(\lambda))$-relationship.	Ch. 13	CALL MIN(ITAN = 0)	7.1
		Calculation of curvature along each parametric curve separately.	Eq. (2.8.4)	CALL MIN	7.2
8	Speeding up convergence using first derivatives.	Scale factor correction.	Sec. 5.4	Insert specific statements	8.1
		Correction for curvature in direction of search.	Ch. 6	CALL BACK	8.2
		Simulation of convergence.	Ch. 10	CALL LIHYPEX	8.3
9	Speeding up convergence using second derivatives.	Step factor prediction.	Eq. (9.4.7)	Insert specific statements	9.1
		Scale factor correction.	Sec. 5.3	Insert specific statements	9.2
		Use of the Hessian.	Sec. 3.8	Redefine N02 = 1	–
10	Information on results using first derivates.	Empirically found curvature of the fitting surface.	Ch. 13	CALL TRACK(NTYPE = 5,6)	10.1
11	Information on results using second derivatives.	Curvature in direction of search.	Sec. 2.8	Redefine INKi options, Ch. 13	–

second derivatives, the second method uses first derivatives only. Comparison is performed best with results obtained in the second cycle.

12.2 Default algorithm

The default deck structure of the main program NLV, given in Appendix 1.2, produces the Gauss-Newton algorithm with step factor optimization. Under control of default options the output contains the following main items

- Numerical analysis of the convergence process in which is involved the calculation of the partial cosines; the total cosine; the length of the vector of differential corrections and the length of the normal vector for the first NPART and the first NPAR permuted parameters.
- Numerical analysis of the fitting procedure by means of a list of intermediate and final results for parameter values; optimal values of the step factor; reduction of the sum of squares to the fitting surface; course of the sum of squares to the tangent plane; course of the multiple cosine and the multiple correlation.
- Numerical information on scale factors.
- Test on normality of the distribution of residuals (Fisher, 1958) and a 95% confidence interval about the empirical frequency distribution of the residuals according to the Kolmogorov-Smirnov statistic (Siegel, 1956).
- Determination of a linearized 95% confidence interval in the tangent plane about the final parameter values.

The default deck can be modified to be suitable for various algorithms and procedures developed in earlier chapters.

12.3 Selective use of main program NLV

In Table 13 useful modifications are collected. Arguments of the subroutine subprograms are omitted unless their use is needed to perform a specific routine. In that case adequate values are listed in the arguments by replacement statements, it being understood that the right-hand side of the replacement statement has to be inserted only. Details on arguments are given in Chapter 13.

13 Description

13.1 Function subprograms

In this section subroutine subprograms that need updating for every new function to be optimized are briefly commented.

SUBROUTINE INITL

This subroutine initializes values of variables to produce a correct first fitting cycle. Summation fields are set zero, devisors are set 1.

ENTRY DEFAULT Algorithm and process parameters are defined in this entry. The user can redefine the default values in the main program NLV, preferably after the CALL DEFAULT statement.
ENTRY HEADING This entry produces a list of default values and values actually used, a list of initial parameter values and bounds to parameters if any. Appropriate values have to be defined in the main program NLV.
*COMDECK XINITL Update deck for specific headings, comments and legends to formulas and symbols to be used.

SUBROUTINE READ (IREADD, IWRITED, LAST1)

IREADD index to read data.
$\leqq 0$ specific read statements bypassed.
> 0 data to be read in specific routine.
IWRITED index to write data.
$= 0$ output list of data is not produced.
$\neq 0$ output list of data will be produced.
LAST1 number of data to be listed if IWRITED $\neq 0$.
$= 1, = 2, \ldots, =$ NDATA.
Last data is listed for any value of LAST1 $\leqq$ NDATA.

*COMDECK RDDATA Update deck for specific statements to read data.
*COMDECK WTDATA Update deck for specific statements to write data.

Observations are stored in the matrix X and the vector YOBS. Working field for

observed function values X(I,1) = YOBS(I). If IREADD > 0 the specific routine to read and to count data is performed. Two lists of initial parameter values are produced. The first in the original order of occurrence, the second according to permutation and grouping as: to be fitted, informative or constants.

ENTRY NEWZERO Arguments are transformed into the following variables.

PROC correction in % of the value of the parameter to define step length in alternating directions according to the parameters.
IREPT1 number of times to repeat the procedure for each parameter to be fitted.
IREPT2 number of times the entire procedure is to be repeated.

An alternating direction method (univariate direct search) is applied. To each parameter separately a step length is added as well as subtracted. If the central value has not the lowest sum of squares the procedure is extended for at most IREPT1 times in the direction of the lowest sum of squares. For all parameters the entire procedure is repeated IREPT2 times unless all parameters in order of search have their lowest sum of squares at the central value of the last three investigated values.

SUBROUTINE FNCTN (C,LSTC,SQ,LSTSQ,ISTOP,YCLC,YOC,LISTY,LAST1)

C parameter values for which the condition function is to be evaluated.
LSTC index to list values of C.
  = 0 no list is produced.
  ≠ 0 output list will be produced if INK3=3.
SQ sum of squares.
LSTSQ index to list observed function values, calculated function values, differences between them and SQ.
  ≦ 0 no list is produced.
  > 0 output list will be produced if INK3=3.
ISTOP index to return to calling program after function has been evaluated.
  = 1 instant return takes place.
  ≠ 1 no instant return, calculation of multiple cosine and multiple correlation between X(I,1) and YCLC(I) is performed.
YCLC calculated function values.
YOC differences between observed and calculated function values.
LISTY index to list specific output, output to be programmed by the user.
  ≦ 0 no list is produced.
  > 0 output list will be produced.
LAST1 see SUBROUTINE READ.

* COMDECK NFNCTN Update deck for specific statements for function evaluation.

*COMDECK WFNCTN Update deck for specific output list of function values.

Function values calculated with the parameter vector C(K) are stored in YCLC(I). Differences between X(I,1) and YCLC(I) are stored in YOC(I) and their sum of squares in SQ. Informative results are the multiple cosine and the multiple correlation between the vectors X(I,1) and YCLC(I), which are stored in respectively XCRLMT and XCOSMT in COMMON. The sum of the YOC(I) values is calculated and printed unless LSTSQ $\leqq 0$.

SUBROUTINE DFDA (C,H,YCLC,YOC,LSTH,LSTFA,LAST1).

C parameter values for which the first derivatives are to be evaluated.
H scale factors.
YCLC calculated function values.
YOC differences between observed and calculated function values to be stored in FA(I,MT).
LSTH index to produce output list of scale factors.
$\leqq 0$ no list is produced.
> 0 output list will be produced if INK7=3.
LSTFA index to produce output list of evaluated first derivatives.
$= 0$ no list is produced.
$\neq 0$ output list will be produced if INK7=3.
LAST1 See SUBROUTINE READ.

*COMDECK NDFDA Update deck for specific statements for first derivative evaluation.

First derivatives evaluated with the parameter vector C(K) are stored in the matrix FA(I,K). The difference vector YOC(I) is stored in FA(I,MT). The square of the length of the derivative vectors are stored in FA2(K), the sum of squares in FA2(MT). Scale factor values are returned via the argument variable H(K), the square root of the sum of squares via H(NT).

ENTRY ORDER The last argument is only used in a specific case.

LAST1 if first derivatives are programmed in SUBROUTINE FNCTN define LAST1=99999

The first NPAR permuted parameters are ordered according decreasing partial cosines. Those parameters for which the absolute value of this cosine is greater than the mean of the absolute values for K=1,NPAR are selected as NPART to be fitted parameters.

SUBROUTINE D2FDA(C,SQ,H,YCLC,YOC,LSTD2,LAST1)

LSTD2 index to produce output list of evaluated second derivatives (first 9 permuted parameters at most).
$\leqq 0$ no list is produced.
> 0 an output list will be produced if INK7=3.
Other variables have already been defined.

*COMDECK ND2FDA Update deck for specific statements for second derivative evaluation.

Second derivatives evaluated with the parameter vector C(K) are stored in the matrix FAA. The square of the length of the derivative vectors are stored in FAA2(K). Scalar products that are elements of the matrix M_{12} are stored in the matrix FAFAA (K,L), those of the matrix N_{02} in F0FAA(K,L).

13.2 Gradient subprograms

In this section subroutine subprograms that need no updating for new functions to be optimized are briefly commented.

SUBROUTINE NRMEQ(YOC,CNORM,LSTNM,ISTOP,H,LSTEQ,KCOS,M02)

YOC see SUBROUTINE FNCTN.
CNORM normal.
LSTNM index to produce output list of the normal.
$= 0$ no list is produced.
$\neq 0$ output list will be produced if INK7=3.
ISTOP index for calculation of the normal and sum of squares only.
$\neq 0$ no further calculations are performed.
$= 0$ execution proceeds.
H scale factors.
LSTEQ index to produce output list of normal equations.
$= 0$ no list is produced.
$\neq 0$ output list will be produced if INK7=3.
KCOS index to calculate the cosine matrix out of the normal equations and the normal.
$= 0$ no cosines are calculated
$\neq 0$ cosines will be calculated and printed for first and last cycle only if LSUMRY =1.
M02 index to use the second derivatives of the condition function to obtain the Hessian $G = 2(M - N_{02})$.
$= 0$ no use is made of N_{02}
$\neq 0$ use is made of N_{02} unless no second derivatives are available.

Normal equations are calculated for the first NPAR permuted parameters. From these equations the cosine matrix of the cosines of the angles between FA(I,K) and FA(I,L) is calculated and the vector of the partial cosines of the angles between FA(I,MT) and FA(I,K).

SUBROUTINE SOLVE(C,SOLN,NP)

C parameter values for which the normal equations are to be compiled.
SOLN solution of differential corrections for NP parameters.
NP first NP permuted parameters for which the solution of the normal equations is stored in SOLN(NP).

Normal equations are solved by means of the Choleski-algorithm. Partial solutions for 1, 1(1)2,..., 1(1)NPAR parameters according to their permutation and grouping are stored in the upper triangle of the matrix M. The inverse matrix M^{-1} (diagonal and upper triangle) is stored in the lower triangle of the extended matrix M. For the first and last cycle the cosine matrix of M^{-1} is calculated and printed if LSUMRY $\neq 0$. The standard deviation of the estimates of the parameters are stored in STDEV(K).

SUBROUTINE HOWA

This subroutine subprogram investigates how the situation of the fitting procedure is at the initial point A for each new cycle. Numerical results are collected, checked and printed in the first and last cycle. A summary of solutions of the normal equations is produced. Informative results on the relative rate of change of scale factors and the curvature are printed. The stopping criterion is tested. The test is performed on basis of a cosine criterion COSCRIT. The default value is 1000 which means that cosines with three zeros after the decimal point fulfill the criterion. A scheme of the complete test is given in Table 14. For NYS2=4Hbb** the fitting procedure is considered to be terminated and one more cycle is performed by defining NCYCLS= NC+1 which leads to NOCLC=NCYCLS in the main program NLV in the next cycle.

Table 14. Variables that save the state of the cosine criterion. Initially four blanks (4Hbbbb) are assigned to all no-yes variables. These are changed in the following cases.

Variable	Cosine criterion fulfilled	
	no	yes
Partial cosines for to be fitted parameters	NYS1 = 4Hbb**	NYS2 = 4Hbb**
Partial cosines for informative parameters	NOYES1 = 4Hbb**	NOYES2 = 4Hbb**
Total cosine for to be fitted parameters	NYS3 = 4Hbb**	NOYES3 = 4Hbb**

ENTRY SUMRY This entry is called by main program NLV if NOCLC $\geqq$ NCYCLS. This condition is fulfilled automatically if NYS2=4Hbb** (see Table 14). A comprehensive summary of the numerical and statistical results will be produced (see Appendix 1.4.2).

ENTRY COMBIN This entry produces a combinatorial search. The minimum value of the sum of squares after one cycle for $\binom{NPAR}{1}$ to $\binom{NPAR}{NPART}$ parameters is calculated. The parameter vector is automatically permuted and reduced to the size of the best combination before the return to the main program takes place. Combinatorial search can be performed for the first 14 permuted parameters at most. The parameter vector can be permuted and grouped in advance in main program NLV.

SUBROUTINE MIN(CRCTN,LETLIST,ITAN,START,STEP,ITIMES,LPX, JUMP)

CRCTN vector of direction of search defined in the calling program.
= CORR for arbitrary direction of search.
= DELTA for differential corrections.
= NORMA for steepest descent at A.

LETLIST index to produce output list. LETLIST options override INK3, INK7 and INK8 conditions
= 0 no list is produced.
= 1 list of iteration steps (EPS, SQ-relationship) will be produced.
= 2 list of initital and optimal parameter values will be produced.
= 3 complete output of the iteration process will be produced.

ITAN index to define type of tangent to be used.
= 0 (LETLIST=0 is required). No iteration takes place but a table of the (EPS, SQ)-relationship will be produced emanating from EPS=START, with step length STEP and number of steps equal to ITIMES;
= 1 numerical tangent is used.
= 2 directional tangent is used.
= 3 cosine tangent is used.

START starting value if ITAN=0.

STEP step length if ITAN=0.

ITIMES total number of (iteration) steps that will be performed. At least 3 steps are done if ITAN $>$ 0. In the last case, however, the iteration stops before the defined number is reached when the required accuracy conditions are fulfilled.

LPX index to define the use of subprogram LIHYPEX.

JUMP index to define a jump to a second branch of the (EPS,SQ)-relationship.
= 0 no jump takes place.
= 1 minimum of a second branch will be used.

Since the direction of search is completely defined when entering this subroutine

the curvature in the direction of search is calculated here. The curvature in the direction of CRCTN is produced without determination of the minimum by taking ITIMES=0. Then the list parameter LETLIST is effective with 0 or 1. The geodesic curvature is stored in XGEO.

SUBROUTINE AISB

The statements to assign variables valid at B to variables valid at the initial point A are collected in this subroutine subprogram. The replacement is for A=B, YCLCA =YCLCB, YOCA=YOCB, HA=HB, ANORM=BNORM and SQA=SQB. Information on the course of scale factors from A to B is produced if LSUMRY=2.

SUBROUTINE LISTING (INDALL)

INDALL index to reduce information to be listed.
= 0 listing of partial cosines for each cycle.
= 50 listing of parameter values obtained in each cycle.
= 1 complete list of intermediate and final results obtained in each cycle.

Independent of the value of INDALL a summary of the iteration to a subminimum in each cycle is given with respect to number of iterations, optimal step factor, sum of squares with respect to the fitting surface and to the tangent plane and lengths of NORMAL and DELTA vectors.

ENTRY PLOT For sequential functions a plot of the curve against time can be produced on the line printer. An upper (TOP) and lower (BOTTOM) function value have to be defined in main program NLV.

ENTRY PUNCH. Final parameter values are punched or printed, controlled by the argument INDALL.

INDALL=1 final parameter values are punched on punch cards and can be used as input in further fitting cycles in main program NLV.
= 0 final parameter values are printed on the line printer in a convenient layout suitable for instant punching on cards. These can be used as input in further fitting cycles.

13.3 Particular subprograms

In this section subroutine subprograms that need no updating and for which loading is optional are briefly commented.

SUBROUTINE BLOCK(NBLOCK,NROTATE,NCYC,CCRIT)

NBLOCK index variable that governs the creation and use of permutation and grouping of parameters in blocks.
= 1,..., = 5 sequential definition of at most 5 blocks. Further arguments equal to zero. Default number of blocks NBLOCK=1. In this case the subroutine need not be called by main program NLV.
= 0 blocks are used in order of creation. Fitting procedures start with first BLOCK defined.

NROTATE number of blocks that sequentially will be used. After block number NROTATE has been used, the first block will be used again.

NCYC number of fitting cycles that the same block will at most be used.

CCRIT cosine criterion with respect to the total cosine. This criterion overrides the criterion COSCRIT when blocks are being used. It acts analogous to COSCRIT. If the criterion is fulfilled the next cycle starts with the next block, whether or not the actual block in execution has already been used during NCYC cycles. If the criterion CCRIT is fulfilled for all blocks defined, execution terminates in a normal way in NLV.

The subroutine subprogram furnishes possibilities to make sequential use of different parameter permutation and grouping. Independent of the values in the argument, the fitting process terminates in a normal way if for all MPAR parameters the cosine criterion is fulfilled. A typical example of the use of blocks is two blocks with MPAR parameters among which some are bounded. If a parameter exceeds his bound in a certain fitting cycle and has to be deleted from the parameter vector, the remaining parameters may fulfill the cosine criterion. As oscillation of parameter values can occur, execution of the fitting procedure with the second block will automatically test whether the deleted parameter can be improved after all. In the affirmative the final result can be a fit for all MPAR parameters again.

SUBROUTINE BACK(NPZ,LISTZ,JACBZ,ICIRCZ,LAST1Z)

NPZ number of the first permuted parameters involved in this algorithm.
A suitable value is NPZ=NPART.

LISTZ index to produce output list.
= 0 no list is produced.
= 1, 2, 3 main results are printed and the value of LISTZ is assigned to LETLIST for output from subprogram MIN.
≧ 5 main results are printed without output from subprogram MIN.

JACBZ index that controls the required metric.
= 0 tangent space, metric I.
= 1 observation space, metric J^TJ.

ICIRCZ index to define the type of algorithm preferred.
= 0 search according to two linear directions.
= 1 circular search.
LAST1Z number of data to be produced in output list.

The subroutine subprogram is used to apply the back projection algorithm. Auxiliary vectors UDELT and VDELT are calculated. Optimization is for both step factor EPS1 = λ and direction of search EPS2 = 2 cos ϕ or EPS2 = ϕ.

ENTRY B1, ENTRY B2 Used to perform two-dimensional circular search.

SUBROUTINE TRACK(NTYPE,NP,START,NSTEP,KTIMES,NAUT,LIST)

In this subroutine several procedures are collected that transform the observation vector YOBS or that determine the curvature empirically. Procedures are defined by the variable NTYPE.

NTYPE=1 subdivision of the vector YOBS minus YCLCA to perform controlled approach.
= 2 subdivision of the vector YOBS minus YCLCB proceeding in the direction of search to perform controlled approach.
= 3 reflection of the terminal point of YOBS in the tangent plane before the minimum is determined.
= 4 reflection of the terminal point of YOBS in the terminal point $\theta^{(t)}$.
= 5 empirical determination of curvature of the fitting surface along intermediate fitting results by calculation of the distance perpendicular to the final tangent plane at $\theta^{(t)}$, and the distance perpendicular to the YOCA vector at $\theta^{(t)}$.
= 6 as before, but along the direction given by $\theta^{(t)} - \theta^{(0)}$ for 10 equal steps.

Use of further arguments depend on the value of NTYPE. They are listed in Table 15.

SUBROUTINE LIHYPEX (LHE,N1,N2,N3,NCRIT,LIST)

LHE index that defines the type of formula to be used.
= 1 a fourth degree polynomial is applied.
= 2 a hyperbolic function is applied.
= 3 an exponential function is applied.
N1,N2,N3 cycle numbers from which the results to be extrapolated are taken (N1< <N2<N3). If LHE=1 a fourth degree polynomial is calculated through the points obtained in cycle numbers N1+I*(N2−N1), where I=0(1)4.
NCRIT criterion for the accuracy of the iteration process calculating the constants if LHE=3; e.g. NCRIT=3.

Table 15. Use of arguments in SUBROUTINE TRACK.

NTYPE	NP	START	NSTEP	KTIMES	NAUT	LIST
1 and 2	number of parameters used in fitting procedure (e.g. NP = NPART)	starting value on YOCA 0. ≦ START < 1.0	number of parts in which the remainder of YOCA is divided	redefinition of number of fitting cycles (e.g. KTIMES = 5)	not used	= 1* gives list = 0 gives only main results
3 through 6	not used	not used	not used	not used	= 0 for a single outcome only = 1 proceeds automatically	= 1* gives list = 0 gives only main results

* For min (NDATA, 10) a list is printed out of YOBS(I), X(I,1) and YCLCA(I) to indicate the relationship between observed and used vectors.

LIST index that defines whether an output list of intermediate results will be produced.
= 0 no list is produced.
= 1 an output list will be produced.

The subroutine subprogram produces convergence of parameter values by extrapolation. When using this subroutine it is necessary to define in main program NLV the value for END greater than 1.

13.4 Default values

Values of algorithm and process parameters can be defined in main program NLV. Experience learned that several of them can be predefined for normal routines. Therefore a default value has been assigned to these parameters. A list of parameters and their default values is given in Table 16.

The contents of tables of optimization results in the output can be chosen by the values assigned to the logical unit in formatted write statements. This is governed by the indices INK3, INK7 and INK8 which take the value 3 to produce an output list and the value 7 to write information onto a scratch file. Pertinent combinations of values are given in Table 17. If INKi options are equal to 7, they override output options in the argument of the subroutine subprograms, except for SUBROUTINE MIN. As the definition of INKi options can occur anywhere in the main program they can be used to serve a single subprogram by changing the value of the logical unit before the CALL statement and redefining it after use to its default value again.

Table 16. Algorithm and process parameters, their function, default value and possible alternative values.

Parameter	Function	Default	Alternative
NCYCLES	Number of fitting cycles to be produced at most.	10	2,...,15
END	Number of times that NCYCLES cycles have to be repeated.	1	> 1
INK3	Index to define desired output.	7	3
INK7	Ibidem.	7	3
INK8	Ibidem.	7	3
LSUMRY	Index to define desired output. = 0 output mainly based on the first part of SUBROUTINE HOWA. = 1 no restrictions. = 2 as = 0 with additional information on scale factors and curvature.	1	0, 2
NDATA	Number of observations to be processed. If not defined in NLV, data will be counted in SUBROUTINE READ.	5HCOUNT	≦ MAXDAT
NOUTRD	Number of data in output list of SUBROUTINE READ, to avoid the production of long tables not necessary. If not defined in NLV the value of NOUTRD will automatically be equalled to min (NDATA, 50).	0 (NDATA)	≦ NDATA
NOUTFN	Number of data in output list of SUBROUTINE FNCTN.	1	≦ NDATA
NOUTDF	Ibidem for SUBROUTINE DFDA.	1	≦ NDATA
NOUTD2F	Ibidem for SUBROUTINE D2FDA.	1	≦ NDATA
NOUTFL	Number of data in output list of SUBROUTINE HOWA. If not defined in NLV the value of NOUTFL will automatically be equalled to min (NDATA, 50).	5HNDATA	≦ NDATA
COSCRIT	Variable to define the cosine criterion that is to be fulfilled to terminate the fitting procedure. Log_{10} (COSCRIT) furnishes number of leading zeros after the decimal point at least to be obtained.	1000.	$10.^n$, n > 0
EPS	Initial step factor along direction of search.	.001	> 0.
EPSEST	Step factor approximated by second derivatives if available.	.001	not used
EQS	Step factor reduction.	.66666	> 0.
RN	Fraction of the negative slope that is at least to be obtained in a given direction of search to conclude that a subminimum of S is reached.	.1	> 0.
RP	Ibidem for positive slope.	.1	> 0.
R	Fraction of progress along the chord to assess the minimum in a given direction of search. R = 1 gives 'Regula Falsi' algorithm.	1.	.0 < R ≦ 1.
IC(1),..., IC(6)	Six variables available for additional comments to be defined in NLV and printed during execution of ENTRY HEADING.	10L(BLANK)	10 Hollerith characters, start with 10L

Table 16. Continued.

Parameter	Function	Default	Alternative
REC1	Two parameters to be used to	0.0	1.0
REC2	define whether sequential treatment should be employed.	0.0	1.0
N02	Index to be used to define whether the Hessian should be applied.	0	1

Table 17. Values of the logical unit in formatted write statements defined by INKi (i = 3, 7 or 8).

To obtain	Process parameter		
	INK3	INK7	INK8
Complete output	3	3	3
Partial output of main results	3	7	3
Information about scale factors and curvature only	7	7	3
Output of intermediate results from special subroutines	7	3	7
Default output	7	7	7

Summary

Research workers who describe their problems with mathematical formulas in which variables and unknown parameters occur, have need for testing their working hypothesis with the aid of observations. Since most models consist of functions that are nonlinear in the parameters, iterative methods have to be employed to optimize the parameter values. The objective function to be minimized is the sum of squares of deviations of calculated function values from observed function values (Chapter 1).

In the present study spaces and surfaces that play a role in least squares techniques were investigated. Main spaces to be considered are the observation space and the parameter space (Chapter 2). Vectors of calculated function values are position vectors to the fitting surface in the observation space. This surface is covered with curvilinear coordinates: the parametric curves. Elements of differential geometry were used to investigate the role of scale factors and of curvature, both being essential properties of curvilinear coordinate systems. Analytical treatment requires the derivation of first and second derivatives of condition and objective function with respect to the parameters. To generalize the analysis, parameters were expressed as functions in which vectors and algorithm parameters occur; condition and objective function considered to be functions of these algorithm parameters. In many methods a step factor that controls the progress in a direction of search is the only algorithm parameter that is applied. In this study the derivation of derivatives has been extended to make the formulas suit algorithm parameter vectors. The notion of linear and strict parameter functions was introduced to particularize general formulas. Formulas for first and second derivatives for several functions that play a role in gradient algorithms are given. Formulas for the curvature of parametric curves were derived as well as those for the curvature of a path in an arbitrary direction of search on the fitting surface. For a simple condition function the use of formulas for arc length is given to demonstrate that it makes sense to distinguish between three important properties of nonlinear fitting surfaces: the curvature of the surface, the curvature of the parametric curves and the course of the scale factors along a path of search. The relationship between parameter, observation and response space is discussed. Geometrical interpretation gives insight in the slow convergence of normally used nonlinear curve fitting methods. This cumulated in the development of procedures that accelerate convergence.

Least squares methods have their own place in nonlinear optimization techniques, and algorithms developed for more general situations can be applied. On the other hand more specific methods sometimes can be considered valid for least squares prob-

lems. This for instance when second degree approximation to the fitting surface is accurate enough. Techniques that can be applied are briefly commented and main features that play a role in nonlinear optimization are discussed (Chapter 3). Gradient methods with step length optimization in each cycle use first derivatives of the condition function. Methods developed in this study make use of the gradient vector and the vector of differential corrections. It therefore was assumed that at least first derivatives of the condition function are available. Their availability gives the further advantage to be able to set stopping criteria based on the fact that it is an orthogonal projection of the terminal point of the observation vector on the fitting surface that is to be determined.

Condition functions may have a complicated structure in that they can be implicit functions, sequential functions or can consist of models involving alternative functions (Chapter 4). It was treated how a system can be set up with which the condition function and the first derivatives can be calculated systematically in these cases. A process parameter that governs the type of use of sequential functions was introduced. Flowcharts were given that illustrate the general treatment of the mentioned types of condition function. The special structure of some of these functions gives complications when exploring the fitting surface. Along a chosen direction of search jumps to additional fitting surfaces can occur and it was explained in which way results have to be interpreted in these cases.

Analysis of the fitting surface and its coordinate system of parametric curves gave an indication on the cause of slow convergence. The concept of using scale factors as weights to the direction of search was worked out analytically (Chapter 5). Two types of weights were introduced using differences and differentials of scale factors respectively. In the second case second derivatives of the condition function must be available. With an example the acceleration of convergence – expressed as decrease in number of new directions of search necessary to fulfill default stopping criteria – was demonstrated. A reduction was found from 25 cycles for the modified Gauss-Newton to 11 cycles for the scale factor differential correction method. Application of differences of scale factors has the disadvantage that two points in the direction of search must be calculated. However, the first stage of the approach to the final parameter values appeared to be faster than with the aid of differentials.

The deviation of points on the path of search on the fitting surface from the direction of differential corrections in the tangent plane – caused by the curvature of the parametric curves and scale factor variations – was measured by orthogonal projection of the path of search in the fitting surface, on the tangent plane (Chapter 6). Since this concerns the same tangent plane as the one used for the determination of the direction of search, this method was called the back projection method. To correct deviations from the required direction, the projection of an intermediate point found by back projection was reflected in the total tangent vector on the tangent plane. Paths found this way may still not be satisfactory because of the properties of the fitting surface. Therefore two algorithms were developed to produce in back projection a two-dimensional linear and nonlinear search respectively, to optimize both the

direction of search and the step length. The method appeared to be most effective when applied in the tangent space using relationships between parameter vectors rather than between vectors in the observation space. For the example chosen, back projection gave slightly better results in that less fitting cycles were required than when using scale factor corrections.

Subdivision of the program makes it possible to link subprograms in specific ways. To make the program suitable for investigating the condition function as well as the convergence process, an analysis was made of the subroutines that occur in gradient methods. Considerations that lead to the subdivision of the program in subroutine subprograms are discussed (Chapter 7). It appeared to be useful to divide the program into three groups of subprograms: subprograms that need updating for each new function to be fitted, subprograms that need no updating and contain the essential parts of gradient algorithms and finally subprograms that need not necessarily be loaded since they consist of specific subroutines. The subprograms are linked by the main program NLV that can be modified to apply special procedures. Attention was paid to the problem of how and when to terminate the execution of an optimization algorithm. Stopping criteria based on the geometric interpretation of the algorithm were introduced and compared. Although the value of some of these criteria – viz. criteria on the total cosine and on partial cosines – converges to zero in the optimal point on the fitting surface, this value is not reached by a monotone decrease of the absolute values of the cosines, not even when the objective function in subsequent fitting cycles is decreasing monotone. It is of importance to recognize this behaviour of the process to be able to judge on the basis of intermediate results whether or not execution is to be terminated. It appeared to make sense to distinguish two types of convergence. Type I was defined as the type for which in subsequent cycles the sum of squares with respect to the fitting surface decreases and, at the same time, the sum of squares with respect to the tangent plane increases. Type II was defined as the type for which both sums of squares decrease at the same time. Experience learned that Type II convergence indicates that progress will be slow. This was demonstrated with an example where both types occur alternately.

Condition functions can be given a particular form by choosing particular values for one or more of its parameters. The program thus must give the possibility to keep constant any of the parameters in order to omit parts of the condition function. In the program this is generalized by an option to permute the components of the parameter vector and to partition the parameters into three groups (Chapter 8). The first group consists of those parameters that have to be fitted, the second group of those that need not to be fitted but give additional information and the third group of those parameters that are kept constant. This partitioning is achieved by putting parameter index cards in the main program NLV in the required order. Permutation and partitioning then takes place automatically throughout all subroutines in the subprograms. For economic reasons the calculation of normal equations is confined to parameters occurring in the first two groups only.

A central part of the entire optimization process is the determination of the mini-

mum response in a given direction of search (Chapter 9). The search for such a subminimum can be done by methods using first and second derivatives of the objective function. First derivatives are useful in setting efficient stopping criteria, second derivatives can be used to give a prediction of the step factor. With second derivatives the geodesic curvature in the direction of search can be determined. From the curvature the radius of curvature can be derived. In the given example fast convergence occurs when this radius is large and consequently the curvature of the path of search is small. Convergence is slow when the radius of curvature is small. Then the path of search deviates appreciably from the path as indicated by the total tangent vector on the tangent plane. Scale factor correction and back projection method were developed to correct for this deviation.

The search for a subminimum can be made more general by introducing nonlinear parameter functions. It then is necessary to determine separately the correction to the parameters and the direction of search, by differentiating the parameter update formula with respect to the algorithm parameter. This generalization, already applied for the back projection method with two dimensional nonlinear search, can also be applied to an empirical approach (Chapter 10). Slow convergence sometimes seems to justify a linear extrapolation of intermediate parameter values. It appeared to be more efficient to extrapolate intermediate results for each parameter separately with simple nonlinear functions. With a few preliminary results available, this extrapolation is useful because during the extrapolation procedure no fitting cycles need be executed. In the given example it seemed if about 10 cycles were skipped when using exponential extrapolation and only 7 further cycles were needed instead of the 19 further cycles required for optimization without intermediate extrapolation.

The only restriction laid upon the intermediate results of the convergence process with methods thus far considered is that they be optimal with regard to moves into chosen directions. A method was developed where intermediate points were required to be the foot of a perpendicular from the terminal point of intermediate observation vectors to the fitting surface (Chapter 11). Heavy oscillation on the fitting surface of intermediate results thus is avoided, as was illustrated with an example. It also appeared that the curvature properties of the parametric curves are of more importance than those of the fitting surface itself. For this reason scale factor correction and back projection are efficient methods. The parametric curves that are passed when following the path of controlled approach can be sketched. In this way an insight can be obtained of the pattern of the parametric curves on the fitting surface. This was illustrated with an example.

For the algorithms and methods developed, a generally applicable computer program for nonlinear parameter optimization was written. Since several algorithms are programmed for analytical methods as well as difference or empirical methods, their results can be used to test the formulas for first and second derivatives (Chapter 12). Since difference methods work best in an approximately linear situation, rounded calculated function values are used as observation to simulate this situation in the neighbourhood of the initial point. When testing of first and second derivative formu-

las gives a positive result, optimization algorithms can be applied.

Methods introduced in this study can easily be made operational with the main program NLV. The default algorithm of the program is the modified Gauss-Newton algorithm. A short description of the subprograms and the use of their arguments is given (Chapter 13). The complete text of the program and the instructions to modify the main program NLV are given in Appendix 1.

Samenvatting

Een bijdrage tot theorie en praktijk van niet-lineaire parameter-optimalisering

Bij het leggen van een verband tussen variabele grootheden wordt aan de variabelen de voorwaarde van het moeten voldoen aan een mathematische betrekking opgelegd. Van een dergelijke betrekking wordt aangenomen dat daarin variabelen en parameters voorkomen en dat waarnemingsuitkomsten voor de variabelen kunnen worden ingevuld. Aan de parameters wordt de eis opgelegd dat ze een waarde moeten aannemen zodanig dat de berekende functiewaarden zo goed mogelijk aan de bijbehorende waarnemingen zijn aangepast. De doelstellingsfunctie waarmede de uitdrukking 'zo goed mogelijk' operationeel wordt gemaakt is in het hier beschreven onderzoek de som van kwadraten van afwijkingen tussen gemeten en berekende functiewaarden – ofwel het kwadraat van de lengte van de verschilvector – aangeduid als de respons. Kleinste kwadraten-technieken bestaan er uit deze respons te minimaliseren. Aangenomen wordt nu dat minstens één van de parameters niet-lineair in de voorwaardefunctie voorkomt. In het algemeen zal de aanpassing dan – beginnend met een beginschatting – iteratief in een aantal ronden moeten plaatsvinden – om de optimale parameterwaarden – de eindoplossing – te leren kennen (hoofdstuk 1).

Eigenschappen van ruimten en oppervlakken die een rol spelen in kleinste kwadraten-technieken werden onderzocht (hoofdstuk 2). De belangrijkste ruimten zijn de waarnemingsruimte en de parameterruimte. Vectoren waarvan de kentallen berekende waarden van de voorwaardefunctie zijn, zijn plaatsvectoren van het vereffeningsoppervlak dat in de waarnemingsruimte ligt. Dit oppervlak bevat een stelsel kromlijnige coördinaten: de parameterkrommen. Een baan waarlangs het vereffeningsoppervlak kan worden onderzocht – of: waarlangs de respons wordt bepaald – kan worden uitgedrukt in deze kromlijnige coördinaten. Met behulp van differentiaalmeetkunde werd de invloed van twee essentiële grootheden van kromlijnige coördinaten, de kromming en de schaalfactoren, op het convergentieproces onderzocht. Een analytische behandeling van het hier geformuleerde probleem vereist het bepalen van eerste en tweede afgeleiden van de voorwaardefunctie en de doelstellingsfunctie naar elk van de parameters. Om tot een generalisatie te komen zijn parameters opgevat als functies van algorithmeparameters waarin ook vectoren en scalaire grootheden kunnen voorkomen. Hiermede worden voorwaardefunctie en doelstellingsfunctie beschouwd als functies van deze algorithmeparameters. Afgeleiden werden bepaald voor het algemene geval dat de voorwaardefunctie een functie is van een algorithmevector. Het begrip lineaire en strikte parameterfunctie werd ingevoerd om weer tot speciale oplossingen

te komen aangezien veel methoden als enige – lineaire – algorithmeparameter een stapfactor bevatten die de staplengte langs de baan van exploratie bepaalt. Formules voor eerste en tweede afgeleiden van functies die een rol spelen bij gradiëntmethoden werden gegeven evenals de formules voor de kromming van parameterkrommen en van de geodetische kromming van de baan waarlangs het vereffeningsoppervlak wordt onderzocht. Voor een eenvoudige voorwaardefunctie werd een voorbeeld van toepassing gegeven van de formules voor het bepalen van booglengten onder meer om aan te geven dat het zinvol is drie belangrijke eigenschappen van niet-lineaire oppervlakken te onderscheiden: de kromming van het oppervlak, de kromming van de parameterkrommen en het verloop van de schaalfactoren langs een gekozen baan. Het verband tussen de parameter-, waarnemings- en responsruimte werd besproken.

Kleinste kwadraten-methoden nemen een eigen plaats in in niet-lineaire optimaliseringstechnieken; methoden ontwikkeld voor meer algemene problemen kunnen echter worden toegepast. Meer specifieke methoden zoals die welke zijn ontwikkeld voor problemen waarbij de te optimaliseren grootheid kwadratisch in de voorwaardefunctie voorkomt, kunnen worden toegepast wanneer een tweedegraads benadering van het vereffeningsoppervlak voldoende nauwkeurig is. Van de samenhang tussen optimaliseringsmethoden werd een kort overzicht gegeven (hoofdstuk 3). Gradiëntmethoden maken gebruik van eerste afgeleiden en aangezien in het hier beschreven onderzoek gebruik wordt gemaakt van de gradiëntvector en van de differentiaalcorrectievector, werd aangenomen dat van voorwaardefuncties in ieder geval eerste afgeleiden beschikbaar zijn. Hun beschikbaarheid heeft het verdere voordeel dat beëindigingscriteria kunnen worden geformuleerd die zijn gebaseerd op het feit dat het de loodrechte projectie van het eindpunt van de waarnemingsvector op het vereffeningsoppervlak is die moet worden bepaald. De theoretische waarde van het criterium is daarmee bekend.

Voorwaardefuncties kunnen een gecompliceerde structuur hebben aangezien hieronder zowel implicite functies, sequentiële functies als modellen waarin keuzemogelijkheden kunnen voorkomen, worden begrepen (hoofdstuk 4). Er werd aangegeven hoe een systeem kan worden opgezet waarmede ook in deze gevallen eerste afgeleiden systematisch kunnen worden bepaald en geprogrammeerd. Een procesparameter werd ingevoerd waarmede de keuze van het al of niet sequentieel gebruik van sequentiële functies kan worden bestuurd. Stroomdiagrammen werden gegeven ten einde de algemene behandeling van de genoemde typen functies te illustreren. Door de bijzondere structuur van sommige van deze functies kunnen complicaties ontstaan bij het onderzoek van de bijbehorende vereffeningsoppervlakken. In een gekozen zoekrichting kunnen discontinuïteiten voorkomen die sprongen naar nevenoppervlakken veroorzaken. Er werd uiteengezet hoe in deze gevallen verkregen uitkomsten moeten worden geïnterpreteerd.

De meetkundige interpretatie van de eigenschappen van het vereffeningsoppervlak en het daarop gelegen coördinatensysteem geeft een inzicht in de reden van het voorkomen van langzame convergentie bij niet-lineaire optimalisering. Dit leidde tot het ontwikkelen van methoden waarmede de convergentie naar de eindoplossing wordt versneld. Het gebruik van schaalfactoren – te weten de lengte van de raaklijnvector

aan de parameterkrommen – als wegingsfactoren voor toe te passen correcties op beginschattingen voor de parameters werd analytisch uitgewerkt (hoofdstuk 5). Twee manieren voor het vaststellen van deze gewichten werden geïntroduceerd, namelijk het gebruik van differenties en het gebruik van differentialen van schaalfactoren. In het laatste geval moeten tweede afgeleiden van de voorwaardefunctie beschikbaar zijn. De met deze methoden te bereiken versnelling in convergentie werd met een voorbeeld toegelicht. Voor voorwaardefunctie en gegevens vermeld in appendix 2.3, waarvoor bij het gebruik van de Gauss-Newton methode met staplengte-optimalisering 25 volledige berekeningsronden nodig waren, waren met gebruik van correcties door middel van differentialen van schaalfactoren nog slechts 11 ronden vereist. Het toepassen van differenties heeft het nadeel dat eerst ook van een tweede punt op het vereffeningsoppervlak de schaalfactoren berekend moeten worden; het voordeel is dat geen tweede afgeleiden behoeven te worden berekend. Voor het kiezen van een tweede punt werden algorithmen gegeven waaruit met een voorbeeld bleek dat de gewogen nieuwe correcties het aantal banen benodigd om de eindoplossing te vinden sterk reduceerde, terwijl in de beginfase van het optimaliseringsproces een nog sterkere reductie van de kwadraatsom van afwijkingen werd bereikt dan met het toepassen van differentialen.

Wanneer op het vereffeningsoppervlak een baan wordt gevolgd die start in de richting gegeven door de differentiaalcorrecties berekend in het raakvlak aan het oppervlak, dan zullen de punten van deze baan na terugprojecteren op het raakvlak niet op de rechte liggen die opgespannen wordt door de vector gegeven door differentiaalcorrecties. Dit tengevolge van het feit dat op het vereffeningsoppervlak de baan gekromd is door de krommingseigenschappen van de parameterkrommen (hoofdstuk 6). Dit feit werd geanalyseerd hetgeen resulteerde in een rekenwijze waarbij de afwijkingen, hanteerbaar gemaakt in het raakvlak door middel van het terugprojecteren, werden gecorrigeerd. Ten einde de methode zo efficiënt mogelijk te maken werden twee algorithmen ontwikkeld waarbij tweedimensionale lineaire respectievelijk niet-lineaire optimalisering werd toegepast om zowel de staplengte als de zoekrichting zo goed mogelijk vast te stellen. De methode gaf voor het gegeven voorbeeld een versnelling van de convergentie die vergelijkbaar is met die welke verkregen werd met de methode waarbij schaalfactoren als gewichten werden gebruikt.

Bij het schrijven van een programma voor niet-lineaire optimaliseringstechnieken vallen een aantal subroutines te onderscheiden die tevens kunnen dienen om de eigenschappen van de voorwaardefunctie zowel als die van het convergentieproces te onderzoeken (hoofdstuk 7). Overwegingen werden gegeven die leidden tot een indeling van het totale programma in subroutine subprogramma's die met behulp van het hoofdprogramma NLV al naar de gewenste berekeningswijze op de vereiste wijze kunnen worden gekoppeld. De subprogramma's kunnen in drie groepen worden ondergebracht. De eerste groep bestaat uit de subprogramma's die voor elke nieuwe voorwaardefunctie moeten worden aangepast omdat daarin de functie en zijn afgeleiden voorkomen, de tweede groep bestaat uit de subprogramma's die de essentiële onderdelen van gradiëntmethoden bevatten en verder geen aanpassing aan nieuwe voorwaardefuncties behoeven terwijl tenslotte de derde groep bestaat uit subprogram-

ma's die geschreven zijn voor speciale algorithmen en procedures en die niet in de computer ingelezen behoeven te worden wanneer deze speciale algorithmen niet worden toegepast. Aandacht werd besteed aan het probleem van het beëindigen van het iteratieproces. Hierbij werd vooral nadruk gelegd op het gebruik van de cosinus van de hoek die de verschilvector maakt met de raaklijnvector aan elk van de parameterkrommen. Er werd op gewezen dat, hoewel het optimaliseringsproces dusdanig kan verlopen dat de kwadraatsom monotoon afneemt, dit niet het geval behoeft te zijn voor deze cosinussen. Deze kunnen in absolute waarde zelfs toenemen waarbij toch vorderingen worden geboekt in het convergentieproces. Twee typen van convergentie werden onderscheiden. Type I werd gedefinieerd als het type waarvoor de kwadraatsom met betrekking tot het vereffeningsoppervlak afneemt terwijl de kwadraatsom met betrekking tot het raakvlak toeneemt. Type II werd gedefinieerd als het type waarvoor deze beide kwadraatsommen tegelijkertijd afnemen. Uit ervaring bleek dat het optreden van Type II een aanwijzing inhoudt dat de convergentie naar de eindoplossing traag zal verlopen. Dit werd met een voorbeeld gedemonstreerd.

Voorwaardefuncties kunnen een specifieke gedaante worden gegeven door één of meer van de parameters constant te houden. Het computerprogramma moet dan de mogelijkheid bevatten deze parameters als constante te behandelen. Aangezien dit niet noodzakelijkerwijze de laatste parameters behoeven te zijn, dient het programma tevens de mogelijkheid te bevatten de kentallen van de parametervector te permuteren. Dit is in het programma algemeen gemaakt door een driedeling van de kentallen van de parametervector toe te passen (hoofdstuk 8). De eerste groep bestaat dan uit die parameters die moeten worden geoptimaliseerd, de tweede groep uit die parameters die aanvullende uitkomsten geven over het optimaliseringsproces alsof deze parameters daarin opgenomen zijn, de derde groep tenslotte bestaat uit die parameters die constant worden gehouden. De permutatie en driedeling wordt bereikt door parameterindexkaarten in het hoofdprogramma NLV in de vereiste volgorde te leggen. Permutatie en driedeling komen dan automatisch tot stand in alle subprogramma's. Ten einde de berekeningen doelmatig te doen verlopen worden de normaalvergelijkingen alleen opgesteld en opgelost voor de parameters uit de eerste twee groepen.

In het gehele optimaliseringsproces neemt het bepalen van een minimum in de richting waarin het vereffeningsoppervlak wordt onderzocht een centrale plaats in (hoofdstuk 9). Het zoeken naar zulk een subminimum kan worden gedaan met methoden waarbij zowel eerste als tweede afgeleiden van de doelstellingsfunctie worden gebruikt. Eerste afgeleiden werden gebruikt om de mate van nauwkeurigheid van het verkregen resultaat vast te stellen terwijl tweede afgeleiden, indien beschikbaar, kunnen dienen om de staplengte in de richting van exploratie vast te stellen. Met tweede afgeleiden kan bovendien de geodetische kromming worden bepaald van de baan waarin het vereffeningsoppervlak wordt onderzocht. Uit het gegeven voorbeeld valt op te maken dat de convergentie snel verloopt wanneer de kromming in het beginpunt gering is en de kromtestraal bijgevolg groot, terwijl de convergentie traag verloopt indien de geodetische kromming een grote waarde heeft. De methode waarbij gebruik wordt gemaakt van gewichten gebaseerd op het verloop van schaalfactoren en de methode

waarbij door afwijkingen terug te projecteren in het raakvlak correcties werden aangebracht op de richting waarin het vereffeningsoppervlak wordt onderzocht, werden ontwikkeld om juist op deze kromming te corrigeren.

Het bepalen van een subminimum werd gegeneraliseerd om het invoeren van niet-lineaire parameterfuncties mogelijk te maken. In dit geval moeten de correctie aan te brengen aan de parametervector en de richting waarin het vereffeningsoppervlak wordt onderzocht afzonderlijk worden berekend, de laatste door de parameterfunctie te differentiëren naar de algorithmeparameter. Deze generalisatie werd reeds toegepast voor de methode van het terugprojecteren maar kan ook worden gebruikt voor een meer empirische benadering van het convergentieproces (hoofdstuk 10). Er werd een rekenwijze opgesteld waarbij met behulp van reeds verkregen tussenresultaten de nadering naar de eindoplossing wordt gesimuleerd door voor elke parameter de tussenresultaten te extrapoleren. Voor deze extrapolatie werd zowel een aanpassing aan een vierdegraads polynomium gebruikt als een aanpassing aan een hyperbolische en aan een exponentiële functie. Tijdens het extrapoleren behoeven geen volledige vereffeningsronden te worden berekend hetgeen de hoeveelheid uit te voeren rekenwerk gunstig beïnvloedt. In het gegeven voorbeeld bleken tijdens de extrapolatie als het ware 10 ronden te zijn overgeslagen waardoor het eindresultaat na nog slechts 7 verdere ronden werd bereikt in plaats van de 19 ronden die nodig waren bij het consequent doorvoeren van de Gauss-Newton algorithme zonder extrapolatie van tussenresultaten.

De enige restrictie die aan de tussenresultaten tot zover werd opgelegd was dat deze ieder in hun eigen richting een optimale deeloplossing moesten vormen. Aan de tussenresultaten werd een verdere voorwaarde opgelegd ten einde banen op het vereffeningsoppervlak te verkrijgen die aan slechts geringe afwijkingen van een rechtstreekse nadering naar het eindpunt onderhevig zouden zijn (hoofdstuk 11). Een uitwerking werd gegeven voor de eis dat alle tussenresultaten op het vereffeningsoppervlak loodrechte projecties zijn van deelpunten van de verschilvector voor de beginschatting. Het bleek hierbij dat de krommingseigenschappen van de parameterkrommen een grotere invloed op het verloop van het convergentieproces hebben dan de krommingseigenschappen van het vereffeningsoppervlak zelf. De ontwikkelde procedure heeft het voordeel dat een inzicht kan worden verkregen in het patroon van de parameterkrommen voor de toegepaste voorwaardefunctie en de daarbij gebruikte gegevens.

De ontwikkelde algorithmen en methoden werden toepasbaar gemaakt in een computerprogramma dat door middel van een hoofdprogramma kan worden bestuurd en waarmee de vereiste modificaties kunnen worden gerealiseerd (hoofdstuk 12). Aangezien met het programma een aantal berekeningen zowel met analytische methoden kunnen worden uitgevoerd als met differentie en empirische methoden, biedt het programma tevens de mogelijkheid om door het vergelijken van beide uitkomsten de programmering van eerste en tweede afgeleiden te testen. De benodigde modificaties in het programma kunnen met het hoofdprogramma NLV tot stand worden gebracht.

Een korte beschrijving van de subprogramma's en het gebruik van hun argumenten is in een afzonderlijk hoofdstuk opgenomen (hoofdstuk 13). Voor de complete tekst van het programma en nadere instructies voor het gebruik zij verwezen naar appendix 1.

References

Aird, T., 1973. Computational solution of global nonlinear least squares problems. Ph. D. thesis. Purdue University, West Lafayette, Indiana: 120 p.

Booth, G. W. & T. I. Peterson, 1960. Nonlinear estimation: Forecasting by generalized regression methods. IBM, Math. and Appl. Dpt. New York. IBM-Share Program WL NLI. IBM-Princeton University. Part IV: 21 pp., Part V: 15 + 7 p.

Box, M. J., 1966. A comparison of several current optimization methods and the use of transformations in constrained problems. Comput. J. 9: 67–77.

Box, M. J., D. Davies & W. H. Swann, 1969. Mathematical and statistical techniques for industry. Monograph No. 5: Non-linear optimization techniques. Oliver and Boyd, Edinburgh. VIII + 60 pp.

Brooks, S. H., 1959. A comparison of maximum-seeking methods. Ops. Res. 7: 430–457.

Cauchy, A., 1847. Méthode générale pour la résolution des systèmes d'équations simultanées. Compt. Rend. hebdo. Séances Acad. des Sci. Paris 25: 536–538.

Crockett, J. B. & H. Chernoff, 1955. Gradient methods of maximization. Pacific J. Math. 5: 33–50.

Curry, H. B., 1944. The method of steepest descent for non-linear minimization problems. Q. appl. Math. 2: 258–261.

Davidon, W. C., 1959. Variable metric method for minimization. AEC Research and Development Report. Argonne Nat. Lab. 5990 (Rev). 21 p.

Davidon, W. C., 1969. Variance algorithms for minimization. In: R. Fletcher (ed.). Optimization. Academic Press, London and New York: 13–20.

Davies, M. & I. J. Whitting, 1972. A modified form of Levenberg's correction. In: F. A. Lootsma (ed.). Numerical methods for non-linear optimization. Academic Press, London and New York: 191–201.

Deming, W. E., 1948. Statistical adjustment of data. John Wiley and Sons, Inc., New York. X + 261 p.

Dixon, L. C. W., 1972. The choice of step length, a crucial factor in the performance of variable metric algorithms. In: F. A. Lootsma (ed.). Numerical methods for non-linear optimization. Academic Press, London and New York. p. 149–170.

Dixon, L. C. W., 1974. Nonlinear optimization: A survey of the state of the art. In: D. J. Evans (ed.). Software for numerical mathematics. Academic Press, London and New York. p. 193–218.

Draper, N. R. & H. Smith, 1967. Applied regression analysis. John Wiley and Sons, Inc. New York, London, Sydney. 407 p.

Evans, D. J., (ed.). 1974. Software for numerical mathematics. Academic Press, London and New York. 451 p.

Fiacco, A. V. & G. P. McCormick, 1968. Nonlinear programming: Sequential unconstrained minimization techniques. John Wiley and Sons, Inc., New York, London, Sydney, Toronto. 210 p.

Fink, D. H. & R. D. Jackson, 1973. An equation for describing water vapour adsorption isotherms of soils. Soil Sci. 116(4): 249–261.

Fisher, R. A., 1958. Statistical methods for research workers. Oliver and Boyd, Edinburgh, London. 356 p.

Fletcher, R., 1965. Function minimization without evaluating derivatives: A review. Comput. J. 8: 33–41.

Fletcher, R., (ed.). 1969a. Optimization. Academic Press, London and New York. 354 p.
Fletcher, R., 1969b. A review of methods for unconstrained optimization. In: R. Fletcher (ed.). Optimization. Academic Press, London and New York. p. 1–12.
Fletcher, R., 1972a. Conjugate direction methods. In: W. Murray (ed.). Numerical methods for unconstrained optimization. Academic Press, London and New York. p. 73–86.
Fletcher, R., 1972b. A survey of algorithms for unconstrained optimization. In: W. Murray (ed.). Numerical methods for unconstrained optimization. Academic Press, London and New York. p. 123–129.
Fletcher, R. & C. M. Reeves, 1964. Function minimization by conjugate gradients. Comput. J. 7: 149–154.
Hartley, H. O., 1961. The modified Gauss-Newton method for the fitting of non-linear regression functions by least squares. Technometrics 3: 269–280.
Hayes, J. G., 1974. Algorithms for curve and surface fitting. In: D. J. Evans (ed.). Software for numerical mathematics. Academic Press, London and New York. p. 219–233.
Himmelblau, D. M., 1972. A uniform evaluation of unconstrained optimization techniques. In: F. A. Lootsma (ed.). Numerical methods for non-linear optimization. Academic Press, London and New York, p. 69–97.
Hooke, R. & T. A. Jeeves, 1961. Direct search: Solution of numerical and statistical problems. J. Ass. comput. Mach. 8: 212–229.
Jacoby, S. L. S., J. S. Kowalik & J. T. Pizzo, 1972. Iterative methods for nonlinear optimization problems. Prentice-Hall, Inc., Englewood Cliffs, New Jersey. 274 p.
Jones, A., 1970. Spiral: A new algorithm for non-linear parameter estimation using least squares. Comput. J. 13: 301–308.
Kowalik, J., 1967. A note on nonlinear regression analysis. Austr. Comput. J. 1: 51–53.
Kowalik, J. & M. R. Osborne, 1968. Methods for unconstrained optimization problems. Elsevier, New York. 148 p.
Levenberg, K., 1944. A method for the solution of certain non-linear problems in least squares. Q. appl. Math. 2: 164–168.
Lootsma, F. A., (ed.). 1972. Numerical methods for non-linear optimization. Academic Press, London and New York. 440 p.
Marquardt, D. W., 1963. An algorithm for least squares estimation of non-linear parameters. SIAM J. 2: 431–441.
McCormick, G. P. & J. D. Pierson, 1969. Variable metric methods and unconstrained optimization. In: R. Fletcher (ed.). Optimization. Academic Press, London and New York. p. 307–325.
Murray, W., (ed.). 1972a. Numerical methods for unconstrained optimization. Academic Press, London and New York. 144 p.
Murray, W., 1972b. Fundamentals. In: W. Murray (ed.). Numerical methods for unconstrained optimization. Academic Press, London and New York. p. 1–12.
Murray, W., 1972c. Second derivative methods. In: W. Murray (ed.). Numerical methods for unconstrained optimization. Academic Press, London and New York. p. 57–71.
Murray, W., 1972d. Failure, the causes and cures. In: W. Murray (ed.). Numerical methods for unconstrained optimization. Academic Press, London and New York. p. 107–122.
Nelder, J. A. & R. Mead, 1965. A simplex method for function minimization. Comput. J. 7: 308–313.
Powell, M. J. D., 1972a. Minimization of functions of several variables. In: J. Walsh (ed.). Numerical analysis: An introduction. Academic Press, London and New York. p. 143–158.
Powell, M. J. D., 1972b. Problems related to unconstrained optimization. In: W. Murray (ed.). Numerical methods for unconstrained optimization. Academic Press, London and New York. p. 29–55.
Rosenbrock, H. H., 1960. An automatic method for finding the greatest or least value of a function. Comput. J. 3: 175–184.
Sargent, R. W. H. & D. J. Sebastian, 1972. Numerical experience with algorithms for unconstrained

optimization. In: F. A. Lootsma (ed.). Numerical methods for non-linear optimization. Academic Press, London and New York. p. 45–68.
Siegel, S., 1956. Nonparametric statistics for the behavioral sciences. McGraw-Hill Book Company, Inc. New York, Kogakuska Company, Ltd. Tokyo. 312 p.
Spang III, H. A., 1962. A review of minimization techniques for non-linear functions. SIAM Rev. 4: 343–365.
Spendley, W., 1969. Nonlinear least squares fitting using a modified simplex minimization method. In: R. Fletcher (ed.). Optimization. Academic Press, London and New York. p. 259–270.
Stanton, R. G., 1961. Numerical methods for science and engineering. Prentice-Hall, Inc., Englewood Cliffs, New Jersey. 266 p.
Stol, Ph. Th., 1962a. Een meetkundige beschouwing over de niet-lineaire vereffening. Nota 138, ICW, Wageningen. 23 p.
Stol, Ph. Th., 1962b. Het gebruik van schaalfactoren als hulpmiddel voor een snelle convergentie bij niet-lineaire vereffening. Nota 139, ICW, Wageningen. 22 p.
Struik, D. J., 1961. Lectures on classical differential geometry. Addison-Wesley Publ. Comp. Inc., Reading, Mass., London. 232 p.
Swann, W. H., 1972. Direct search methods. In: W. Murray (ed.). Numerical methods for unconstrained optimization. Academic Press, London and New York. p. 13–28.
Tornheim, L., 1963. Convergence in nonlinear regression. Technometrics 5: 513–514.
Walsh, J., (ed.). 1972. Numerical analysis: An introduction. Academic Press, London and New York. 212 p.
Wilde, D. J., 1965. Optimum seeking methods. Prentice-Hall, Inc., Englewood Cliffs, New Jersey. 202 p.

Appendices

```
1) THIS PROGRAM IS DEVELOPED BY PH.TH.STOL. HEAD SECTION MATHEMATICS. INSTITUTE
   FOR LAND AND WATER MANAGEMENT RESEARCH. 11 MARIJKEWEG. P.O.BOX 35. WAGENINGEN
   THE NETHERLANDS. TEL. 08370 - 19100.        LAST PROGRAM UPDATE 5 JANUARY 1975
2) THE PROGRAM IS WRITTEN IN FORTRAN EXTENDED AS DESCRIBED IN CONTROL DATA-S
   FORTRAN EXTENDED REFERENCE MANUAL. PUBLICATION NO 60176600 (1971) REVISION 8
   BY CONTROL DATA CORPORATION. 215 MOFFETT PARK DRIVE. SUNNYVALE. CALIFORNIA
   94086. USE IS MADE OF 7-CHARACTER VARIABLES. THE PROGRAM IS PUNCHED IN 026
   HOLLERITH CODE . OUTPUT IS FOR A 136 CHARACTER LINE PRINTER.
3) THE PROGRAM IS RUN ON THE CDC-6600 COMPUTER OF CONTROL DATA NEDERLAND. DATA
   SERVICES DIVISION. 5 J.C.VAN MARKENLAAN. RIJSWIJK (ZH). THE NETHERLANDS. BY
   MEANS OF THE MARC-II TERMINAL OF THE TECHNICAL AND SCIENTIFIC COMPUTER CENTER
   OF THE GOVERNMENT SERVICE FOR LAND AND WATER USE. 12 MALIESINGEL. UTRECHT.
4) PERFORMANCE OF THE PROGRAM IS UNDER SCOPE 3.3 (355.9 INSTALLED AT 08/08/73).
   AS DESCRIBED IN CYBERNET SCOPE 3.3 REFERENCE MANUAL BATCH AND REMOTE BATCH
   OPERATING SYSTEM. PUBLICATION NO 86618000 (1973) BY CONTROL DATA CORPORATION
   HQW25F. P.O.BOX 0. MINNEAPOLIS. MINNESOTA 55440.
5) THE PROGRAM IS SPLIT UP IN SEVERAL PARTS. ALL PROGRAM PARTS ARE ON FILES IN
   UPDATE CREATION MODE. THE ALGORITHM ROUTINES ARE FIXED EXCEPT FOR THE DIMEN-
   SIONS OF THE SUBSCRIPTED VARIABLES. THE VARIABLE PROGRAM PARTS CONSIST OF THE
   CONDITION FUNCTION AND ITS DERIVATIVES WITH RESPECT TO THE PARAMETERS.
   VARIABLE PROGRAM PARTS ARE USED AS UPDATE COMDECKS.  SCOPE 3.3 UPDATE SYSTEM
   PROGRAM IS USED TO MANIPULATE THE LIBRARY FILES TO LINK THE PROGRAM PARTS.
6)
   ARRAY SIZES ARE DETERMINED WITH THE AID OF THE INTEGERS MENTIONED BELOW

     NUMBER OF DIRECTIVE CONSTANTS IN NEW FUNCTION   C=NUMBER USED. ELSE C=1
     NUMBER OF PARAMETERS IN NEW FUNCTION            P=MPAR
     NUMBER OF DATA                                  N=NDATA
     NUMBER OF OBSERVED VARIABLES                    M=MAXVEC
     SECOND DERIVATIVES USED                         T=1H+ (NO). T=1H* (YES).
     DEFINE P1=P. M1=N. Q1=Q. CP1=CP.  IF(T.EQ.1H-)  THEN  P1=M1=Q1=CP1=1
     FROM THESE CALCULATE THE INTEGERS               Q=P+1 AND CP=Q*P/2

   THERE ARE NO RESTRICTIONS IN THE PROGRAM WITH RESPECT TO THE NUMBERS C. P. N
   AND M. HOWEVER. THE TABLEAU OF PARTIAL SOLUTIONS. PRODUCED IN SUBROUTINE
   LISTING (FORMATS 35. 37 AND 25) AND THE PARAMETER PERMUTATION TABLES PRODUCED
   IN SUBROUTINES INITL (FORMATS 32. 33 AND 34) AND BLOCK (FORMATS 41 AND 44)
   GIVE A PROPER LAYOUT FOR 30 PARAMETERS AT MOST. COMBINATORIAL SEARCH IS FOR
   THE FIRST 14 PERMUTED PARAMETERS AT MOST - DIMENSION KLAS=14 IN SUBROUTINE
   MOWA USED IN ENTRY COMBIN AND FORMAT 527 - . FORMATS 520. 521 AND 522 ARE
   FOR 30 PARAMETERS AT MOST.
7)
   ARRAY SIZES OF VARIABLE COMMON DIMENSIONS ARE (SEE APPENDIX 1.4.1)
*COMDECK DVAR
      COMMON A(P),B(P),MAMB(P),NAME(P),CORR(P),DELTA(P),STDEV(P)              1
      COMMON MA(Q),MB(Q),ANORM(Q),BNORM(Q),IP(Q),JP(Q)                        2
      COMMON YCLCA(N),YCLCB(N),YOCA(N),YOCB(N),YOBS(N)                        3
      COMMON KFIRC(P),ALB(P),AUB(P),IB(P),FA(N,Q),FA2(Q)                      4
      COMMON X(N,M),XA(15,P),XMA(15,P),KCOSN(15,P),IPX(15,P)                  5
      COMMON IPAR(C),H(Q,Q),XH(Q,Q),FAA(N1,CP1),FAFAA(P1,P1)                  6
      COMMON FAFAA(P1,P1),FFAAMH(P1),FSS(N1),CURVTOT(P1),CURVGEO(P1)          7

   ARRAY SIZES OF VARIABLE DIMENSIONS IN SUBPROGRAMS ARE
*COMDECK DIMINITL   DIMENSION ISTAR(P)
*COMDECK DIMFNCTN   DIMENSION C(P),YCLC(N),YOC(N)
*COMDECK DIMDFDA    DIMENSION C(P),H(Q),YOC(N),YCLC(N),ORDERED(P),IPERM(P)
*COMDECK DIMD2FDA   DIMENSION C(P1),FAA2(P1),H(Q1),IL(CP1),YCLC(N1),YOC(N1)
*COMDECK DIMNRMEQ   DIMENSION H(Q),CNORM(Q),COSN(Q),YOC(N)
*COMDECK DIMSOLVE   DIMENSION C(P),SOLN(P),E(P),BB(P),CN(Q),COSINF(Q),D(Q)
*COMDECK DIMMOWA    DIMENSION D(P),E(P),ISTAR(P),FCALC(N),FOC(N),TJFSS(P1)
                    DIMENSION DCURVE(P1)
*COMDECK DIMMIN     DIMENSION KS(P1),FKS(N1),FJS(N1),TJFSS(P1),DCURVE(P1)
                    DIMENSION CRCTN(P),IBN(P)
*COMDECK DIMLIST    DIMENSION A2A1(P),TOT21(P),ITALLY(P),TVALUE(P)
*COMDECK DIMBLOCK   DIMENSION IPB(5,P),JPB(5,P),ISTER(P)
*COMDECK DIMBACK    DIMENSION VDELT(P),UDELT(P),G(Q),YBA(N)
*COMDECK DIMTRACK   DIMENSION XAA(P),YCALC(N),YDIF(N)
*COMDECK DIMLIPEX   DIMENSION AA(P),BB(P),CC(P),DD(P),EE(P),ITEL(P)

   OTHER SUBSCRIPTED VARIABLES HAVE FIXED DIMENSIONS AND NEED NOT BE UPDATED.
8)
   INSTRUCTIONS FOR USE OF VARIABLE  AND PARAMETER  NAMES WHICH ARE OBLIGATORY

   MAIN PROGRAM NLV       INITIAL PARAMETERS TO BE DEFINED
                          ON PARAMETER VALUE CARDS BY          A(K)=....(K=1,MPAR)

   SUBROUTINE READ        OBSERVED FUNCTION VALUES             YOBS(I),I=1,NDATA
                          INDEPENDENT OBSERVED VARIABLES       X(I,J),I=1,NDATA
                                                                    J=2,MAXVEC
                          X(I,1) IS WORKING FIELD FOR
                          YOBS(I) AND CANNOT BE USED.
                          WHEN DATA ARE PUNCHED ON 1(1)NDATA
                          PUNCH CARDS THE FOLLOWING ROUTINE
                          CAN BE USED IN *COMDECK RDDATA
                          WHERE FINALLY THE ACTUAL NUMBER
                          OF DATA NDATA=MIN(NDATA,I-1)
                          I=0
                     1000 I=I+1  $  READ(2,2) X(I,2),YOBS(I)
                          IF(I.GT.NDATA) GO TO 999
                          IF(EOF(2)) 999,1000
                      999 CONTINUE
                        2 FORMAT(2F6.0)
                          NDATA=I-1

   SUBROUTINE FNCTN       CALCULATED FUNCTION VALUES           YCLC(I),I=1,NDATA
                          PARAMETERS TO BE FITTED              C(K),K=1,MPAR
                          INDEPENDENT OBSERVED VARIABLES       X(I,J),I=1,NDATA
                                                                    J=2,MAXVEC
                          TWO TYPES OF FURTHER VARIABLES IN
                          COMMON STATEMENTS ARE AVAILABLE      CPAR(L),L=1,25
                          AND                                  IPAR(J),J=1,C
                          FOR WHICH THE INTEGER C IS
                          TO BE DEFINED (IF NOT USED. C=1)
                          FOR IMPLICIT FUNCTIONS X(I,1) CAN
                          BE USED AS STARTING VALUE FOR
                          ITERATION ON YCLC(I)

   SUBROUTINE DFDA        VALUES OF FIRST DERIVATIVES          FA(I,K1),I=1,NDATA
                                                               K1,K2,....,KP
                                                               FOR P=MPAR FIRST
                                                               DERIVATIVES
                          FOR NAMES OF PARAMETERS AND
                          VARIABLES SEE SUB FNCTN
                          FOR IMPLICIT FUNCTIONS USE           YCLC(I),I=1,NDATA

   SUBROUTINE D2FDA       VALUES OF SECOND DERIVATIVES         FAA(I,K1),I=1,NDATA
                                                               K1,K2,....,KCP
                                                               FOR CP=P*(P+1)/2
                                                               SECOND DERIVATIVES
                          FOR NAMES OF PARAMETERS AND
                          VARIABLES SEE SUB FNCTN
                          FOR IMPLICIT FUNCTIONS USE           YCLC(I),I=1,NDATA
```

```
9)
   THE USE OF FIRST DERIVATIVES OF THE CONDITION FUNCTION IS OBLIGATORY AS THE
   ALGORITHMS IN THE PROGRAM ARE BASED ON THEIR AVAILABILITY.
   THE USE OF SECOND DERIVATIVES OF THE CONDITION FUNCTION IS OPTIONAL. THEIR
   USE THROUGH ALL SUBROUTINES IS GOVERNED MERELY BY THE CALL STATEMENT IN THE
   MAIN PROGRAM NLV.
10)TOTAL TIME TO READ THE ENTIRE PROGRAM ON A MARC-II READER ABOUT 25 MINUTES
   TOTAL TIME FOR UPDATE AND COMPILATION (OPT=2,L=0) ABOUT 75 DEC.SYSTEM SECONDS
   TOTAL TIME REQUIRED TO PRODUCE A LIST OF THE MAIN PROGRAM AND ALL SUBPROGRAMS
   UNDER R=3 OPTION ABOUT 45 MINUTES ON THE MARC-II TERMINAL LINE PRINTER.
11) PROGRAMS ARE COMPILED UNDER FORTRAN OPT=2 COMPILE MODE. DEFAULT FIELD LENGTH
   FOR LOADING CAN BE TAKEN 76000(OCT). A REDUCTION OF FIELD LENGTH BY MEANS OF
   A REDUCTION CONTROL CARD WILL BE HONORED FOR MODERATE SIZED PROBLEMS.
12) SUBROUTINE SUBPROGRAMS ARE CALLED BY MAIN PROGRAM NLV. INSTRUCTIONS FOR
   APPLICATION ARE GIVEN IN STOL (1975. APPENDIX 1.5) THE DEFAULT ALGORITHM IS
   GAUSS-NEWTON WITH STEP LENGTH OPTIMIZATION.
13) PARAMETERS PRESENT IN THE CONDITION FUNCTION CAN BE PERMUTED BY PARAMETER
   PERMUTATION CARDS IN THE MAIN PROGRAM NLV. GROUPING OF PARAMETERS IS ACHIEVED
   BY USING THE PARAMETER GROUPING CARDS IN NLV WHICH ARE FOR
     NPART = PARTIAL PARAMETER VECTOR TO OPTIMIZED.
     NPAR  = NUMBER OF PARAMETERS FOR WHICH NORMAL EQUATIONS WILL BE SOLVED.
             INFORMATION ABOUT THE NUMERICAL PROCESS IS OBTAINED .
     MPAR  = TOTAL NUMBER OF PARAMETERS THAT OCCUR IN THE CONDITION FUNCTION.
14)
   OUTPUT OF INTERMEDIATE RESULTS CAN BE OBTAINED BY CHANGING THE DEFAULT VALUES
   OF THE OUTPUT PARAMETERS. THIS CAN BE DONE IN MAIN PROGRAM NLV BY
   CHOOSING INK3=INK7=INK8=3 AND CHANGING THE OUTPUT VARIABLES IN THE ARGUMENTS
   OF THE SUBROUTINES. UNDER CONTROL OF DEFAULT VALUES THE OUTPUT OF ARRAYS
   WHICH DEPEND ON THE NUMBER OF DATA IS AUTOMATICALLY REDUCED TO MIN(NDATA,50).
15)
   EXECUTION TERMINATES IN THE FOLLOWING CASES BECAUSE OF AN ERROR:

   SUBROUTINE INITL       STOP 10   COMMON STATEMENTS IN MAIN- AND SUBPROGRAMS
                                    HAVE DIFFERENT DIMENSIONS OF ARRAYS. OR
                                    COMMON CARDS OUT OF ORDER.

                          STOP 11   NPART=0. PARAMETER PERMUTATION CARDS IN MAIN
                                    PROGRAM NLV NOT IN PROPER ORDER.

   SUBROUTINE READ        STOP 20   NUMBER OF DEGREES OF FREEDOM (NDATA-NPART)
                                    LESS THAN ZERO. NUMBER OF DATA (NDATA) HAS TO
                                    BE CHECKED.

   SUBROUTINE FNCTN       STOP 30   INITIAL PARAMETER VALUES OUTSIDE BOUNDS.
                                    DEFINITION OF BOUNDS AND INITIAL PARAMETER
                                    VALUES IN NLV SHOULD BE CHECKED.

   SUBROUTINE DFDA        STOP 40   LENGTH OF VECTOR OF DERIVATIVES (SCALE FAC-
                                    TOR) EQUAL TO ZERO. CHECK FIRST DERIVATIVE
                                    FORMULAS IN SUB DFDA OR CHOOSE NEW INITIAL
                                    PARAMETER VALUES IN MAIN PROGRAM NLV.

   SUBROUTINE SOLVE       STOP 50   ZERO DIAGONAL ELEMENTS OCCUR IN THE MATRIX H
                                    OF NORMAL EQUATIONS. IT IS ALSO TESTED
                                    WHETHER ZEROS OCCUR DURING THE NUMERICAL
                                    REDUCTION IN SOLVING THE NORMAL EQUATIONS.
                                    MATRIX SINGULAR. PARAMETRIC CURVE DIRECTIONS
                                    PROBABLY DEPENDENT. SUPERFLUOUS PARAMETERS
                                    CAN BE PRESENT IN THE CONDITION FUNCTION.

   SUBROUTINE MOWA   INFORMATIVE    OPTIMAL STEP LENGTH IS FOUND TO BE ZERO IN
                                    THREE CONSECUTIVE CYCLES. CORRECTION VECTOR
                                    PROBABLY ZERO  OR ALL PARAMETER VALUES TOO
                                    CLOSE TO PARAMETER BOUNDS.
                                    EXECUTION TERMINATES IN MAIN PROGRAM NLV AT
                                    STOP 1 (SEE BELOW).
                                    DER STOP 1 (SEE BELOW).

   SUBROUTINE MIN         STOP 60   SUBROUTINE MIN TWO TIMES IN A ROW ENTERED
                                    WITH STEP FACTOR (EPS) EQUAL TO ZERO. MORE
                                    MINIMA IN DIRECTION OF SEARCH CAN OCCUR. USE
                                    A WRITE OPTION IN THE ARGUMENT OF SUB MIN TO
                                    INVESTIGATE THE (EPS-SQ) RELATIONSHIP.

   SUBROUTINE MIN         STOP 61   IN TWO CONSECUTIVE ITERATIONS THE SIGN OF
                                    THE DIRECTION OF SEARCH CHANGED TO OBTAIN A
                                    NEGATIVE SLOPE AT EPS=.0. NO INDICATION FOR
                                    A SPECIAL ERROR. USE WRITE OPTION SEE STOP 60

   SUBROUTINE LINHYPEX    STOP 70   THE VALUE OF THE INTEGER END THAT DEFINES THE
                                    NUMBER OF REPETITIONS OF THE TOTAL FITTING
                                    PROCEDURE HAS TO BE GREATER THAN OR EQUAL TO
                                    2. REDEFINE END IN THE MAIN PROGRAM NLV.

                          STOP 71   NUMBER OF CYCLES TOO SMALL TO APPLY THIS
                                    SUBROUTINE. INCREASE VALUE OF NCYCLES IN
                                    MAIN PROGRAM NLV.

                          STOP 72   COLLINEARITY OF POINTS. HYPERBOLIC AND EX-
                                    PONENTIAL SIMULATION CAN NOT BE PERFORMED.
16)
   EXECUTION TERMINATES IN A NORMAL WAY IN THE MAIN PROGRAM NLV WHEN THE
   REQUIRED CONVERGENCE CONDITION IS FULFILLED OR WHEN THE DEFINED NUMBER OF
   CYCLES IS PERFORMED. THE ALTERNATIVES ARE:

   PROGRAM NLV            STOP  1   DEFINED NUMBER OF CYCLES HAS BEEN PERFORMED
                                    (DEFAULT NCYCLES=10. TO BE REPEATED END=1
                                    TIMES) MAXIMUM NUMBER OF CYCLES THAT CAN BE
                                    DEFINED IN NLV NCYCLES=15.

                          STOP 55   COSINE OF THE ANGLE  BETWEEN THE DIRECTION
                                    OF THE PARAMETRIC CURVES AND THE RESIDUAL
                                    VECTOR FOR THE FIRST NPART PERMUTED
                                    PARAMETERS LESS THAN A PRE-DEFINED CRITERION
                                    (DEFAULT VALUE 0.001. DEFINED BY COSCRIT=
                                    1000.). FULFILLED CONVERGENCE CONDITION IS
                                    CARRIED OVER THROUGH THE SUBROUTINES BY THE
                                    STATEMENT MYS2=4H  **. WHICH IS DEFINED IN
                                    SUBROUTINE MOWA.

   NUMBER OF STOP DEPENDS ON WHICH CONDITION IS FULFILLED FIRST. IN NLV SPECIAL
   PROCEDURES THAT HAVE TO BE CARRIED OUT WITH THE FINAL PARAMETER VALUES CAN
   BE INSERTED BEFORE THE STOP 55 STATEMENT.
17)
   EXECUTION TERMINATES IN A NORMAL WAY IN

   SUBROUTINE BLOCK                 EXECUTION TERMINATES IN PROGRAM NLV IF THE
                                    COSINE CRITERION IN THE ARGUMENT OF SUB BLOCK
                                    (CCRIT) IS FULFILLED FOR ALL BLOCKS DEFINED.

   MODIFICATIONS TO THE DEFAULT DECK RETURN TO THE CALLING PROGRAM AND EXECUTION
   TERMINATES AT STOP 1 OR STOP 55 IN MAIN PROGRAM NLV RESPECTIVELY.
18)
   OUTPUT STARTS WITH SUBJECT HEADING PRINTED BELOW THE SECOND JOB SEQUENCE
   NUMBER IF OUTPUT OF PROGRAM TEXT IS SUPPRESSED .
19) UNDER CONTROL OF DEFAULT VALUES STOP 55 TERMINATION PRODUCES A SUMMARY OF
   NUMERICAL AND STATISTICAL RESULTS OF THE FITTING PROCESS. UNDER STOP 1
   CONTROL THE PROCESS CANNOT BE REGARDED TO BE TERMINATED. HOWEVER. MOST
   PARTS OF THE INFORMATION ABOUT THE FITTING PROCESS IS GIVEN IN THIS CASE TOO.
20) THIS PROGRAM CONTAINS ALGORITHMS AND METHODS OF WHICH THE UNDERLYING THEORY
   IS GIVEN IN STOL. PHILIP TH. 1975. A CONTRIBUTION TO THEORY AND PRACTICE OF
   NONLINEAR PARAMETER OPTIMIZATION. PUDOC. WAGENINGEN. THE NETHERLANDS.
```

1.2 Default main program NLV

```
1)
   DEFAULT DECK MAIN PROGRAM NLV WITH PARAMETER PERMUTATION CARDS FOR A FOUR
   PARAMETER CONDITION FUNCTION. CARDS THAT HAVE TO BE REPLACED ARE DENOTED
   BY * IN THE FIRST COLUMN. SEE EXPLANATION IN SECTION 3).
2)
                 ***UNMODIFIED DEFAULT DECK FOR MAIN PROGRAM NLV***
*********************************************************************************
      PROGRAM NLV(INPUT,OUTPUT,TAPE2=INPUT,TAPE3=OUTPUT,TAPE7)
      REAL M  $   INTEGER  END

*INSERT COMDECK DVAR  (CARD 1 - 7)
*INSERT COMDECK DFIX  (CARD 8 - 16)

      MPAR= 4   $  MAXVEC= 2   $  MAXDAT= 20   $  NB=0   $  A(1)=CK=1.     17 NLV
      CALL DEFAULT                                                          / N02=0
*NCYCLES=10/END=1/INK3,7,8=7/LSUMRY=1/NOUTFN,DF,D2F=1,RD=FL=N/COSCRIT=(3)/IC=(6)

*REDEFINE DEFAULT VALUES IF DESIRED

*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ****
***             INITIAL VALUES, BOUNDS AND PERMUTATION OF PARAMETERS          ***
*** - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -***

*INSERT PARAMETER VALUE CARDS FOR ALL MPAR PARAMETERS
*INSERT PARAMETER BOUNDARY CARDS IF NECESSARY

C     BLOCK 1
       K=0                                                                    *
       K=K+1    $    IP(K)= 1                           $    JP(IP(K))=K
       K=K+1    $    IP(K)= 2                           $    JP(IP(K))=K
       K=K+1    $    IP(K)= 3                           $    JP(IP(K))=K
       K=K+1    $    IP(K)= 4                           $    JP(IP(K))=K
                                NPART = K                                     1
                                NPAR  = K                                     2
                                MPAR  = K                                     3
*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ****
***                 HEADING, INITL, READ, NOCLC=1 THROUGH 2 FORMAT            ***
*** - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -***
      CALL HEADING
      CALL INITL
      CALL READ (1,1,NOUTRD)
      IREP = 0
   10 IREP = IREP+1   $          I3=3           $   IF(LSUMRY.NE.1) I3=7
      NOCLC= 1          $  WRITE(I3,2) NOCLC    $   DO 5   K = 1,NPAR
    5                                                   B(K)=XA(1,K)=A(K)

    2 FORMAT(1H1/**************START OF CYCLE NO*I4*  ***6/)
*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ****
***                          ALGORITHM STATEMENTS                            ***
*** - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -***
               NC=0                                 $   NCYCLS=NCYCLES
   20          NC=NC+1
      CALL FNCTN (A,1,SQA,1,8,YCLCA,YOCA,1,NOUTFN)
      CALL DFDA(A,MA,YCLCA,YOCA,1,1,NOUTDF)
      CALL D2FDA(A,SQA,MA,YCLCA,YOCA,1,NOUTD2F)
      CALL NRMEU(YOCA,ANORM,1,0,MA,1,1,NB2)
      CALL SOLVE(A,DELTA,NPART)
      CALL MOVA
                IF(NOCLC.GE.NCYCLS) GO TO 30   $   NOCLC=NOCLC+1
      CALL MIN(DELTA,0,1,.0,.8,20,0,0)
      CALL AISB
                IF(NC.LT.NCYCLS)     GO TO 20
*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ****
***   30 OUTPUT ROUTINE, FOR CALL PUNCH(1): DEFINE FILE PUNCH IN PROGRAM CARD***
***   40       FOR CALL PLOT, DEFINE UPPER AND LOWER VALUES                  ***
*** - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -***
   30 CONTINUE
      CALL SUMRY,RETURNS(40)
              IF( IREP .LT. END ) GO TO 10
      CALL PUNCH(0)                                                       LIST
      STOP  1

   40 CONTINUE
      STOP 55
      END
---------------------------------------------------------------------------------
 3)
   AN EXECUTABLE DEFAULT DECK IS OBTAINED AS FOLLOWS. (SEE ALSO APPENDIX 1.1)
*INSERT COMDEK DVAR
   INSERT AT THIS PLACE *COMDECK DVAR (SEE APPENDIX 1.4.1 FOR COMPLETE EXAMPLE)
   A COMDECK WITH THE SAME DIMENSIONS AS USED IN THE SUBPROGRAMS IS OBLIGATORY.
*INSERT COMDECK DFIX
   INSERT AT THIS PLACE *COMDECK DFIX (SEE APPENDIX 1.4.1 FOR COMPLETE EXAMPLE)
*REDEFINE DEFAULT VALUES
   IF DESIRED CHANGE DEFAULT VALUES HERE.
*INSERT PARAMETER VALUES
   INSERT AT THIS PLACE CARDS THAT ASSIGN INITIAL VALUES TO ALL PARAMETERS
   PRESENT IN THE CONDITION FUNCTION.
*INSERT PARAMETER BOUNDARY VALUES
   BOUNDS ON THE K-TH PARAMETER CAN BE DEFINED BY THE FOLLOWING STATEMENT.

     ALB(K)= -.99E+99  $   AUB(K)= .99E+99   $  NB=NB+1  $  IB(NB)= K

   ONLY THOSE FOR PARAMETERS WITH FINITE BOUNDS NEED BE USED. THE EXAMPLE GIVING
   DEFAULT VALUES. ANY ORDER AND ANY NUMBER FROM B(1)NPAR IS PERMITTED.
   PARAMETER BOUNDARY CARDS NEED NOT BE PERMUTATED IN CASE THE PARAMETERS ARE.
   PARAMETER VALUES AND BOUNDS THAT DEPEND ON THE MAGNITUDES OF OBSERVED VALUES
   CAN BE DEFINED AFTER OBSERVED VALUES HAVE BEEN READ. AN EXAMPLE WOULD READ
*********************************************************************************
      CALL READ (1,1,NOUTRD)
      A(1)  =YOBS(1)+1.                                                      1
      ALB(1)=YOBS(1)   $   AUB(1)= .99E+99   $ NB=NB+1 $ IB(NB)= 1           2
      IREP = 0
---------------------------------------------------------------------------------
4)
   IF SECOND DERIVATIVES ARE PROGRAMMED AND USED, THE COMPLETE HESSIAN MATRIX
   G/2 = (H-H02) CAN BE APPLIED BY REDEFINING N0Z=1 (DEFAULT N0Z=0).
   APART FROM THIS USE OF SECOND DERIVATIVES THE DEFAULT VALUE N0Z=0 PERMITS
   THE USE OF THE APPROXIMATION OF THE STEP FACTOR AS FOLLOWS
*********************************************************************************
                IF(NOCLC.GE.NCYCLS) GO TO 30   $   NOCLC=NOCLC+1
      NX=20   $    IF(EPSEST.GT..5*EPS.AND.EPSEST.LT.2.*EPS) NX=3            1
      CALL MIN(DELTA,0,1,.0,.8,NX,0,0)                                       2
      CALL AISB
---------------------------------------------------------------------------------
   IF NO SECOND DERIVATIVES ARE PROGRAMMED (OR IF THEY WILL NOT BE USED OR WILL
   BE SKIPPED) THE CALL D2FDA STATEMENT HAS TO BE DELETED ONLY.
```

1.3 Subprograms in UPDATE creation mode

```
*DECK INITL
      SUBROUTINE INITL
C
      REAL M
      INTEGER END
*CALL DIMINITL
*CALL DVAR
*CALL DFIX
C
      GO TO 15
C
      ENTRY DEFAULT
C
C      IF DEFAULT VALUES ARE CHANGED CHECK OUTPUT FORMATS 12,14,16,18
C
      NOCLC=IREP=NC=NCYCLS=1
      IF(CK.EQ.A(1)) GO TO 54
      WRITE(3,55)   $   WRITE(3,56) A(1),CK   $   STOP 10
   55 FORMAT(///1H *CK.NE.A(1), CHECK COMMON COMDECKS ON PROPER ORDER*/)
   56 FORMAT(1H *A(1)=*E14.5*  ,  BUT LAST COMMON VARIABLE ON COMMON CAR
     *D NO 15 CK=A(1)=*E14.5//1H *STOP 10 (INITL)*)
   54 CONTINUE

      DO 4 I=1,NPAR
      ALB(I)=-.1E+90
    4 AUB(I)=+.1E+90
      NO2=0
      NRINSUB(4)=0   $   NRINSUB(1)=1
      NCYCLES=10  $   END=1   $   INK3=INK7=INK8=7   $   LSUMRY=1
      NOUTRD=NOUTFN=NOUTDF=NOUTD2F=NOUTFL=1   $   NDATA=5HCOUNT
      NOUTFL=5HNDATA   $   NOUTRD=0
      COSCRIT=1000.   $   EPSEST=EPS=.001   $   EQS=.66666
      R=1.   $   RN=RP=.1   $   REC1=REC2=.0
      DO 11 I=1,6
   11 IC(I)=10L
      DO 1 I=1,25
    1 CPAR(I)=.10101
      RETURN
C
C      INITIAL VALUES ARE GENERATED IN THIS PART
C
   15 CONTINUE
      MAXCYCL=0
      DO 6 I=1,25
    6 NRINSUB(I)=EXTIME(I)=0.0
      CPTIME=SECOND(ST)   $   NRINSUB(1)=NRINSUB(1)+1
      NUMBR(1)=EPMIN(1)=EPSNEW(1)=.0
      METHD=0
      NT=NPAR+1
      DO 3 I=1,MAXVEC
    3 BOTTOM(I)=TOP(I)=.0
      DO 7 I=1,15
      XBETA(I)=1.
    7 XCOSNT(I)=XCRLMT(I)=0.0
      DO 10 I=1,NPAR
      XA(1,I)=A(I)
       KFIXC(I)=0
      MA(I)=MB(I)=1.   $   DO 10 J=1,NCYCLES
      NPARTX(J)=NPART
      IPX(J,I)=1H.
      XCOSN(J,I)=.0
   10 XNA(J,I)=XALFA(J)=1.

      WRITE(3,2) NPAR,NPAR,NPART,NDATA
    2 FORMAT(/1H0*TOTAL NUMBER OF PARAMETERS     NPAR =*I4/1H *TOTAL NUMB
     *ER TO BE SOLVED      NPAR =*I4/1H *TOTAL NUMBER TO BE FITTED      NP
     *ART=*I4/1H0*TOTAL NUMBER OF DATA            NDATA=*I4//1H *NUMBER OF
     * DATA WILL BE COUNTED IN SUBROUTINE READ IF NDATA=X*/)
      WRITE(3,20)
   20 FORMAT(1H ****INITL XEQ****/)
      IF(NPART.GT.0) GO TO 118
      WRITE(3,21)   $   STOP 11
   21 FORMAT(///1H *NPART=0, IMPROVE ORDER OF PARAMETER PERMUTATION CARD
     *S IN MAIN PROGRAM NLV*//1H *STOP 11 (INITL)*)


      ENTRY HEADING
      CPTIME=SECOND(ST)   $   NRINSUB(1)=NRINSUB(1)+1
      ITEXT=1   $   GO TO 43
   44 ITEXT=0
      WRITE(3,12) NCYCLES,NDATA,EPS,IC(1),END,EPSEST,IC(2)
      WRITE(3,14) NOUTRD,IC(3),INK3,NOUTFN,EQS,IC(4),INK7,NOUTDF,IC(5)
      WRITE(3,16) INK8,NOUTD2F,RN,IC(6),NOUTFL,RP
      WRITE(3,18) LSUMRY,REC1,COSCRIT,R,REC2
      NOT=3HNOT
      WRITE(3,29) NO2 $  IF(NO2.EQ.0) WRITE(3,27) NOT
   29 FORMAT(1H+85X*MATRIX NO2*5X*USED*5X*(NO2= 0 /*I2* )*)
   27 FORMAT(1H+ 96X,A3)

   12 FORMAT( //1H127X*VALUES OF SYSTEM PARAMETERS     DEFAULT   /   USED
     * IN THIS JOB*/1H 135(*=*)//1H *NCYCLES=10  /*I4,9X*NDATA  =5HCOUNT
     *  /*I4,16X*EPS    = .001   /*F10.6,9X*IC(1)=10L(BLANK)  /*2XA10/1H
     **END    = 1  /*I4,T66*EPSEST= .001   /*F10.6,9X*IC(2)=10L*9X*/*2X
     *A10)

   14 FORMAT(1H T20*NOUTRD =0(NDATA) /*I4T101*IC(3)=10L*9X*/*2XA10/1H *I
     *NK3    = 7  /*I4,9X*NOUTFN =    1      /*I4,16X,*EQS    = .66666 /*F10
     *.6,9X*IC(4)=10L*9X*/*2XA10 /1H *INK7    = 7  /*I4,9X*NOUTDF =    1
     *    /*I4,51X*IC(5)=10L*9X*/*2XA10)

   16 FORMAT(1H *INK8    = 7  /*I4,9X*NOUTD2F=    1      /*I4,16X*RN     = .
     *1       /*F10.6,9X*IC(6)=10L*9X*/*2XA10/1H T20*NOUTFL =5HNDATA  /*I4
     *,16X*RP     = .1      /*F10.6)

   18 FORMAT(1H *LSUMRY = 1  /*I4,T101*REC1 =    0.0       /*F10.1/1H T20
     **COSCRIT= 1000.   /*E10.4,10X*R      =1.        /*F10.6,9X*REC2 =    0
     *.0      /*F10.1/1H 135(*=*)//1H0*IF NDATA = X    DATA    WILL BE*  *
     * COUNTED IN READ*/1H *IF NOUTRD= 0    NOUTRD WILL AUTOMATICALLY BE
     *EQUALLED TO MIN(NDATA,50)*/1H *IF NOUTFL= X    NOUTFL WILL AUTOMATI
     *CALLY BE EQUALLED TO MIN(NDATA,50)*)

       IF(NB.EQ.0) WRITE(3,13)
   13 FORMAT(//1H *NO BOUNDS HAVE BEEN DEFINED BEFORE CALL HEADING*/)
      IF(NB.EQ.0) GO TO 22
      WRITE(3,17)
   17 FORMAT(//1H * I*6X*NAME*10X*LOWER BOUND*6X*STARTING VALUE*5X*UPPER
     * BOUND*/1H  69(*=*)/)
      DO 19 I=1,NPAR
      WRITE(3,23) I,NAME(I),A(I)
      IF(ALB(I).GT.-.1E+90) WRITE(3,24) ALB(I)
      IF(AUB(I).LT.+.1E+90) WRITE(3,25) AUB(I)
   19 CONTINUE
   23 FORMAT(1H I2,4X,A10,22X,E13.5)
   24 FORMAT(1H+18X,E15.5,*   -*)
   25 FORMAT(1H+51X,*   -*E14.5)
   22 CONTINUE
      IF(NRINSUB(14).EQ.0) GO TO 40

      CALL BLOCK(10,0,0,.0)

      GO TO 42
   40 WRITE(3,41)
   41 FORMAT(/1H *NO BLOCKS HAVE BEEN DEFINED*/)
   42 CONTINUE
C
C
      WHEN=DATE(TA(1))  $  CLOCK=TIME(TA(2))  $  CPTIME=SECOND(TA(3))
      WRITE(3,50)  $  WRITE(3,51)  $  WRITE(3,52)
   50 FORMAT(1H1)
   51 FORMAT(1H T9*SUBJECT*T45*DATE*T60*TIME*T71*CP-SEC*T85*USED FOR*/)
   52 FORMAT(1H *NONLINEAR PARAMETER OPTIMIZATION*)
      WRITE(3,53) TA(1),TA(2),TA(3)
   53 FORMAT(1H+40X,A10,5X,A10,F10.3*    COMPILATION/LOADING*)
      WRITE(3,100) $ WRITE(3,103) IC $ WRITE(3,101) $ WRITE(3,102)
  103 FORMAT(/1H ******64X******/1H ****4X6(A10)4X****/1H *****64X*****
     **)
   43 CONTINUE
      IF(ITEXT.EQ.0) GO TO 130
      WRITE(3,197)$WRITE(3,198)$WRITE(3,198)$WRITE(3,198)$WRITE(3,200)
      WRITE(3,198)$WRITE(3,198)$WRITE(3,199)$WRITE(3,196)
  197 FORMAT(///1H 9X68(*$*))
  198 FORMAT(1H 9X*$*66X*$*)
  199 FORMAT(1H 9X68(*$*)///)
  196 FORMAT(1H 9X***66X***/1H 9X*****5X*      CONTROL PARAMETERS USED IN
     * FUNCTION, IF ANY       **X**** /1H 9X***66X***//)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
***                          HEADINGS, USE LABELS 200 T/M 300
******- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ***
*CALL XINITL
******* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
***                               PROCEED WITH SUBROUTINE
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  130 CONTINUE
      IF(ITEXT.EQ.1) GO TO 44
      WRITE(3,34) (I,I=1,NPAR)   $   WRITE(3,32) (IP(I),I=1,NPAR)
   34 FORMAT(/1H *PERMUTATION OF THE FIRST NPAR PARAMETERS*/1H0* K=*30I4
     *)
   32 FORMAT(1H  *IP=*30I4)
      DO 119 I=1,NPAR
  119 ISTAR(I)=4H****
      WRITE(3,33) (ISTAR(I),I=1,NPART)
   33 FORMAT(1H 4X,30A4)
  118 CPTIME=SECOND(FS)   $   EXTIME(1)=EXTIME(1)+FS-ST
      RETURN
      END
*DECK READ
      SUBROUTINE READ (IREADD,IWRITED,LAST1)
C
      REAL M
*CALL NDINRD
*CALL DVAR
*CALL DFIX

      CPTIME=SECOND(ST)   $   NRINSUB(2)=NRINSUB(2)+1
      IF(NDATA.EQ.5HCOUNT) NDATA=9999999
      ISTAR=1H*
      WRITE(3,30)
   30 FORMAT(1H1****XEQ READ****/)
      IF(IREADD)10,10,11
   11 CONTINUE

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
**                        READ DATA TO YOBS(I), X(I,2), X(I,3), ...
***
****                      X(I,1) IS WORKING FIELD FOR YOBS(I)              *
*****                     LABELS 300 T/M 400                              **
******- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ***
*CALL RDDATA
******* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
*****                                                                    **
***                               PROCEED WITH SUBROUTINE
***
**
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   10 NHELP=NDATA   $   IF(NDATA.GT.50) NHELP=50
      IF(NOUTFL.EQ.5HNDATA) NOUTFL=NDATA
      IF(NOUTRD.EQ.0) NOUTRD=NHELP
      IF(NOUTRD.EQ.0) NOUTRD=NDATA
      IF(NDATA.GE.NPART) GO TO 1   $   NDF=NDATA-NPART
      WRITE(3,21) NDATA,NPART,NDF   $   STOP 20
   21 FORMAT(///1H *NUMBER OF DEGREES OF FREEDOM < 0, NDATA=*I5*  NPART
     *=*I4*   D.F.=*I5//1H *STOP 20 (READ)*)
    1 CONTINUE
      DO 12 I=1,NDATA
   12 X(I,1)=YOBS(I)

      IF(IWRITED.EQ.0) GO TO 13
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
**
***                               WRITE DATA I=1,LAST1
****                                                                      *
*****                                                                    **
******- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ***
*CALL WTDATA
******* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
*****                                                                    **
***                               PROCEED WITH SUBROUTINE
***
**
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   13 CONTINUE

      WRITE(3,44)   $   WRITE(3,40)
   44 FORMAT(///1H *+*41(*=*)*+*9X*+*54(*=*)*+*)
   40 FORMAT(1H 4X* INITIAL VALUES PARAMETERS  AND NAME*32X*PERMUTED*
     *12X*NAME*/1H 70X*AND PARTIAL (*)*/)
      WRITE(3,4)   $   WRITE(3,41) (I,XA(1,I),NAME(I),I,IP(I),XA(1,IP(I)),
     *ISTAR,NAME(IP(I)),IP(I),I=1,NPART)
    4 FORMAT(1H 54X*TO BE FITTED*/)
   41 FORMAT(1H I3,2X,E14.5,6X,A10,16,I22,E18.5,2X,A2,5X,A10,I6)
      NPART1=NPART+1   $   IF(NPART1.GT.NPAR) GO TO 5
      WRITE(3,6)   $   WRITE(3,43) (I,XA(1,I),NAME(I),I,IP(I),XA(1,IP(I)),
     *NAME(IP(I)),IP(I),I=NPART1,NPAR)
   43 FORMAT(1H I3,E16.5,6X,A10,I6,I22,E18.5,9X,A10,I6)
    6 FORMAT(1H0 55X*INFORMATIVE*/)
    5 IF(NT.GT.NPAR) GO TO 7
      WRITE(3,8)   $   WRITE(3,43) (I,XA(1,I),NAME(I),I,IP(I),XA(1,IP(I)),
     *NAME(IP(I)),IP(I),I=NT,NPAR)
    8 FORMAT(1H0 56X*CONSTANTS*/)
    7 WRITE(3,45)   $   WRITE(3,42)
   45 FORMAT(1H *+*41(*=*)*+*9X*+*54(*=*)*+*)
   42 FORMAT(//1H0****READ XEQ****/)
      WRITE(3,50)
   50 FORMAT(1H+43(*=*)9X56(*=*))
      CPTIME=SECOND(FS)   $   EXTIME(2)=EXTIME(2)+FS-ST
      RETURN

      ENTRY NEWZERO
C
C             DIRECT (UNIVARIATE) SEARCH
C
C
      PROC=IREADD  $  IREPT1=IWRITED  $   IREPT2=LAST1   $   JA3=J3=INK3
C
C  NPAR=1, TEMPORARY, TO SIMPLIFY EVALUATION OF DO-LOOPS OF DERIVATIVES
C                  THAT MIGHT OCCUR IN SUBROUTINE FNCTN
```

```
C
C  DETERMINATION OF B(K) BY PROPORTIONAL IMPROVEMENTS TO A(K)
C
      NPR=NPAR   $   NPAR=1   $   EPS2=EPS
      DO 60 K=1,NPAR
   60 B(K)=XA(1,K)=A(K)

      CALL FNCTN(B,0,SQ1,0,1,YCLCA,YOCA,0,0)

      DO 61 K=1,NPART   $   IK=IP(K)
   61 CORR(IK)=A(IK)*PROC/100.
      WRITE(3,78) PROC,IREPT1,IREPT2,PROC,SQ1
C
   78 FORMAT(//1H0#APPLICATION OF RELATIVE CORRECTIONS IN DIRECT (UNIVAR
     *IATE) SEARCH BY NEWZERO(0/0 OF PAR,STEPS FOR NPART,NO OF ROUNDS)#
     */1H 76XF8.1,I12,I14//1H04X#K    IP#6X#NAME#6X#READY#4X#B#8X#FACTOR#
     *3X#STEP=#F4.0# 0/0 OF A#5X#B MIN A#8X#SQB#10X#START  WITH  SQA=#
     *E12.5/)
C
C     REPEAT PROCEDURE IREPT2 TIMES FOR ALL NPART PARAMETERS
C     ORDER OF ARGUMENTS      PROC,  IREPT1,   IREPT2
C
      DO 65 IR2=1,IREPT2   $   KX2=0   $   IR2W=IR2
C
      DO 64 K=1,NPART
      KXF=1   $   KXL=3   $   CNTR=2.   $   IK=IP(K)
C
C     REPEAT PROCEDURE AT MOST IREPT1 TIMES FOR EACH PARAMETER
C
      DO 63 IR1=1,IREPT1
      DO 66 KX=KXF,KXL
      EPS=FLOAT(KX)-CNTR
      B(IK)=A(IK)+EPS*CORR(IK)
      IF(B(IK).LE.ALB(IK)) GO TO 67
      IF(B(IK).LT.AUB(IK)) GO TO 62
   67 SQMIN(KX+10)=0.99999E+90   $   GO TO 66

   62 CALL FNCTN(B,0,SQMIN(KX+10),0,1,YCLCA,YOCA,0,0)

   66 CONTINUE
C
      SN=AMIN1(SQMIN(11),SQMIN(12),SQMIN(13))
C
      DO 68 KX=1,3
      IF(SN.EQ.SQMIN(KX+10)) GO TO 52
   68 CONTINUE
C
C     IF CENTRAL VALUE OUT OF 3 IS NOT OPTIMUM, PROCEED WITH EQUAL STEPS,
C     AT MOST IREPT1 TIMES
C
   52 IF(IR1.EQ.IREPT1) GO TO 55
      IF(KX.NE.1) GO TO 53
      SQMIN(13)=SQMIN(12)   $   SQMIN(12)=SQMIN(11)
      KXF=KXL=1   $   CNTR=CNTR+1.
      GO TO 63

   53 IF(KX.NE.+3) GO TO 55
      SQMIN(11)=SQMIN(12)   $   SQMIN(12)=SQMIN(13)
      KXF=KXL=3   $   CNTR=CNTR-1.

   63 CONTINUE
C
C     IF KX=2 THEN PARAMETER ENCLOSED AND READY
C
   55 KX2W=2H     $    IF(KX.EQ.2) KX2=KX2+1   $  IF(KX.EQ.2) KX2W=2H**
C
C     DELTA USED AS AUXILLIARY FIELD FOR SQMIN OUTPUT
C
      DELTA(K)=SQMIN(KX+10)
      K22=0  $        KN=K
      EPS=FLOAT(KX)-CNTR
      B(IK)=A(IK)+EPS*CORR(IK)
   69 BMINA=B(IK)-XA(1,IK)
      EPSALL=BMINA/CORR(IK)
      IF(J3.EQ.3.OR.IR2.EQ.IREPT2)
     *WRITE(3,73)KN,IK,NAME(IK),KX2W,B(IK),EPSALL,CORR(IK),BMINA,
     *DELTA(KN)
   73 FORMAT(1H 2I5,3X,A10,4X,A2,E12.5,F8.1,E18.5,E17.5,E13.5)
      A(IK)=B(IK)
      IF(K.LT.NPART)  GO TO 64
      IF(KN.LT.NPART) GO TO 71
      SQ2=SQMIN(KX+10)   $   SQPROC=SQ2*100./SQ1   $   SQ1=SQ2
      WRITE(3,82) IR2,SQ2
   82 FORMAT(1H+104X#END OF ROUND#I3# =#E12.5/)
   71                                  IF(KX2.NE.NPART) GO TO 64
                                       J3=3   $   KN=K22=K22+1
                                       IF(K22.GT.NPART) GO TO 64
                                       IK=IP(K22)   $   GO TO 69
   64 CONTINUE

      IF(KX2.EQ.NPART) GO TO 74
   65 CONTINUE

   74 CONTINUE
      NPAR=NPR   $   EPS=EPS2   $   INK3=JR3
      IF(NPART.LT.NPAR) WRITE(3,81)
   81 FORMAT(1H #NOT USED#/)
       NPRT1=NPART+1
      IF(NPART.LT.NPAR) WRITE(3,59)(K,IP(K),NAME(IP(K)),A(IP(K)),K=NPRT1
     *,NPAR)
   59 FORMAT(1H 2I5,3X,A10,6X,E12.5)
      WRITE(3,58) IREPT2,IREPT1,IR2W,NPART,KX2
   58 FORMAT(//1H #REPEAT PROCEDURE#I6#  TIMES#12X#REPETITIONS PER PARAM
     *ETER AT MOST#I4#  TIMES#/1H 5X#ACTUAL DONE#I6#  TIMES#//1H #NUMBER
     * OF PARAMETERS#I4/1H 10X#READY#I9# ###10X#TRANSPORT FROM B TO A AL
     *READY PROCESSED AUTOMATICALLY#)
      END
*DECK FNCTN
      SUBROUTINE FNCTN(C,LSTC,  SQ,LSTSQ,ISTOP,YCLC,YOC,LISTY,LASTI)
C
      REAL M   $   INTEGER END
*CALL DIMFNCTN
*CALL NDIMFN
*CALL DVAR
*CALL DFIX
C
      CPTIME=SECOND(ST)   $   NRINSUB(3)=NRINSUB(3)+1
      IS=1H-
      SQ=GYO=GYC=YO2=YC2=YOYC=.0
C
C          CHECK ON BOUNDS ON PARAMETER-VALUES
C
      IF(NOCLC.GT.1) GO TO 5
      IF(NB.EQ.0) GO TO 5
      ISTOPB=0
      DO 1 I=1,NB
      K=IB(I)
      IF(C(K).GE.ALB(K).AND.AUB(K).GE.C(K))GO TO 1
      WRITE(3,4) NOCLC
    4 FORMAT(/1H 12X#INITIAL PARAMETER OUTSIDE BOUNDS IN CYCLE#I3//1H
     *#STOP 30 (FNCTN)#///1H 24X8X#LOWER#17X#UPPER#/)
      WRITE(3,3) K,C(K),ALB(K),AUB(K),K,C(K)
    3 FORMAT(1H #A(#I2#)=#E12.5,10XE12.5,10XE12.5,10X#A(#I2#)=#E12.5)
      ISTOPB=1
    1 CONTINUE
      IF(ISTOPB.EQ.1) STOP 30
    5 CONTINUE
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
**                      YC=F(X(I,2),...,C(I),...) WHERE YCLC(I)=YC
***
****                    AUXILIARY VARIABLES T,U,V,W,Q,D,E,F            *
*****                   LABELS 200 T/M 1000                           **
******- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ***
*CALL NFNCTN
******* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
*****                                                                 **
***                          PROCEED WITH SUBROUTINE
***
**
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      LASTDAT=MIN0(LASTI,NDATA)
      NOUTFN=MIN0(NOUTFN,NDATA)
      NOUTDF=MIN0(NOUTDF,NDATA)
      NOUTD2F=MIN0(NOUTD2F,NDATA)
      NOUTFL=MIN0(NOUTFL,NDATA)
C
C        VARIABLE METHD USED IN SUB TRACK
C
      IF(METHD.EQ.1000) GO TO 30
      DO 41 I=1,NDATA
      YO=X(I,1)   $   YC=YCLC(I)   $   P=YO-YC
      SQ=SQ+P**2
C
      YOC(I)=P
      IF(ISTOP.EQ.1) GO TO 41
      GYO=GYO+YO   $   GYC=GYC+YC
      YO2=YO2+YO**2 $ YC2=YC2+YC**2 $ YOYC=YOYC+YO*YC
   41 CONTINUE
      IF(NOCLC.EQ.1) SQMIN(1)=SQ
      XYO=YO2   $   XYC=YC2
      IF(ISTOP.EQ.1) GO TO 30
C
      FDATA=NDATA
      VOC=SQRT(YO2*YC2)
      COSMT=YOYC/VOC
      GYO=GYO/FDATA
      GYC=GYC/FDATA
      YO2=YO2-GYO*GYO*FDATA
      YC2=YC2-GYC*GYC*FDATA
      YOYC=YOYC-GYO*GYC*FDATA
      YO2=SQRT(YO2)
      YC2=SQRT(YC2)
      RMLT=YOYC/YO2/YC2
      XCOSMT(NOCLC)=COSMT   $   XCRLMT(NOCLC)=RMLT

      IF(INK3.EQ.7) GO TO 30
      IF(LSTC.EQ.0) GO TO 10
      WRITE(INK3,110)
  110 FORMAT(1H0#***XEQ FNCTN***#/)
      WRITE(INK3,116)
  116 FORMAT(1H #  K   #8X#B(K)#8X#NAME(K)#/)
      DO 15 I=1,NPAR   $   II=IP(I)
   15 WRITE(INK3,115) II,C(II),NAME(II)
  115 FORMAT(1H I3,2X,E15.6,4X,A10)
   10 IF(LSTSQ)20,20,21
   21 WRITE(INK3,112) (IS,JS=1,85)
  112 FORMAT(//1H 85A1)
      WRITE(INK3,111) XBETA(END)
  111 FORMAT(1H027X#BETHA=#F9.5/1H0#   I#10X#YOBS#12X#X1=Y USED#10X#YCAL
     *C#12X#X1-YCALC#10X#I#/)
      LASTDAT=MIN0(LASTI,NDATA)
      IF(LASTDAT.EQ.0) GO TO 118
      WRITE(INK3,29) (I,YOBS(I),X(I,1),YCLC(I),YOC(I),I,I=1,LASTDAT)
   29 FORMAT(1H I4,4E18.6,I9)
  118 SOM=.0
      DO 120 I=1,NDATA
  120 SOM=SOM+YOC(I)
      WRITE(INK3,113) SOM,NDATA
  113 FORMAT(/1H 58X#SUM=#E14.6,I9)
      WRITE(INK3,117) (IS,JS=1,85)
  117 FORMAT(/1H085A1//)
   20 IF(LISTY)30,30,31
   31 CONTINUE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
**
***                     WRITE CALCULATED FUNCTION VALUES I=1,LASTI
****                    LABELS 2000 T/M 3000                            *
****                    NEXT TABLE IS BUILT IN, IN ROUTINE              *
****                    I  YOBS(I)  X(I,1)  YCLC(I)  YOC(I)  I, I=1,NOUTFL *
****                    I4 E18.6    E18.6   E18.6    E18.6   I9         *
****                    WITHOUT SPECIAL OUTPUT NFNCTN,2 READS  -CONTINUE- *
****                                                                   **
*****- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ***
*CALL NFNCTN
******* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
*****                                                                 **
***                          PROCEED WITH SUBROUTINE
***
**
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      WRITE(INK7,114)
  114 FORMAT(1H0#***FNCTN XEQ#D***#/)
   30 CPTIME =SECOND(FS)   $   EXTIME(3)=EXTIME(3)+FS-ST
      RETURN
      END
*DECK DFDA
      SUBROUTINE DFDA(C,M,YCLC,YOC,LSTM,LSTFA,LASTI)
C
      REAL M
*CALL DIMDFDA
*CALL NDIMDF
*CALL DVAR
*CALL DFIX
C
    4 CPTIME=SECOND(ST)   $   NRINSUB(4)=NRINSUB(4)+1
      MT=NPAR+1   $   DO 30 J=1,MT
   30 FA2(J)=.0
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * *  FA(I,K1),...,FA(I,KNPAR) WITH  X(I,2),...,C(I),...
**
***                     AUXILIARY VARIABLES T,U,V,W,Q,D,E,F             *
****                    LABELS 200 T/M 1000                            **
*****                                                                ***
******- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*CALL NDFDA
******* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
*****                                                                 **
***                          PROCEED WITH SUBROUTINE
***
**
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 42 I=1,NDATA
      FA(I,MT)=YOC(I)
      FA2(MT)=FA2(MT)+FA(I,MT)**2
      DO 42 J=1,NPAR
   42 FA2(J)=FA2(J)+FA(I,J)**2
      DO 31 J=1,NPAR
   31 M(J)=SQRT(FA2(J))
C
C          CHECK ALL SCALE FACTORS ON VALUES GREATER THAN ZERO
C
      IF(NOCLC.GT.1.OR.IREP.GT.1) GO TO 41
      KZERO=0
      NSTOP=0
      DO 40 K=1,NPAR
      IF(M(K).EQ..0) GO TO 43
      GO TO 40
```

```
43 KZERO=KZERO+1
   IF(K.LE.NPART) NSTOP=1
   IF(KZERO.EQ.1) WRITE(3,44)
44 FORMAT(//1H0#***FOLLOWING SCALE FACTORS ARE FOUND TO BE ZERO#/1H #
  *PARAMETERS PRODUCING IT SHOULD NOT OCCUR BEFORE NPAR PERMUTATION O
  *PTION IN MAIN PROGRAM NLV  ***#/1H 99X#*****#/1H0#CHECK FORMULAS OF
  * FIRST DERIVATIVES#/)
   WRITE(3,45) IP(K), NAME(IP(K)),H(K)
45 FORMAT(1H #*** A(#I3# ) , #A10,#  =#E14.8)
   IF(K.LE.NPART) WRITE(3,47)
47 FORMAT(1H+75X#****#)
48 CONTINUE

   IF(KZERO.NE.0) WRITE(3,46)
46 FORMAT(/1H 49(#*#))
   IF(NSTOP.EQ.0) GO TO 49
   WRITE(3,40)
40 FORMAT(1H+60X#STOP, BECAUSE  *** EVEN OCCURS BEFORE NPART#27X#***
  *#/1H 99X5(#*#)///1H #STOP 40 (DFDA)#)
   STOP  40
49 WRITE(3,50)
50 FORMAT(///)
41 CONTINUE

   H(NT)=SQRT(FA2(NT))
   IF(NOCLC.GT.1.AND.LAST1.NE.-1) GO TO 32
   DO 33 J=1,NPAR
33 XHA(NOCLC,IP(J))=H(J)
32 CONTINUE

   IF(LSTFA.EQ.0) GO TO 10   $   IF(INK7.EQ.7) GO TO 20
   WRITE(INK7,12)
12 FORMAT(1H0#***XEQ DFDA***#)
   NFST=1       $   NB=8
57 NLST=NPAR
   IF(NPAR.GT.NB) NLST=NB
   WRITE(INK7,52) (IP(I),I=NFST,NLST)
52 FORMAT(/1H01X#DFDA(0)#7X#DFDA(#I2#)#I9,8I14)
   WRITE(INK7,51) (NAME(IP(I)),I=NFST,NLST)   $   WRITE(INK7,50)
51 FORMAT(1H 19X,8(4X,A10))
50 FORMAT(1H )
   DO 56 I=1,LAST1
56 WRITE(INK7,53) I,FA(I,NT),(FA(I,J),J=NFST,NLST)
53 FORMAT(1H I3,2X,9E14.5)
10 IF(LSTH)20,20,21
21 WRITE(INK7,13)
13 FORMAT(/1H023X#SUM OF SQUARES AND LENGTHS OF VECTORS#)
   WRITE(INK7,52) (IP(I),I=NFST,NLST)
   WRITE(INK7,54)
54 FORMAT(1H+# NT#/)
   WRITE(INK7,53)NT,FA2(NT),(FA2(J),J=NFST,NLST)
   WRITE(INK7,53) NT,H(NT),(H(J),J=NFST,NLST)
   IF(NPAR.LE.NB) GO TO 14
   NFST=NFST+8   $   NB=NB+8      $  GO TO 57
14 WRITE(INK7,55)
55 FORMAT(1H0#***DFDA XEQ#D***#/)
20 CPTIME=SECOND(FS)   $   EXTIME(4)=EXTIME(4)+FS-ST

   RETURN


   ENTRY ORDER
   IF(NOCLC.EQ.1) WRITE(3,9)
 9 FORMAT(/1H #*** XEQ ENTRY ORDER IN SUB DFDA ***#//1H #     FIRST
  *NPAR  PARAMETERS PERMUTED ACCORDING DECREASING COSINES#/ 1H 11X#--
  *-----#//1H #THOSE .GT. MEAN OF ABS(COS) ARE SELECTED AS  NPART#
  */1H 45X#-------#/)
   COSMEAN=.0
   DO 1 K=1,NPAR
   JP(K)=IP(K)
   COSZERO=.0
   DO 2 I=1,NDATA
 2 COSZERO=COSZERO+FA(I,K)*FA(I,NT)
   COSZERO=COSZERO/H(K)/SQRT(SQA)
   ORDERED(K)=COSZERO
 1 COSMEAN=COSMEAN+ABS(COSZERO)
   COSMEAN=COSMEAN/FLOAT(NPAR)
   DO 60 K=1,NPAR
          IPERM(K)=0
   DO 61 J=1,NPAR
   IF(ABS(ORDERED(K)).LE.ABS(ORDERED(J))) IPERM(K)=IPERM(K)+1
61 CONTINUE
   IF(K.EQ.1) GO TO 65   $   K1=K-1
   DO 62 L=1,K1
   IF(IPERM(K).EQ.IPERM(L)) IPERM(K)=IPERM(L)-1
62 CONTINUE
65 IP(IPERM(K))=JP(K)
60 CONTINUE
   DO 66 K=1,NPAR
66 JP(IP(K))=K
   KP=0
   DO 63 K=1,NPAR
   IF(ABS(ORDERED(K)).GT.COSMEAN) KP=KP+1
63 CONTINUE
   NPART=KP

   IN CASE FIRST DERIVATIVES ARE PROGRAMMED IN FNCTN, CHOOSE LAST1=99999

   IF(LAST1.LT.99999) GO TO 4
   CALL FNCTN(A,0,SQA,0,1,YCLCA,YOCA,0,0)
   LAST1=NOUTDF
   GO TO 4
   END
*DECK D2FDA
   SUBROUTINE D2FDA(C,SQ,H,YCLC,YOC,LSTD2,LAST1)

   REAL H
*CALL DIMD2FDA
*CALL HDIMD2
*CALL DVAR
*CALL DFIX

   CPTIME=SECOND(ST)   $   NRINSUB(5)=NRINSUB(5)+1
   DO 31 I=1,NPAR  $  FAA2(I)=.0  $  DO 31 J=1,NPAR
31 FOFAA(I,J)=FAFAA(I,J)=.0
   WRITE(INK7,10)
10 FORMAT(1H0#***XEQ D2FDA***#/)
   KL=0
   DO 4 K=1,NPAR   $   DO 4 L=K,NPAR   $   KL=KL+1
   KV=JP(K)   $   LV=JP(L)   $   IF(KV.LE.LV) GO TO 5
   KRES=KV    $   KV=LV      $   LV=KRES
 5 LV=LV-KV
 4 IL(KL)=(KV-1)*(2*NPAR-KV)/2 + KV + LV

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                     FAA(I,K1),.......................,FAA(I,KNPAR)
*                               FAA(I,KNPAR+1),.....
*                     WITH X(I,2),....,C(1),....
**                    AUXILIARY VARIABLES T,U,V,W,Q,D,E,F               *
***                   LABELS 208 T/M 1000                              **
****- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ***
*CALL HD2FDA
***** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
***                                                                   **
*                          PROCEED WITH SUBROUTINE
*

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      CALCULATION OF THE MATRIX H02
```



```
    DO 108 I=1,NPART
    IDGL=(I-1)*(2*NPAR-I)/2+I
    FAA2(I)=.0
    DO 108 J=1,NPART
    IF(J-I)100,100,101
100 K=(J-1)*(2*NPAR-J)/2+I
    GOTO 102
101 K=(I-1)*(2*NPAR-I)/2+J
102 DO 103 L=1,NDATA
    FAFAA(I,J)=FAFAA(I,J)+FA(L,I)*FAA(L,K)
    IF(J.LT.I) GO TO 103
    FOFAA(I,J)=FOFAA(I,J)+YOC(L)*FAA(L,K)
103 CONTINUE
108 CONTINUE
    DO 120 K=1,NPART
    DO 120 L=K,NPART
120 FOFAA(L,K)=FOFAA(K,L)
    DO 104 I=1,NPAR
    IDGL=(I-1)*(2*NPAR-I)/2+I
    DO 104 L=1,NDATA
104 FAA2(I)=FAA2(I)+FAA(L,IDGL)**2
C
    INK=7
    IF(INK3.NE.3.AND.INK8.NE.3) GO TO 25   $   INK=3
    WRITE(INK,81)   $   WRITE(INK,82) (IP(I),I=1,NPART)
 82 FORMAT(1H 2X,9I14/)
 81 FORMAT(/1H #USED FOR THE DETERMINATION OF THE COURSE OF SCALE FACT
   *ORS, FAFAA(K,L)#/)
    DO 83 I=1,NPART
 83 WRITE(INK,67) IP(I),(FAFAA(I,J),J=1,NPART)
 67 FORMAT(1H I3,3X,9E14.5)
    WRITE(INK,54)
 54 FORMAT(1H )

    WRITE(INK,21)   $   WRITE(INK,82) (IP(I),I=1,NPART)
 21 FORMAT(/1H #MATRIX H02 OF 2ND DERIVATIVES#/)
    DO 23 I=1,NPART
 23 WRITE(INK,67) IP(I),(FOFAA(I,J),J=1,NPART)
    WRITE(INK,54)
 25 CONTINUE
C
C         OUTPUT SECOND DERIVATIVES FOR THE FIRST 9 PERMUTED PARAMETERS
C
    IF(LSTD2)70,70,71
 71 WRITE(INK7,72)
 72 FORMAT(1H #   I #6X#D2FDAA#9X#D2FDAB#9X#D2FDAC#/)
C
    NFAR=NPAR
    IF(NPAR.GT.9) NFAR=9
    DO 12 J=1,NFAR
    JFST=(J-1)*(2*NPAR-J)/2+J
    JLST=JFST+NFAR-J
    WRITE(INK7,75) (IP(J),IP(I),I=J,NFAR)
 75 FORMAT(1H 1X,9(10X#(#I3#,#I3#)#))
    DO 11 L=1,LAST1
 11 WRITE(INK7,13) L,(FAA(L,K),K=JFST,JLST)
    WRITE(INK7,74) IP(J),FAA2(J)
    IF(J.EQ.1) WRITE(INK7,73)
 12 CONTINUE
 73 FORMAT(1H #   I #6X#D2FDBB#/)
 13 FORMAT(1H I4,2X,9E14.5)
 74 FORMAT(/1H I4,2X,E14.5//)
C
 70 WRITE(INK8,113)
113 FORMAT(1H0#***D2FDA EXQ#D***#/)
    CPTIME=SECOND(FS)   $   EXTIME(5)=EXTIME(5)+FS-ST
    RETURN
    END
*DECK NRMEQ
    SUBROUTINE NRMEQ(YOC,CNORM,LSTNM,ISTOP,H,LSTEQ,KCOS,H02)

    REAL H
*CALL DIMNRMEQ
*CALL DVAR
*CALL DFIX

    CPTIME=SECOND(ST)   $   NRINSUB(6)=NRINSUB(6)+1
    NT=NPAR+1
    DO 6 I=1,NT
  6 CNORM(I)=.0
    IF(ISTOP.EQ.0) GO TO 17
    DO 204 I=1,NPAR
    H(I,NT)=.0
    DO 204 K=1,NDATA
204 H(I,NT)=H(I,NT)+FA(K,I)*FA(K,NT)
    H(NT,NT)=0.0
    DO 206 K=1,NDATA
206 H(NT,NT)=H(NT,NT)+FA(K,NT)**2
    DO 205 I=1,NPART
205 CNORM(I)=H(I,NT)
    CNORM(NT)=H(NT,NT)

    IF(LSTNM.EQ.0.OR.INK7.EQ.7) GO TO 90
    WRITE(INK7,504)  $  WRITE(INK7,505) (IP(I),CNORM(I),I=1,NPAR)
504 FORMAT(/1H 4X#K#6X#NORMAL #/)
505 FORMAT(1H 2I5,E17.8)
    WRITE(INK7,506)
506 FORMAT(//)
    IF(ISTOP.EQ.0) GO TO 90

 17 WRITE(INK7,5)
  5 FORMAT(1H #***NRMEQ***#/)
    DO 201 I=1,NT
    DO 201 J=1,NT
201 H(I,J)=.0
    DO 22 K=1,NDATA
    DO 21 I=1,NPAR
    H(I,NT)=H(I,NT)+FA(K,I)*FA(K,NT)
    DO 21 J=I,NPAR
 21 H(I,J)=H(I,J)+FA(K,I)*FA(K,J)
 22 H(NT,NT)=H(NT,NT)+FA(K,NT)*FA(K,NT)
    IF(H02.EQ.0) GO TO 25
    IF(NRINSUB(5).EQ.0) GO TO 25
    DO 23 I=1,NPAR
    DO 23 J=I,NPAR
 23 H(I,J)=H(I,J)-FOFAA(I,J)
 25 CONTINUE
    Q=.0
    DO 19 I=1,NPART
    H(I)=SQRT(H(I,I))
    CNORM(I)=H(I,NT)
 19 Q=Q+H(I,NT)**2
    XNRM(NOCLC)=SQRT(Q)
    IF(NPART.EQ.NPAR) GO TO 18
    IFST=NPART+1
    DO 20 I=IFST,NPAR
    H(I)=SQRT(H(I,I))
 20 Q=Q+H(I,NT)**2
 18 CNORM(NT)=H(NT,NT)
    XNORM(NOCLC)=Q=SQRT(Q)
    H(NT)=SQRT(H(NT,NT))
    IF(LSTEQ.EQ.0.OR.INK7.EQ.7) GO TO 80
C
    WRITE(INK7,30)
 30 FORMAT(/1H #NORMAL EQUATIONS#)
    NFST=1          $   N9=10
 37 NLST=NT $ IF(NT.GT.N9)NRLST=NLST=N9 $ IF(NT.LE.NLST)NRLST=NPAR
    WRITE(INK7,34) (IP(I),I=NFST,NRLST)
 34 FORMAT(1H0I11,9I13)
    DO 35 I=1,NLST  $  IPNR=IP(I)  $  IF(I.EQ.NT) IPNR=NPAR+1
 35 WRITE(INK7,36) IPNR,(H(I,J),J=NFST,NLST)
 36 FORMAT(1H I3,10E13.4)
```

```
      IF(NT.LE.N9) GO TO 38
      NFST=NFST+10 $   N9=N9+10   $   GO TO 37
   38 WRITE(INK7,32)
   32 FORMAT(/1H0*+---K*10(*-*)*H*6(*-*)*+    +---K*8(*-*)*NORM*6(*-*)
     **+   +---K*6(*-*)*PART.NORM---+*/)
      WRITE(INK7,31) (IP(I),H(I),IP(I),H(I,NT),IP(I),CNORM(I),I=1,NPAR)
   31 FORMAT(1H I5,2X,E14.5,I13,E15.5,I12,E15.5)
      WRITE(INK7,33) H(NT),XNORM(NOCLC),XNRM(NOCLC)
   33 FORMAT(1H0*LENGTHS*7X*S0*26X*NORM*21X*PART.NORM*/1H 7X,E14.5,13X,
     *E15.5,12X,E15.5)
C
   88 IF(KCOS.EQ.0) GO TO 555
      IF(LSUMRY.NE.1) GO TO 555
      I7=INK7   $   INK7=3
      IF(NOCLC.GE.NCYCLS) WRITE(INK7,45) NOCLC
   45 FORMAT(1H1,13(***)*LAST CYCLE, NO*I4*  ***//)
      IF(NOCLC.NE.1.AND.NOCLC.NE.NCYCLS) GO TO 46
      WRITE(INK7,42) NPAR
   42 FORMAT(/1H  9X*COS-MATRIX FOR *I3*  PARAMETERS AND PARTIAL COSINES
     * COS(F0,FK)*)
      NFST=1        $   N9=10
   44 NLST=NPAR
      IF(NT.GT.N9)  NLST=N9
      WRITE(INK7,43) (IP(I),I=NFST,NLST)
      WRITE(INK7,47) (NAME(IP(I)),I=NFST,NLST)
   43 FORMAT(1H0 1X,10I13)
   47 FORMAT(1H 4X,10(3X,A10))
      DO 50 J=NFST,NPAR
             LM=MIN0(J,NLST)
      DO 60 I=NFST,LM
   60 COSN(I)=H(I,J)/(H(I)*H(J))
      WRITE(INK7,51) IP(J),(COSN(K),K=NFST,LM)
   50 CONTINUE
      DO 62 I=NFST,LM
   62 COSN(I)=H(I,NT)/(H(I)*H(NT))
      WRITE(INK7,52) (COSN(K),K=NFST,LM)
   52 FORMAT(1H 3X*NORM*F9.5,9F13.5)
      DO 61 J=NFST,LM
   61 XCOSN(NOCLC,IP(J))=COSN(J)
   51 FORMAT(1H I3,10F13.5)
      IF(NT.LE.N9) GO TO 63
      NFST=NFST+10 $   N9=N9+10   $   GO TO 44
   46 CONTINUE
   63 INK7=I7   $   WRITE(INK7,110)

  555 DO 95 J=1,NPAR
   95 XCOSN(NOCLC,IP(J))=H(J,NT)/(H(J)*H(NT))

  110 FORMAT(1H0****NRMEQ XEQ*D****/)
   90 CPTIME=SECOND(FS)   $  EXTIME(6)=EXTIME(6)+FS-ST
      RETURN
      END
*DECK SOLVE
      SUBROUTINE SOLVE(C,  SOLN,NP)

      REAL M
*CALL DIMSOLVE
*CALL DVAR
*CALL DFIX

      CPTIME=SECOND(ST)    $   NRINSUB(7)=NRINSUB(7)+1
      KZERO=0
      WRITE(INK7,100)
  100 FORMAT(1H0****XEQ SOLVE****/)
      DO 8 I=1,NP   $   IF(M(I,I).NE.0.0)   GO TO 8
      KZERO=KZERO+1   $   IF(KZERO.EQ.1) WRITE(3,45)
      WRITE(3,42) I,I,IP(I),IP(I),NAME(IP(I)),NOCLC
    8 CONTINUE
   45 FORMAT(///1H *BEFORE REDUCTION OF THE MATRIX M ITS DIAGONAL CONTAI
     *NS ZEROS ON THE FOLLOWING PLACES*///1H *STOP 50 (SOLVE)*///)
   42 FORMAT(1H *M(*I2*,*I2* )= .0 THIS IS THE PERMUTED M(*I2*,*I2*)= .0
     *  PARAMETER NAME= * A10*   NOCLC= *I3)
      IF(KZERO.EQ.0) GO TO 1
C
    4 CALL LISTING(50)
C
      WRITE(3,2)
    2 FORMAT(//1H *STOP 50 (SOLVE)*)
      STOP 50
    1 DO 7 I=1,NP
    7 IPX(NOCLC,IP(I))=1HX
      NPARTX(NOCLC)=NP
      DO 3 I=1,NP   $   DO 3 J=I,NP
    3 XM(I,J)=M(I,J)
C
C         REDUCTION OF MATRIX M
C
      DO 10 I=1,NPAR
      CN(I)=M(I,NT)
      DO 11 J=1,I
   11 M(I+1,J)=.0
   10 M(I+1,I)=1.
      S=M(NT,NT)
C
C
      DO 15 I=2,NPAR
      I1=I-1
      DO 14 J=I,NT
      DO 14 K=1,I1
      IF(M(K,K).EQ.0.0.AND.K.LE.NP) GO TO 16
   14 M(I,J)=M(I,J)-M(K,I)*M(K,J)/M(K,K)
      GO TO 38
   16 WRITE(3,41) K,K,IP(K),IP(K),I,J,NOCLC
      CPTIME=SECOND(FS)   $   EXTIME(7)=EXTIME(7)+FS-ST   $  GO TO 4

   41 FORMAT(//1H *STOP 50, DURING REDUCTION OF THE MATRIX M THE ELEMENT
     * M(*I2*,*I2*)= .0.  THIS IS THE PERMUTED M(*I2*,*I2* )*/1H 8X*EXEC
     *UTION TERMINATES AT ROW, COLUMN=(*I2*,*I2*)*/1H 8X*NOCLC=*I3)
   38 CONTINUE
C
      DO 15 J=1,I1
      DO 15 K=J,I1
   15 M(I+1,J)=M(I+1,J)-M(K,I)*M(K+1,J)/M(K,K)
C
C-----
C         SAVE DIAGONAL
C
      DO 39 I=1,NPAR
   39 D(I)=M(I,I)
C
C         SOLUTION
C
      DO 17 I=1,NPAR
   17 M(I,1)=M(1,NT)*M(I+1,I)/D(I)
      SQ=M(NT,NT)
      N1=NPAR-1
      DO 40 I=1,N1
      I1=I+1
      DO 40 J=I1,NPAR
   40 M(I,J)=M(I,J-1)+M(J,NT)*M(J+1,I)/D(J)
C
C         REDUCTION OF SQA
C
      M(1,NT)=M(NT,NT)-M(1,NT)*M(1,NT)/D(1)
      DO 18 K=2,NPAR
   18 M(K,NT)=M(K-1,NT)-M(K,NT)*M(K,NT)/D(K)
C
C         INVERSE MATRIX, SAVED IN XM
C
      Q=.0
      DO 19 I=1,NP
      DO 19 J=I,NP
      DO 20 K=J,NP
   20 Q=Q+M(K+1,I)*M(K+1,J)/D(K)
      XM(J+1,I)=M(J+1,I)=Q
   19 Q=.0
C
C         OUTPUT PARTIAL SOLUTIONS AND TOTAL SOLUTION
C
      IF(INK7.EQ.7) GO TO 71
      WRITE(INK7,53)
   53 FORMAT(/1H 9X*SOLUTION*12X*SOF*11X*DECREASE*/)
      WRITE(INK7,30) S
   30 FORMAT(1H 20X,E17.8/)
      Q=S-M(1,NT)
      DO 21 I=1,NPAR
      DO 22 J=1,I
   22 WRITE(INK7,31) IP(J),M(J,I)
   31 FORMAT(1H I3,E17.8)
      WRITE(INK7,32) M(I,NT),Q
   32 FORMAT(1H+,20X,2E17.8//)
   21 Q=M(I,NT)-M(I+1,NT)
C
C         WRITE INVERSE MATRIX
C
   71 IF(INK7.EQ.7) GO TO 78
      WRITE(3,52)
   52 FORMAT(/1H *INVERSE MATRIX*)
      NFST=1        $   N8=8
   70 NLST=NP
      IF(NP.GT.N8)    NLST=N8
      WRITE(3,55) (IP(I),I=NFST,NLST)
   55 FORMAT(1H0I13,8I14)
      DO 23 I=NFST,NP
             LM=MIN0(I,NLST)
   23 WRITE(3,33) IP(I),(M(I+1,J),J=NFST,LM)
      IF(NP.LE.N8) GO TO 69
      NFST=NFST+8  $  N8=N8+8      $  GO TO 70
   69 CONTINUE
   33 FORMAT(1H I3,2X,8E14.5)
C
C         COS-MATRIX OF INVERSE, NOT SAVED
C
   78 IF(NOCLC.NE.1.AND.NOCLC.LT.NCYCLS) GO TO 79
      IF(LSUMRY.NE.1) GO TO 79
      WRITE(3,82) NPART
   82 FORMAT(////1H *CORRELATION MATRIX OF ESTIMATES, OBTAINED FROM THE
     *INVERSE OF M, FOR NPART=*I3)
      NFST=1   $   N8=10
   80 NLST=NP
      IF(NP.GT.N8)    NLST=N8
      WRITE(3,59) (IP(I),I=NFST,NLST)
      WRITE(3,58) (NAME(IP(I)),I=NFST,NLST)
   58 FORMAT(1H 4X,10(3X,A10))
   59 FORMAT(1H0 1X,10I13)
      DO 84 I=NFST,NP
             LM=MIN0(I,NLST)
      DO 83 J=NFST,LM
   83 COSINF(J)=M(I+1,J)/(SQRT(M(J+1,J)*M(I+1,I)))
      WRITE(3,81) IP(I),(COSINF(J),J=NFST,LM)
   84 CONTINUE
   81 FORMAT(1H I3,10F13.5)
      WRITE(3,46) (M(K+1,K),K=NFST,LM)
   46 FORMAT(/1H *DIAGONAL* E12.5,8E13.5,E12.4)
      IF(NP.LE.N8) GO TO 72
      NFST=NFST+10   $   N8=N8+10   $   GO TO 80
C
C      OUTPUT LINEAR REDUCTION WITH RESPECT TO PARAMETERS USED IN SOLUTION
C
   72 WRITE(3,54)    $   WRITE(3,76)
   76 FORMAT(1H 12X*LINEAR REDUCTION OF SQ0 BY APPLICATION OF DELTA CORR
     *ECTIONS TO L PARAMETERS*/1H0*  K*10X*A*11X*REDUCED SQ0*7X*REDUCTIO
     *N*7X*NAME*7X*K*4X*L*6X*REDUCTION/DELTA*/)
      WRITE(3,85) S
   85 FORMAT(1H 20X,E17.8)
      Q=S-M(1,NT)
      DO 73 K=1,NPART   $   IK=IP(K)
      REDDELT=Q/M(K,NP)
      WRITE(3,74) K,A(IK),M(K,NT),Q,NAME(IK),IK,K,REDDELT
   73 Q=M(K,NT)-M(K+1,NT)
   74 FORMAT(1H I3,3E17.8,2X,A10,2I5,E18.5)
      IF(NPART.GE.NPAR) GO TO 79   $   NP1=NPART+1
      WRITE(3,77)
   77 FORMAT(1H )
      Q=M(NPART,NT)-M(NPART+1,NT)
      DO 75 K=NP1,NPAR   $   IK=IP(K)
      REDDELT=Q/M(K,NP)
      WRITE(3,74) IK,A(IK),M(K,NT),Q,NAME(IK),IK,K,REDDELT
   75 Q=M(K,NT)-M(K+1,NT)

   79 IF(NRINSUB(5).EQ.0) GO TO 65
C
C
C          CALCULATION FOR COURSE OF SCALE FACTORS
C
      DO 60 I=1,NP
      TOT=.0
      DO 61 J=1,NP
   61 TOT=TOT+FAFAA(I,J)*M(J,NP)
   60 FFAAMM(I)=TOT/(HA(I)*HA(I))
C
C          SOLUTION BY INVERSE MATRIX
C
   65 IF(INK3.EQ.7) GO TO 66
      DO 68 I=1,NPAR
   68 OQ(I)=.0
      DO 62 I=1,NP
      DO 62 J=1,NP
      IF(I-J) 63,63,64
   64 OQ(I)=OQ(I)+M(I+1,J)*CN(J)
      GOTO 62
   63 OQ(I)=OQ(I)+M(J+1,I)*CN(J)
   62 CONTINUE
C
C          OUTPUT PARTIAL SOLUTIONS
C
   66 DO 50 I=1,NPAR
      E(I)=SQRT(SQ*M(I+1,I)/FLOAT(NDATA-NP))
   50 STDEV(I)=SOLN(I)=.0
C
C          TOTAL AND PARTIAL DELTA (XDELTA, XDELT)
C
      QLT=.0
      DO 49 I=1,NPAR
   49 QLT=QLT+M(I,NPAR)**2
      XDELTA(NOCLC)=SQRT(QLT)
C
      QL=.0
      DO 51 I=1,NP
      STDEV(I)=E(I)    $   SOLN(I)=M(I,NP)
   51 QL=QL+SOLN(I)**2
      XDELT(NOCLC)=SQRT(QL)
      IF(INK3.EQ.7) GO TO 67
      WRITE(INK3,54)
   54 FORMAT(//)
      WRITE(INK3,56)
   56 FORMAT(/1H 2X*K*8X*NORMAL *9X*DELTA-VECTOR*10X*A*12X*STAND-DEV.*3X
     ** K     M(INV)*NORM*/)
      WRITE(INK3,57)(IP(I),ANORM(I),SOLN(I),C(IP(I)),STDEV(I),IP(I),OQ(I
     *),I=1,NPAR)
   57 FORMAT(1H I3,2X,4E17.8,I5,E17.8)
      WRITE(INK3,43) XNRM(NOCLC), XDELT(NOCLC)
   43 FORMAT(1H0*LENGTH*E16.8,E17.8)
      WRITE(INK3,44) XNORM(NOCLC),XDELTA(NOCLC)
   44 FORMAT(1H *TOTAL *E16.8,E17.8)
      WRITE(INK3,101) NP
```

```
101 FORMAT(//1H ***SOLVE AE**D***FOR   NPART=*I4*  PARAMETERS****/)
 67 CPTIME=SECOND(FS)   $   EXTIME(7)=EXTIME(7)+FS-ST
    RETURN
    END
ECK NOWA
    SUBROUTINE NOWA, RETURNS(A44)

    REAL M   $   INTEGER END
    DIMENSION CHI1(14),FN(14),KLAS(14),FCLSS(14)
    DIMENSION SK(35),SN(15),SKU(15),SKL(15)
ALL DIMNOWA
ALL DVAR
ALL DFIX

    CPTIME=SECOND(ST)   $   NRINSUB( 6)=NRINSUB( 6)+1
    IF(NOCLC.LT.3)            GO TO 15
    IF(EPMIN(NOCLC ).NE..0)   GO TO 15
    NOCLC1=NOCLC-1
    IF(EPMIN(NOCLC1).NE..0)   GO TO 15
    NOCLC2=NOCLC-2
    IF(EPMIN(NOCLC2).NE..0)   GO TO 15
    WRITE(3,17) NOCLC2,NOCLC1,NOCLC,NOCLC
 17 FORMAT(///1H *NOTE THAT EPS(*I2*), EPS(*I2*) AND EPS(*I2*) ALL ARE
   * ZERO*//1H *DETECTED IN SUB NOWA IN CYCLE*I4/1H *CORRECTION VECTOR
   * PROBABLY A ZERO VECTOR, OR*/1H *PARAMETER VALUES TOO CLOSE TO BOU
   *NDS*//1H *THIS CAUSES STOP 1 TERMINATION IN MAIN PROGRAM NLV*//)
    NCYCLS=NOCLC   $   END=IREP
 15 CONTINUE

    SQ=.0
    SQF=.0
    SQAF=.0
    DO 2 I=1,NDATA
    FCALC(I)=.0
    DO 3 J=1,NPART
  3 FCALC(I)=FCALC(I)+DELTA(J)*FA(I,J)
    SQ=SQ+YOCA(I)**2
    SQAF=SQAF+FCALC(I)**2
    FOC(I)=YOCA(I)-FCALC(I)
    SQF=SQF+FOC(I)**2
    QCOSA=SQ - SQF   $   IF(QCOSA.LT.0.0) QCOSA=.0
    QCOSA=SQRT(QCOSA/SQ)
    XCOSQ(NOCLC)=QCOSA
  2 CONTINUE

    XSQF(NOCLC)=SQF
    HSQF=SQRT(SQF)

       CHECK DIRECTION OF PERPENDICULAR TO TANGENT PLANE

    IF(NOCLC.NE.1.AND.NOCLC.LT.NCYCLS.AND.NRINSUB(5).EQ.0) GO TO 7

    DO 21 I=1,NPAR
    E(I)=.0
    D(I)=.0
    DO 20 L=1,NDATA
    E(I)=E(I)+YOCA(L)*FA(L,I)
 20 D(I)=D(I)+FOC(L)*FA(L,I)
    E(I)=E(I)/SQRT(SQ)/MA(I)
 21 D(I)=D(I)/HSQF/MA(I)
    TEST=SQ-SQF

    IF(NOCLC.NE.1.AND.NOCLC.LT.NCYCLS) GO TO 63
    IF(NOCLC.EQ.1.AND.IREP.GT.1.AND.LSUMRY.NE.1) GO TO 63

    WRITE(3,40) NOCLC,XBETA(END),IREP,XCRLAT(NOCLC),END
 40 FORMAT(1H1134(*=*)/1H ***= NOW A *** AFTER CYCLE NO*I4*  ****  8X*X
   *(I,1)* YA(I) **F8.5* * YOC(I)*8X*NOWA NO**I3*  *****8X*MULT.CORR.**
   *F9.6*  ****/)1H 85X**** END=*I3*  ****/)
    DO 5 I=1,NPAR
  5 ISTAR(I)=1H
    ISTAR(NPAR-NPART+1)=1H*
    NFST=1          $      N14=14
 46 NLST=NPAR       $      IF(NPAR.GT.N14) NLST=N14
    NNN=NPAR-N14+14   $   WRITE(3,47)
 47 FORMAT(1H ***********/1H *I  SUMMARY OF SOLUTIONS*
   */1H *I  SOLVED***)
    WRITE(3,45)   (ISTAR(I),I=NFST,NLST)
 45 FORMAT(1H ***********4X,(3A9,A2)
    DO 49 I=1,NNN
          LM=MIN0(14,(NNN+1-I))
    WRITE(3,44)      IP(I),(M(I,NNN+1-J),J=1,LM)
 49 CONTINUE
 44 FORMAT(1H *I   *I4,2X,14(1X,E8.2))
    WRITE(3,48)
 48 FORMAT(1H ***********)
    IF(NPAR.LE.N14) GO TO 41
    NFST=NFST+14   $      N14=N14+14    $ GO TO 46
 41 CONTINUE
    RTXYO=SQRT(XYO)   $   RTSQA=SQRT(SQ)   $   RTXYC=SQRT(XYC)
    COSOR=(XYO+XYC-SQ)/(2.*RTXYO*RTXYC)
    SERROR=SQRT(SQ/FLOAT(NDATA-NPART))
    WRITE(3,50)   $   WRITE(3,51) RTXYO,SQ,SQF,COSOR,RTSQA,RTXYC
 50 FORMAT(//1H 27X*RESULT OF SOLVING NAME$*T98*LENGTH OF VECTORS*/)
 51 FORMAT(1H 30X*YOBS*/1H 39X***T09*/YOBS/=*E11.5/1H 38X****
   */1H 37X** **/1H+20X*SQA=*E11.5* *  * SQF=*E11.5,T74*MULT.COSINE =*
   *F7.5+6X*/SQA/=*E11.5/1H 35X**    **/)H 34X**     **/1H 27X,19(***)
   ** TANGENT PLANE*T88*/YCALC/=*E11.5/1H 33X*A      F*)
    WRITE(3,52) SQAF,TEST,QCOSA,SERROR,SQ,NOCLC,NPARTX(NOCLC)
 52 FORMAT(1H 27X*SQA/=*E11.5/1H 23X*FCALC**2=*E11.5//1H 18X*TOTAL COS
   *INE =*F7.5,T05*STANDARD ERROR=* E16.10/1H T90*SQA       =*E16.10/1H
   * T86*NPARTX(*I3* ) =*I4/)

 63 COSDN=COSD2=COSN2=.0
    DO 62 I=1,NPART
    IF(DELTA(I).EQ.0.0) GO TO 62
    COSDN=COSDN+ANORM(I)*DELTA(I)
    COSN2=COSN2+ANORM(I)**2
    COSD2=COSD2+DELTA(I)**2
 62 CONTINUE
    TANABS=-2.*COSDN
    COSDN=COSDN/SQRT(COSD2)/SQRT(COSN2)
    IF(NO2.EQ.0) GO TO 59   $   EP=.0   $   GO TO 58
 59 CONTINUE
    QRS=QRSN=.0   $   IF(NRINSUB(5).EQ.0) GO TO 53
    EP=.0
    DO 54 L=1,NPART
    ED=.0
    DO 55 K=1,NPART
 55 ED=ED+DELTA(K)*FDFAA(K,L)
 54 EP=EP+ED*DELTA(L)
 58 QRSA=ABS(.5*TANABS)   $   QRS=QRSA/TEST   $   QRSN=QRSA/(TEST-EP)
    EPSNEW(NOCLC+1)=EPSEST*QRSN
    IF(NOCLC.NE.1.AND.NOCLC.LT.NCYCLS) GO TO 7
 53 IF(NOCLC.EQ.1.AND.IREP.GT.1.AND.LSUMRY.NE.1) GO TO 7
    WRITE(3,134) COSDN,TANABS,QRSN,NO2,QRS

134 FORMAT(1H0*****  COS(NORMAL,DELTA) FOR NPART =*F11.7*   ****   TAN(
   *EPS=0,SQA,1) =*E14.5*   **** EPS(PREDICTED) =*E16.7*  *****/)H
   *90X*****  (NO2 =*I2* GIVES =*E16.7*) *****//1H 32X*PARTIAL   PARTIA
   *L*)

    WRITE(3,32)
 32 FORMAT(1H 2X*K*10X*A(K)*8X*USED*4X*COS A*5X*COS F*6X*DELTA-VECTOR*
   *8X*NORMAL *8X*MA*DELTA*6X*MA*NORM*8X*MA(K)*7X*K*/)

    DO 33 I=1,NPAR
    DELMA=DELTA(I)*MA(I)   $   ANRMMA=ANORM(I)*MA(I)
    WRITE(3,34) IP(I),A(IP(I)),E(I),D(I),DELTA(I),ANORM(I),DELMA,
   *ANRMMA,MA(I),IP(I)
    IF(I.LE.NPART) WRITE(3,38)
 33 CONTINUE
```



```
 38 FORMAT(1H+26X***)
 34 FORMAT(1H I3,E20.10,5X,2F10.5,2E18.8,3E14.5,I5)
C
    IF(NRINSUB(5).EQ.0) GO TO 16
C
C   CURVATURE IN DIRECTION OF PARAMETRIC CURVES
C
    IF(LSUMRY.LE.1) GO TO 16
    DO 180 KJ=1,NPART
    FSS2=.0
    JDGL=(KJ-1)*(2*NPAR-KJ)/2+KJ
    DO 181 I=1,NDATA
    FSS(I)=FAA(I,JDGL)*MA(KJ)-FA(I,KJ)*FAFAA(KJ,KJ)/MA(KJ)
    FSS(I)=FSS(I)/(MA(KJ)**3)
181 FSS2=FSS2+FSS(I)*FSS(I)
    CURVTOT(KJ)=SQRT(FSS2)
C
C   GEODESIC CURVATURE
C
    DO 182 K=1,NPART
    TJ=.0
    DO 183 I=1,NDATA
183 TJ=TJ+FA(I,K)*FSS(I)
182 TJFSS(K)=TJ
    DO 195 K=1,NPART
    DC=.0
    DO 192 J=1,NPART
    IF(K-J) 193,193,194
194 DC=DC+XM(K+1,J)*TJFSS(J)
    GO TO 192
193 DC=DC+XM(J+1,K)*TJFSS(J)
192 CONTINUE
    DCURVE(K)=DC
195 CONTINUE
    FSSG2=.0
    DO 184 I=1,NDATA
    FJD=.0
    DO 185 K=1,NPART
185 FJD=FJD+FA(I,K)*DCURVE(K)
    FSSG2=FSSG2+FJD*FJD
184 CONTINUE
    CURVGEO(KJ)=SQRT(FSSG2)
180 CONTINUE
    WRITE(3,75)RTSQA   $   WRITE(3,76)          $   WRITE(3,77)
 75 FORMAT( ///1H *ANALYSIS OF CURVATURE OF PARAMETRIC CURVES IN ABSOL
   *UTE AND IN RELATIVE SENSE (WITH RESPECT TO /SQA/),   /SQA/=*E16.5
   */1H 88X*+*/)
 76 FORMAT(1H *+*15(*-*)*ABSOLUTE  VALUES*15(*-*)*+*16X*+*15(*-*)*RELA
   *TIVE VALUES*15(*-*)*+*)
 77 FORMAT(1H *  K*6X*TOTAL*8X*GEODESIC*8X*REST*10X*NAME*8X*  K*6X
   **TOTAL*8X*GEODESIC*8X*REST*/)
    DO 78 K=1,NPART
    CVT=CURVTOT(K)   $   CVG=CURVGEO(K)   $   CVR=SQRT(CVT*CVT-CVG*CVG)
    CVTS=CVT*RTSQA   $   CVGS=CVG*RTSQA   $   CVRS=CVR*RTSQA
    IPK=IP(K)
 78 WRITE(3,79) IPK,CVT,CVG,CVR,NAME(IPK),IPK,CVTS,CVGS,CVRS
 79 FORMAT(1H I3,3E14.5,6X,A10,I6,3E14.5)
    WRITE(3,207)   $   WRITE(3,207)

 16 IF(NRINSUB( 9).EQ.0.AND.NRINSUB(5).EQ.0) GO TO 6
C
    IF(LSUMRY.LE.1) GO TO 6
    WRITE(3,201)
    IF(NC.NE.1) WRITE(3,202)   $   IF(NRINSUB(5).NE.0) WRITE(3,203)
    WRITE(3,204)
    IF(NC.NE.1) WRITE(3,205)   $   IF(NRINSUB(5).NE.0) WRITE(3,206)
    WRITE(3,207)
    DO 211 I=1,NPAR   $   WRITE(3,208) IP(I),MA(I)
    IF(NC.NE.1) WRITE(3,209) XMA(NOCLC-1,IP(I)),MAMB(I)
    IF(NRINSUB(5).NE.0.AND.I.LE.NPART) WRITE(3,210) FFAAMM(I),IP(I)
211 CONTINUE
201 FORMAT(/1H0*COURSE OF SCALE FACTORS *)
202 FORMAT(1H+25X*+--ACCORDING TO DIFFERENCES--+*)
203 FORMAT(1H+60X*+--ACCORDING TO DIFFERENTIALS--+*)
204 FORMAT(1H0*  K*10X*MB*)
205 FORMAT(1H+32X*MA*8X*(MB-MA)/MA*)
206 FORMAT(1H+67X*D(MA)DA/MA*9X*K*)
207 FORMAT(1H )
208 FORMAT(1H I3,E16.5)
209 FORMAT(1H+19X,E19.5,E14.5)
210 FORMAT(1H+62X,E15.5,I10)
  6 CONTINUE
C
    WRITE(3,43) NOCLC
 43 FORMAT(/1H0****END OF CYCLE NO*I4*  ****)
    WRITE(3,42)
 42 FORMAT(1H 134(***)//)

  7 CONTINUE

    IF(MAXCYCL.EQ.1000) NYS2=4H
    IF(NYS2.EQ.4H  **) GO TO 67   $   MAXCYCL=0
 67 CC=COSCRIT*QCOSA
    NYS1=4H  **   $ NOYES1=4H  **   $  NYS2=4H        $  NOYES2=4H
    NYS3=4H  **   $   NOYES3=4H
              IF(IFIX(CC).NE.0) GO TO 64
    NYS3=4H       $   NOYES3=4H  **
 64 DO 60 ICT=1,NPART   $   CC=COSCRIT*XCOSN(NOCLC,IP(ICT))
              IF(IFIX(CC).NE.0) GO TO 66
 60 CONTINUE
    NYS1=4H       $   NYS2=4H  **
              IF(NRINSUB(14).GT.0) GO TO 68
              IF(NC.LT.NCYCLS) NCYCLS=NC+1
 68           CONTINUE
    N1=NPART+1   $   IF(N1.GT.NPAR) GO TO 65
    DO 61 ICT=N1,NPAR   $   CC=COSCRIT*XCOSN(NOCLC,IP(ICT))
              IF(IFIX(CC).NE.0) GO TO 66
 61 CONTINUE
 65 NOYES1=4H       $   NOYES2=4H  **
 66 CPTIME=SECOND(FS)   $   EXTIME( 8)=EXTIME( 8)+FS-ST
    RETURN


    ENTRY SUMRY
C
C  CALCULATION OF CONFIDENCE INTERVALS ON F(B), CALCULATION OF T-VALUES
C
    FND=FLOAT(NDATA)
    IF(LSUMRY.NE.1) GO TO 110
    CPTIME=SECOND(ST)   $   NRINSUB(12)=NRINSUB(12)+1
    I3=INK3  $  I7=INK7  $  INK3=3  $  INK7=3
    WRITE(3,90)   $   WRITE(3,91) IREP,END
 90 FORMAT(1H1,14(***)5X,6(***))
 91 FORMAT(1H *SUMMARY NO*I4      ,5X*END=*I2//)
    WRITE(3,104)
104 FORMAT(//1H 9X*INITIAL VALUE*6X*DIFFERENCE*8X*FINAL VALUE*8X*NAME*
   *11X*       B/A*8X*CONTROLLED*/1H *  K*11X*A(K)*11X*B(K)-A(K)*
   *12X*B(K)*22X*K*36X*K*/)
    DO 105 IK=1,NPAR  $  IJ=IP(IK)  $  BIJAIJ=A(IJ)-XA(1,IJ)
    BP=.0   $   IF(XA(1,IJ).EQ..0) GO TO 105
    BP=A(IJ)/XA(1,IJ)
    I3H=3H     $   IF(KFIXC(IJ).GT.0) I3H=3H***
105 WRITE(3,106)IJ,XA(1,IJ),BIJAIJ,A(IJ),NAME(IJ),IJ,BP,I3H,KFIXC(IJ),
   *IJ
106 FORMAT(1H I3,3E18.6,5X,A10,I5,E16.6,9X,A3,I3,I6)
    WRITE(3,107) NDATA, NPART
107 FORMAT(//1H0*TOTAL NUMBER OF USED DATA*9X*NDATA=*I4/1H *TOTAL NUMB
   *ER OF FITTED PARAMETERS NPART=*I4)
C
    WRITE(3,111)
111 FORMAT(/1H039X*FUNCTION EVALUATION WITH FINAL PARAMETER VALUES B(K
   *)*)
```

```
      DO 80 K=1,14
   80 KLAS(K)=0
      IDOT=1H-     $     JDOT=1H=
      WRITE(3,170) (JDOT,MI=1,136)
      WRITE(3,169)    $   WRITE(3,170) (IDOT,MI=1,136)
  169 FORMAT(1H #ANALYSIS OF OBSERVED AND CALCULATED FUNCTION VALUES
     *M.E.=SUM/NDATA     S.E.=STANDARD ERROR     X1=YA+BETHA*YOC     V=X1-Y
     *CALC   T=V/S.E.#)
  170 FORMAT(1H 136A1/)
      WRITE(3,171)XBETA(END),(IDOT,MI=1,26),(IDOT,MI=1,20),(IDOT,MI=1,20
     *),(IDOT,MI=1,20)
  171 FORMAT(1H 14X#BETHA=#F8.4,12X#95 0/0 CONFIDENCE INTERVAL FOR F(B)#
     *24X#X1-YCALC  ANALYSIS#/1H 5X#+#26A1#+    +#20A1#+#20A1#+#4X
     *#REMAINDER#5X#+#20A1#+#)

      WRITE(3,172)
  172 FORMAT(1H 3X#I#7X# X2 #7X#X1= Y USED#5X#Y LOWER BOUND#5X#YCALC#5X
     *#Y UPPER BOUND#+X#              #10X#V#12X#T#5X#S.E(F(B))#3X#I#/)

      SUMPLUS=SUMMIN=.0
      NDF=DF=NDATA-NPARTX(NOCLC)   $   VAREST=SQMIN(NOCLC)/DF
      SERROR1=SQRT(VAREST)    $    SEVMEAN=SQRT(VAREST/FND)
      INTVL=0
      S1=S2=S3=S4=.0


      DO 160 I=1,NDATA
      IGT=3H
      SIGMAF=.0
      EXPF=.0
      DO 162 L=1,NPART
      OPE=.0
      DO 165 K=1,NPART
      IF(K.LT.L) GO TO 163
      OPE=OPE+FA(I,K)*M(K+1,L)
      GO TO 165
  163 OPE=OPE+FA(I,K)*M(L+1,K)
  165 CONTINUE
      SIGMAF=SIGMAF+OPE*FA(I,L)
      IF(NRINSUB(5).EQ.0) GO TO 662
C
C     EXPF STANDS FOR F OF EXPECTATION B
C
      DO 600 J=1,NPART
      IF(L-J) 634,634,633
  633 K=(J-1)*(2*MPAR-J)/2+L        $  GO TO 694
  634 K=(L-1)*(2*MPAR-L)/2+J
      EXPF=EXPF+FAA(I,K)*M(J+1,L)   $  GO TO 600
  694 EXPF=EXPF+FAA(I,K)*M(L+1,J)
  600 CONTINUE
  662 CONTINUE
  162 CONTINUE
      IF(SIGMAF.LE..0)SIGMAF=.0
      SIGMAF=SQRT(SIGMAF*VAREST)
      X1=X(I,1)    $    YOC=YOCA(I)
      YMEAN=YCLCA(I)
      YLOWER=YMEAN-2.*SIGMAF
      YUPPER=YMEAN+2.*SIGMAF
      EXPF    =0.5*VAREST*EXPF
      IF(NRINSUB(5).EQ.0) EXPF=.0
      IF(X1.GT.YUPPER)IGT=3H>UB
      IF(X1.LT.YLOWER)IGT=3H<LB
      IF(IGT.NE.3H    )INTVL=INTVL+1
      TVALUE=YOC/SERROR1
C
      IF(NYS2.NE.4H  **) GO TO 85

      S1=S1+TVALUE    $    TV2=TVALUE*TVALUE
      S2=S2+TV2       $    TV3=TVALUE*TV2
      S3=S3+TV3       $    TV4=TV2*TV2
      S4=S4+TV4

C
C
C     DETERMINATION OF EMPERICAL FREQUENCIES OF T
C
      IF(TVALUE.LT.0) GO TO 82

      DO 81 ICLSS=1,6
      FTEST=FLOAT(ICLSS)/2.
      IF(TVALUE.GT.FTEST) GO TO 81
      KLAS(7+ICLSS)=KLAS(7+ICLSS)+1
      GO TO 85
   81 CONTINUE
      KLAS(14)=KLAS(14)+1    $    GO TO 85

   82 DO 83 ICLSS=1,6
      FTEST=-FLOAT(ICLSS)/2.
      IF(TVALUE.LT.FTEST) GO TO 83
      KLAS(8-ICLSS)=KLAS(8-ICLSS)+1
      GO TO 85
   83 CONTINUE
      KLAS(1)=KLAS(1)+1

   85 CONTINUE
      IF(I.LE.NOUTFL)WRITE(3,173)I,X(I,2),X1,IGT,YLOWER,YMEAN,YUPPER,
     *EXPF,YOC,TVALUE,SIGMAF,I
  173 FORMAT(1H I4,2E14.5,A3,E13.5,2E14.5,E16.3,E17.5,F9.2,E13.4,I4)

      IF(YOC.LT..0) SUMMIN=SUMMIN+YOC
      IF(YOC.GE..0) SUMPLUS=SUMPLUS+YOC
  160 CONTINUE

      PINTVL=FLOAT(INTVL)/FLOAT(NDATA)*100.
      SUMTOT=SUMMIN+SUMPLUS    $    VMEAN=SUMTOT/FND
      SUMABS=(SUMPLUS-SUMMIN)/FND
      WRITE(3,174)NDATA,INTVL,PINTVL,SUMPLUS,NPART,SUMMIN,NDF,SUMTOT,
     *(IDOT,MI=1,68),SERROR1
  174 FORMAT(/1H 8X#NDATA=#I4,15X#+----#I4# (=#F5.1# 0/0) DATA OUTSIDE IN
     *TERVAL#T96#+ =#E12.5/1H 8X#NPART=#I4T96#- =#E12.5/1H 8X#D.F. =#I4T
     *95#SUM=#E12.5/1H #+#68A1#+#T117#S.E.=#E13.6T74#TO BE NEAR ZERO#)
      TVALUE=VMEAN/SEVMEAN
      WRITE(3,177) VMEAN,SEVMEAN,TVALUE,SUMABS
  177 FORMAT(1H # M.E.=#E12.5,6X#S.E.(M.E.)=#E12.5,5X#  RATIO=#F9.4,
     *#  <---------------  MEAN/V/=#E12.5)
      WRITE(3,170) (JDOT,MI=1,136)
      WRITE(3,178)
  178 FORMAT(/1H 88X#ESTIMATE      F(B)   = YCALC#)
      IF(NO2.EQ.0) WRITE(3,175)
      IF(NO2.EQ.1) WRITE(3,176)
  175 FORMAT(1H #SECOND DERIVATIVE MATRIX NO2 NOT USED IN B/2=(N-NO2)#)
  176 FORMAT(1H #USE IS MADE OF SECOND DERIVATIVE MATRIX B/2=(N-NO2)#)

      WRITE(3,676)
  676 FORMAT(1H+88X#2ND ORDER TERM       =            FAA*INV(M)*VAR/2#
     */1H 108X#=               REMAINDER#
     *        /1H 88X#STAND.ERROR   F(B)   = SQRT(FA*INV(M)*FA*VAR)#)
C
      IF(NYS2.NE.4H  **) GO TO 88
C
C     TESTS ON NORMALITY OF T
C     TEST ESTIMATES OF MOMENTS
C     IN 2ND AND HIGHER MOMENTS M1 IS TAKEN EQUAL TO ITS TRUE VALUE 0
C     IN 3RD AND HIGHER MOMENTS M2 IS TAKEN EQUAL TO ITS TRUE VALUE 1
C
      FND1=DF       $   FND2=FND1-1.    $   FND3=FND2-1.
      IF(FND3.LE..05) WRITE(3,310) NDATA, NPART,FND3
  310 FORMAT(/1H #NDATA=#I4# ,   NPART=#I4# , DEGREES OF FREEDOM FOR 3RD
     * AND 4TH MOMENT =#I4/1H #NO DETERMINATION OF HIGHER MOMENTS#/)
      IF(FND3.LE..05) GO TO 316
      T1=S1    $    S12=S1*S1    $    T2=S2-S12/FND
      G1=T1/FND    $    G2=T2/FND1
      SG1=G2/FND    $    SG1=SQRT(SG1)    $    G1SG1=G1/SG1
      T2=S2    $    G2=T2/FND1
      SG2=2.*(G2**2)/(FND-1.)
      SG2=SQRT(SG2)    $    G2SG2=(G2-1.)/SG2
      T3=S3    $    T4=S4
      G3=FND*T3/(FND1*FND2)
      G4=(FND+1.)*T4-3.*(FND-1.)*S2*S2/FND
      G4=FND*G4/(FND1*FND2*FND3)
      FND1=FND-1.    $    FND2=FND-2.    $    FND3=FND-3.

      SG3=6.*FND*FND1/(FND2*(FND+1.)*(FND+3.))
      SG4=24.*FND*FND1*FND1/(FND3*FND2*(FND+3.)*(FND+5.))

      SG3=SQRT(SG3)    $    SG4=SQRT(SG4)
      G3SG3=G3/SG3    $    G4SG4=G4/SG4
      WRITE(3,313) NDATA    $    WRITE(3,314)
  313 FORMAT(/1H #ANALYSIS OF T=V/S.E. (EXPECTATIONS ARE USED IN HIGHER
     *MOMENTS)     NDATA=#I4/1H 76(#-#)3X#CHARACTERISTICS#)
  314 FORMAT(1H 22X#EXPECTATION# 6X#ESTIMATE#5X#STAND.ERROR#5X#(ES-EX)/S
     *E#)
      MOMENT1=10HZERO        $MOMENT2=10HUNITY       $MOMENT3=10HNORMAL
      MOMENT4=10HNORMAL
      IF(G1SG1.GE.2.) MOMENT1=10HPOSITIVE
      IF(G1SG1.LE.-2.)MOMENT1=10HNEGATIVE
      IF(G3SG3.GE.2.) MOMENT3=10HRIGHT SKEW
      IF(G3SG3.LE.-2.)MOMENT3=10HLEFT SKEW
      IF(G4SG4.GE.2.) MOMENT4=10HTOO STEEP
      IF(G4SG4.LE.-2.)MOMENT4=10HTOO FLAT
      WRITE(3,315) G1,SG1,G1SG1,MOMENT1,G2,SG2,G2SG2,MOMENT2,G3,SG3,G3SG
     *3,MOMENT3,G4,SG4,G4SG4,MOMENT4
  315 FORMAT(1H #MEAN#23X#0#4X,2E14.5,F14.2,8X,A10/1H #VARIANCE#19X#1#4X
     *2E14.5,F14.2,8X,A10/1H #SKEWNESS,GAMMA(1)#10X#0# 4X 2E14.5,F14.2,
     *8X,A10/1H #KURTOSIS,GAMMA(2)#10X#0#4X2E14.5,F14.2,8XA10/1H 76(#-#)
     */)
  316 CONTINUE


      FN(1)=FN(14)=.1$FN(2)=FN(13)=.5$FN(3)=FN(12)=1.7$FN(4)=FN(11)=4.4
      FN(5)=FN(10)=9.2$FN(6)=FN(9)=15.0$FN(7)=FN(8)=19.1
      DO 145 I=1,14
  145 CHI1(I)=FN(I)*FLOAT(NDATA)/100.
      DO 84 ICLSS=1,14
   84 FCLSS(ICLSS)=FLOAT(KLAS(ICLSS))*100./FLOAT(NDATA)
      WRITE(3,170) (JDOT,MI=1,136)
      WRITE(3,86) (KLAS(I),I=1,14),(FCLSS(I),I=1,14)
   86 FORMAT(1H+14X15(#        (#)/1H #HISTOGRAM OF T    #1X,14I8/1H+
     *14X15(#          )#)/1H #RELATIVE TO 100 0/0 #14F8.1/1H+14X15(#
     * (#)/1H 14X15(#          )#))
      WRITE(3,87) NDATA,(CHI1(I),I=1,14)
   87 FORMAT(1H #CLASS LIMITS#15X#-3.0#4X#-2.5#4X#-2.0#4X#-1.5#4X#-1.0#/
     * 4X#-0.5#4X# -0#4X#+0.5#4X#+1.0#4X#+1.5#4X#+2.0#4X#+2.5#4X#+3.0#/
     *1H+14X#        .#)04X#        ,#/1H 14X15(#          )#)/
     *1H #NORMAL DISTRIBUTION#6X#0.1#5X#0.5#5X#1.7#5X#4.4#5X#9.2#4X#15.0
     *#4X#19.1#4X#19.1#4X#15.0#5X#9.2#5X#4.4#5X#1.7#5X#0.5#5X#0.1#/1H+
     *14X15(#          (#)/1H #FOR NDATA=#I4,5X,14F8.0/1H+14X15(#       )#
     */1H 14X15(#          (#))
      WRITE(3,170) (JDOT,MI=1,136)

      SK(1) =.975 $ SK(2) =.842 $ SK(3) =.708 $ SK(4) =.624 $ SK(5)=.565
      SK(6) =.521 $ SK(7) =.486 $ SK(8) =.457 $ SK(9) =.432 $SK(10)=.410
      SK(11)=.391 $ SK(12)=.375 $ SK(13)=.361 $ SK(14)=.349 $SK(15)=.338
      SK(16)=.328 $ SK(17)=.318 $ SK(18)=.309 $ SK(19)=.301 $SK(20)=.294
      SK(21)=.287 $ SK(22)=.281 $ SK(23)=.275 $ SK(24)=.269 $SK(25)=.264
      SK(26)=.259 $ SK(27)=.254 $ SK(28)=.250 $ SK(29)=.246 $SK(30)=.242
      SK(31)=.238 $ SK(32)=.234 $ SK(33)=.231 $ SK(34)=.227 $SK(35)=.224
       SKA=136./SQRT(FLOAT(NDATA)) $  IF(NDATA.LE.35) SKA=100.*SK(NDATA)

      SN(1) =.000001$ SN(2) =00.135 $ SN(3) =00.621 $ SN(4) =02.275
      SN(5) =06.682 $ SN(6) =15.866 $ SN(7) =30.854 $ SN(8) =50.000
      SN(9) =69.146 $ SN(10)=84.135 $ SN(11)=93.319 $ SN(12)=97.725
      SN(13)=99.379 $ SN(14)=99.865 $ SN(15)=100.000


C     ANALYSIS OF T-VALUES
C
C     KOLMOGOROV-SMIRNOV TEST
C
C
      WRITE(3,300) NDATA,SKA    $   WRITE(3,307)
  300 FORMAT(/1H #KOLMOGOROV-SMIRNOV ONE-SAMPLE 95 0/0 TWO-TAILED TEST
     *FOR CUMULATIVE DISTRIBUTION OF T=V/S.E.,  NDATA=#I4,#,   CRITICAL
     * VALUE=#F7.1/)
      SKU(1)=SKA $    FCLSS0=.0    $    SKL(1)=.0
      DO 301 I=2,15    $ IF(I.GT.2) FCLSS(I-1)=FCLSS(I-1)+FCLSS(I-2)
      SKU(I)=FCLSS(I-1)+SKA    $    IF(SKU(I).GT.100.)SKU(I)=100.000001
      SKL(I)=FCLSS(I-1)-SKA    $    IF(SKL(I).LT..0  )SKL(I)=    .0
  301 CONTINUE
      WRITE(3,302)(SKU(I),I=1,15),FCLSS0,(FCLSS(I),I=1,14)
  302 FORMAT(1H #UPPER BOUND#5X15F8.2 /1H #DISTR. OF T    #2X15F8.2)
      WRITE(3,303)(SKL(I),I=1,15),(SN(I),I=1,15)
  303 FORMAT(1H #LOWER BOUND#5X15F8.2 /1H0#NORMAL DISTR(0,1)#1X15F8.2/)
                          ISN=1H#    $    IS0=1H     $    ISK=0

      DO 305 I=1,15    $    ISK=ISK+8
      IF(SN(I).LE.SKL(I).OR.SKU(I).LE.SN(I)) WRITE(3,306)(IS0,J=1,ISK),
     *ISN
  305 CONTINUE    $    WRITE(3,307)
  306 FORMAT(1H+13X123A1)
  307 FORMAT( 1H 16(#-#),15(#-------)#),#--#)
       WRITE(3,304)
  304 FORMAT(1H #OUTSIDE INTERVAL = *#)
      FCPL=100.-FCLSS(7)    $    WRITE(3,308) FCLSS(7),FCPL
  308 FORMAT(/1H #NEGATIVE T VALUES  =#F7.1# 0/0#/1H #POSITIVE T VALUES
     * =#F7.1# 0/0#)

   88 CONTINUE

      WRITE(3,108) NOCLC,IREP,END,TA(1)                  $   WRITE(3,109)
  108 FORMAT(1H1 #*****COLLECTION OF RESULTS AFTER#I4#   CYCLES *** SUMR
     *Y NO#I3#  *** TOTAL NO OF SUMRIES TO BE PRODUCED#I3#  *** DATE #A1
     *0# ******#)
  109 FORMAT(1H 32X,7(#=#),19X,5(#=#),38X,5(#=#),8X,12(#=#))
C
      CALL LISTING(1)
C
  110 IF(LSUMRY.EQ.1) GO TO 119
      IF(NYS2.EQ.4H  **) MAXCYCL=1000
      IF(NOCLC.EQ.NCYCLS.AND.IREP.EQ.END) GO TO 119
      IF(MAXCYCL.NE.1000) GO TO 118
      WRITE(3,207)    $    WRITE(3,42)
  119    NCRIT=ALOG10(COSCRIT) + 0.5
      WRITE(3,116) NCRIT,NDATA,NPART,NYS1,NYS2,NYS3,NOYES3
  116 FORMAT(/1H 13X#COS CRITERIUM FULFILLED WITH#I3#  ZEROS FOR /#12X
     *#NO    YES# T101#NUMBER OF DATA# I7/1H 42X#****#I3)#NPART=#I3,2X,A4,
     *2X,A4/1H 69X#TOTAL COSINE #3X,2X,A4,2X,A4)
      WRITE(3,117)NPAR, NOYES3,NOYES2,NC,NCYCLES
  117 FORMAT(1H 54X#NPAR =#I3,2(2X,A4)#  /  IN CYCLE NO#I4#  OUT OF#I4#
     * CYCLES#)
      IF(LSUMRY.EQ.1) GO TO 120
      IF(MAXCYCL.NE.1000) GO TO 118
      CALL PUNCH(0)    $    RETURN 440
  120 CONTINUE
C
      DO 120 I=1,NPAR    $    DO 120 J=1,NCYCLES
  120 IPA(J,I)=1H,
      WHEN=DATE(TA(1))   $  CLOCK=TIME(TA(4))   $  CPTIME=SECOND(TA(5))
      TASTA3=TA(5)-TA(3)
      WRITE(3,124)
  124 FORMAT(///1H0#STATISTICS#61(#=#)/)
      WRITE(3,112)   $   WRITE(3,113) TA(1),TA(2),TA(3)
```

```
  112 FORMAT(1H #FROM#)
  113 FORMAT(1H+11X# DATE  #A10#    TIME  #A10#    CP-SEC.USED#F8.3)
      WRITE(3,114) $ WRITE(3,113) TA(1),TA(4),TA(5) $ WRITE(3,115)TASTA3
  114 FORMAT(1H #TILL#)
  115 FORMAT(/1H 33X#USED FOR THIS JOB  CP-SEC.#F12.3)
      INK3=I3   $   INK7=I7
      NCYC=NCYCLES*(IREP-1)+NOCLC
      WRITE(3,121)NRINSUB(3),NRINSUB(4),NRINSUB(5),NRINSUB(7),NRINSUB(9)
  121 FORMAT(//1H #TOTAL NO OF ENTRIES IN  FNCTN  DFDA  D2FDA  SOLVE  MI
     *N#/1H 23X,2I6,2I7,I5)
      WRITE(3,122) END,IREP,NCYCLES,NOCLC,NCYC
  122 FORMAT(/1H #DEFINED NO OF SUMMARIES,  PRODUCED,  CYCLES/SUMRY,  PR
     *ODUCED IN LAST ONE,  TOTAL NO OF CYCLES#/1H 12X,2I11,I15,I23,I21/1
     *H 93(#=#)/)
      IF(MAXCYCL.NE.1000) GO TO 118
      WRITE(3,123)
  123 FORMAT(1H+ 105X#REQUIRED CONDITION FULFILLED#/1H 105X  #PROCESS TE
     *RMINATES AT-STOP 55-#/1H 105X,#IN NLV FROM MOWA ENTRY SUMRY#
     */1H 105X,30(#=#))
      WRITE(3,125)
  125 FORMAT(//1H 53X18(#-#)/1H 53X#=====END OF JOB=====#/1H 53X18(#-#))
      RETURN A40
  118 CPTIME=SECOND(FS)   $   EXTIME(12)=EXTIME(12)+FS-ST
      RETURN
C
      ENTRY COMBIN
C
CALCULATION FOR (NPAR COMBINATORIAL 1) TO (NPAR COMB. NPART) PARAMETERS
C
      IF(NC.GE.3) RETURN
      NOCLC=2        $        NC=1        $       NCYCLS=4
      CALL MIN(DELTA,0,1.,0.,0,30,0,0)
      DO 501 K=1,NPAR
  501 B(K)=A(K)
      IF(NRINSUB(20).GT.0) GO TO 514
      SQRESM=SQRES=SQA
      NCOMB=NPART
      NP=NPAR    $    IF(NPAR.GT.14) NP=14
      DO 500 K=1,NP
  500 ISTAR(K)=IP(K)
      NRINSUB(20)=NRINSUB(20)+1
      WRITE(3,520) (K,K=1,NPAR)          $    NEQUAL=4H====
      WRITE(3,521) (IP(K),K=1,NPAR)      $    WRITE(3,522) (NEQUAL,I=1,NP)
      WRITE(3,524) SQA,NP,NP,NPART       $    WRITE(3,525)
      WRITE(3,42)
  520 FORMAT(1H1#     K =#30I4)
  521 FORMAT(1H # IP(K)=#30I4)
  522 FORMAT(1H 8X30A4)
  524 FORMAT(1H018X#SQA=#E12.5#  WHICH  IS  100  0/0 .  COMBINATIONS (NP
     *AR,NPART) ARE TAKEN OUT OF THE FIRST#I4#  PARAMETERS#/1H 72X#(#I3
     *# ,#I4# )#/)
  525 FORMAT(1H  #(NPAR )#9X#MIN IN#5X#SUM OF SQUARES#          /1H #(NPA
     *RT)  NPART  ... IT#5X#SQB#8X#0/0# 10X #EPS# 11X #NAME# 8X   #CURVA
     *TURE#5X#PARAMETERS USED TO OBTAIN SQB#/)

      DO 502 NCMB=1,NCOMB
      NUSED=0
      DO 503 K=1,NP
  503 KLAS(K)=0
      NPART=NCMB
      NCB=1    $    KCB=1    $    NP1=NP+1
      DO 504 I=1,NCMB
      NCB=NCB*(NP1-I)
  504 KCB=KCB*I
      NCB=NCB/KCB

      DO 506 ITH=1,NCB
  507 DO 508 K1=1,NP      $    K=NP1-K1
      IF(KLAS(K).EQ.1) GO TO 509
      KLAS(K)=1
  508 NUSED=NUSED+1
      GO TO 510
  509 KLAS(K)=0
      NUSED=NUSED-1
  510 IF(NUSED.NE.NCMB) GO TO 507
C
      INSERT=0    $    NSO=NUSED
      DO 511 K=1,NP
      IF(KLAS(K).NE.0) GO TO 512
      NSO=NSO+1
      JP(IP(K))=NSO    $    GO TO 511
  512 INSERT=INSERT+1
      JP(IP(K))=INSERT
  511 CONTINUE
      DO 513 K=1,NPAR
  513 IP(JP(K))=K
      RETURN

  514 SQMN=SQMIN(NOCLC)

      SQ00=100.*SQMN/SQRES
      WRITE(3,527) NPART,NRINSUB(25),SQMN,SQ00,EPS,(IP(J),J=1,NCMB)
  527 FORMAT(1H 4X,2I8,E14.5,F8.3,E15.4,32X14I3)
      IF(SQMN.GE.SQRESM) GO TO 530
      SQRESM=SQMN $ NPSAVE=NPART $ NCSAVE=NCB $ SQ00SV=SQ00
      DO 531 J=1,NP
  531 IPX(15,J)=IP(J)
      WRITE(3,535)
  535 FORMAT(1H+43X#*#46X#*#)
  530 CONTINUE
      IF(ITH.EQ.1) WRITE(3,536) NCB
  536 FORMAT(1H+I4)
      IF(NCMB.EQ.1) WRITE(3,528) NAME(IP(1))
  528 FORMAT(1H+62XA10)
      IF(NRINSUB(5).GT.0) WRITE(3,529) XGEO(1)
  529 FORMAT(1H+74XE12.3)
      DO 516 K=1,NP
  516 IP(K)=ISTAR(K)
  506 CONTINUE

      WRITE(3,523)
  523 FORMAT(1H )
  502 CONTINUE

      DO 532 J=1,NP
  532 IP(J)=IPX(15,J)
      DO 534 J=1,NPAR
  534 JP(IP(J))=J
      WRITE(3,42)
      WRITE(3,533) NCSAVE,NPSAVE,SQRESM,SQ00SV
  533 FORMAT(  1H #BEST SOLUTION FOUND IN#/1H0I4,I8,8XE14.5,F8.3# ***#)
      NPART=NPSAVE
      NC=2    $    NOCLC=3
      NUMBR(NOCLC)=0   $   EPMIN(NOCLC)=.0
      WRITE(3,515)
  515 FORMAT(//1H #NEW INITIAL POINT OBTAINED BY COMBINATORIAL SEARCH CO
     *ULD BE#//1H 2X#K#10X#A(K)#8X#TO BE USED#/)
      DO 517 K=1,NPAR   $   IJ=IP(K)
      WRITE(3,518) IJ,A(IJ)
      IF(K.LE.NPART) WRITE(3,519)
  517 CONTINUE
  519 FORMAT(1H+29X#**#)
  518 FORMAT(1H 13,E20.10)
      RETURN
      END
*DECK MIN
      SUBROUTINE MIN(CRCTN,LETLIST,ITAN,START,STEP,ITIMES,LPX,JUMP)

      REAL M    $    INTEGER END
*CALL DIMMIN
*CALL DVAR
*CALL OFIX
```

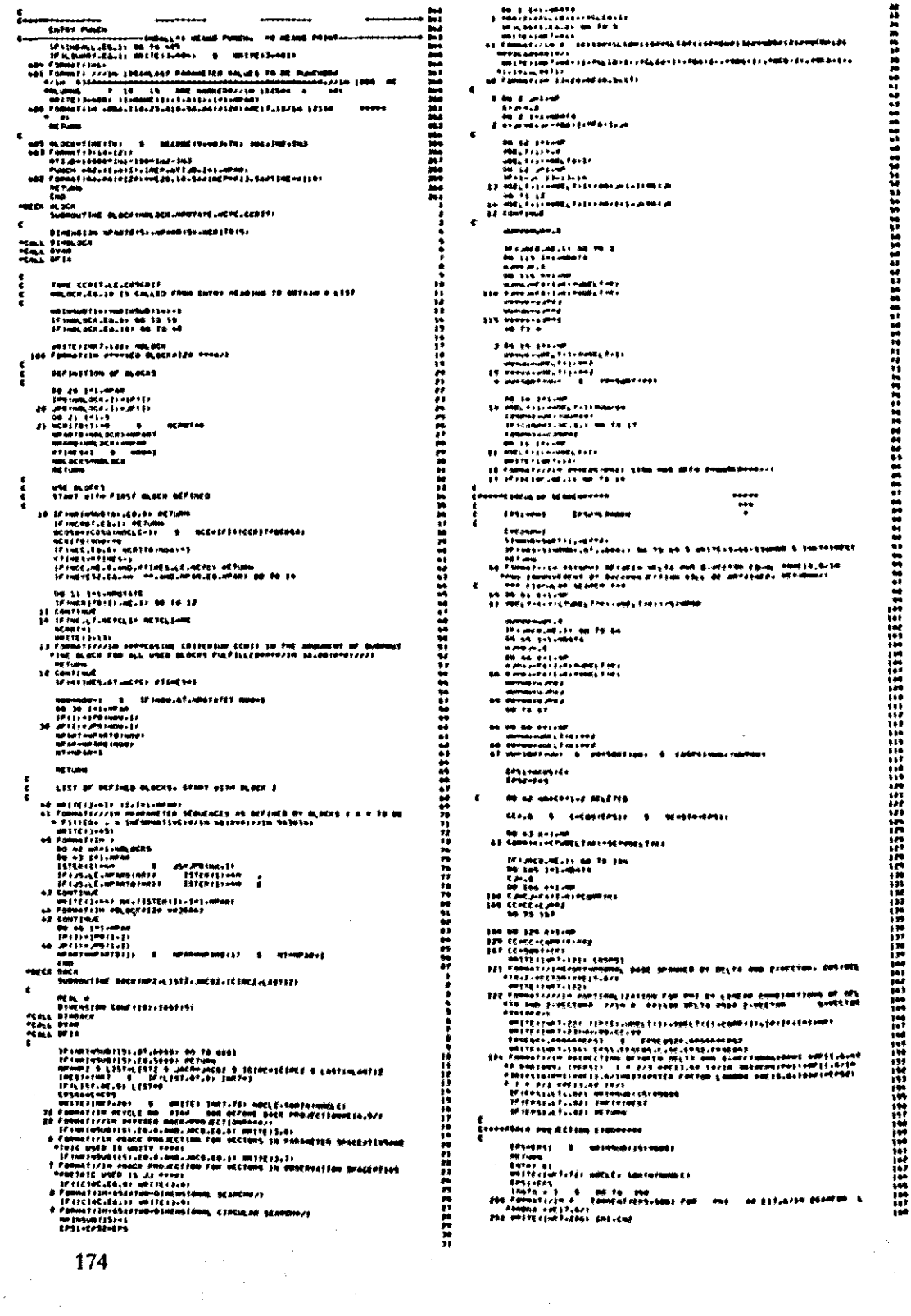
```
C      CURVATURE IN DIRECTION OF CRCTN
C
      IF(NRINSUB(5).EQ.0) GO TO 5000
      DO 130 I=1,NDATA
  130 FKS(I)=FJS(I)=.0
      DO 131 L=1,NPART
      DO 132 J=1,NPART
      IF(J-L) 133,133,134
  133 K=(J-1)*(2*NPAR-J)/2+L
      GO TO 132
  134 K=(L-1)*(2*NPAR-L)/2+J
  132 KS(J)=K

      DO 135 I=1,NDATA
      FKSSUM=.0
      DO 136 J=1,NPART
      K=KS(J)
  136 FKSSUM=FKSSUM+FAA(I,K)*CRCTN(J)
      FKSSUM=FKSSUM*CRCTN(L)
  135 FKS(I)=FKS(I)+FKSSUM
  131 CONTINUE
      DO 137 I=1,NDATA
      FJSSUM=.0
      DO 138 J=1,NPART
  138 FJSSUM=FJSSUM+FA(I,J)*CRCTN(J)
      FJS(I)=FJSSUM
  137 CONTINUE

      FJSJS=FJSKS=.0
      DO 139 I=1,NDATA
      FJSJS=FJSJS+FJS(I)**2
  139 FJSKS=FJSKS+FJS(I)*FKS(I)
      DO 140 I=1,NDATA
      FSS(I)=FKS(I)*FJSJS-FJS(I)*FJSKS
  140 FSS(I)=FSS(I)/FJSJS**2
      DO 142 K=1,NPART
      FJ=.0
      DO 141 I=1,NDATA
  141 FJ=FJ+FA(I,K)*FSS(I)
  142 TJFSS(K)=FJ

      DO 195 K=1,NPART
      DC=.0
      DO 192 J=1,NPART
      IF(K-J) 193,193,194
  194 DC=DC+M(K+1,J)*TJFSS(J)
      GO TO 192
  193 DC=DC+M(J+1,K)*TJFSS(J)
  192 CONTINUE
      DCURVE(K)=DC
  195 CONTINUE
      FSSG2=.0
      DO 184 I=1,NDATA
      TJD=.0
      DO 185 K=1,NPART
  185 TJD=TJD+FA(I,K)*DCURVE(K)
      FSSG2=FSSG2+TJD*TJD
  184 CONTINUE
      XGEO(NC)=SQRT(FSSG2)

      IF(ITIMES.NE.0) GO TO 5000
      IF(LETLIST.EQ.0) RETURN

      DO 200 K=1,NPAR
  200 B(K)=A(K)
      WRITE(3,146)
  146 FORMAT(/1H #GEODESIC CURVATURE/IN DIRECTION OF SEARCH/EMANATING FR
     *OM#4X#K#7X#NAME#/)
      DO 143 K=1,NPAR
      KP=IP(K)
      WRITE(3,144) CRCTN(K),A(KP),KP,NAME(KP)
      IF(K.EQ.1) WRITE(3,145) XGEO(NC)
  143 CONTINUE
  144 FORMAT(1H 15XE22.6,E17.6,I7,3XA10)
  145 FORMAT(1H+E15.6)
      WRITE(3,147)
      WRITE(3,199)
  199 FORMAT(1H #PARAMETER VALUES B = A#//)
  147 FORMAT(1H 74(#-#)//)
      RETURN

 5000 CONTINUE

      IF(NRINSUB(9).GT.0) GO TO 19
C
C     NEPS0 IS USED TO DETECT WHETHER SUB MIN WAS ENTERED WITH EPS=.0,
C     WHICH MEANS THAT NO PROGRESS CAN BE MADE. SEE LABEL 300
C     NBREAK SAVES THE BREAK DOWN CONDITION OF THE PARAMETER VECTOR,
C     SEE 315
C     NOWXMA CONTROLS WRITE DFDA VECTORS BY
C     LAST ARGUMENT VARIABLE OF SUBPROGRAM DFDA
C     I2000 FOR LISTING SQB-EPS RELATIONSHIP (IS AUTOMATICALLY USED)
C
      NEPS0=-1    $    NBREAK=3HNO    $    NPWAS=NPART
C
C
C     NPWAS IS USED FOR ITAN=0 TO OBTAIN OUTPUT WITH PARAMETER
C     THAT HAS BEEN DELETED IN PROCEEDING SUB MIN ROUTINE AFTER BREAK DOWN
C
   19 CONTINUE
      NOWXMA=0
      CPTIME=SECOND(ST)   $   NRINSUB( 9)=NRINSUB( 9)+1
      LIST=LETLIST    $    IF(LIST.LE.1) GO TO 400
      I33=INK3    $    I77=INK7    $    INK3=INK7=3    $    LIST=LIST-2
  400 CONTINUE
      KFIK=ITCOND=EPSLB=0.0   $   EPSBE=EPS
      NPDO=NPART    $    IF(ITAN.EQ.0) NPDO=NPWAS
C
C-----PARAMETER CONDITIONS, IF ANY, TO BE PERMUTED, ELSE GO TO 300---
C
      NBL=0    $    IF(NB.EQ.0) GO TO 300
      NBL=NB   $    IF(NPDO.LT.NPAR) GO TO 310
      DO 301 I=1,NBL
  301 IBM(I)=IB(I)     $    GO TO 300
  310 NBL=0
      DO 320 I=1,NB    $    IBI=IB(I)
      IF(JP(IBI).GT.NPDO) GO TO 320
      NBL=NBL+1    $    IBM(NBL)=IBI
  320 CONTINUE
C
C-----300 ROUTINE MINIMUM-----INITIAL VALUES-----
C
  300 CONTINUE
      IFZERO=0
      IF(EPS.GT.10.) EPSBE=EPS=10.
      MOVE=EPS    $    IF(MOVE.GT..0) GO TO 8    $    EPSBE=EPS=.05
      WRITE(3,10) MOVE,EPSBE,NOCLC
      NEPS0=NOCLC
   10 FORMAT(/1H0#EPS WAS#E13.5#   EPS IS NOW#E13.5#     IN CYCLE#I5/)
      IF(NOCLC-NEPS0.NE.1) GO TO 8    $    WRITE(3,28) NEPS0, NOCLC
   28 FORMAT(//1H #SUB MIN ENTERED WITH EPS= 0#//1H #CORRECTION VECTOR P
     *ROBABLY A ZERO VECTOR#//1H #OR#//1H #PARAMETER VALUES TOO CLOSE TO
     * BOUNDS#//1H #STOP 60 (MIN)#
     *///1H #FIRST TIME WAS IN CYCLE NO#I3#, SECOND TIME IN CYCLE NO#I3)
      STOP 60

    8 CONTINUE   $    EPSUB=EPS*EOS*EPS
      I2000=0    $    INEG=0
      SQTMN=SQTNP=SQA
      IF(ITAN.NE.0) GO TO 5
      I33=INK3    $    I77=INK7    $    INK3=INK7=3
    5 IX=3    $    IF(ITIMES.GT.IX) IX=ITIMES
```

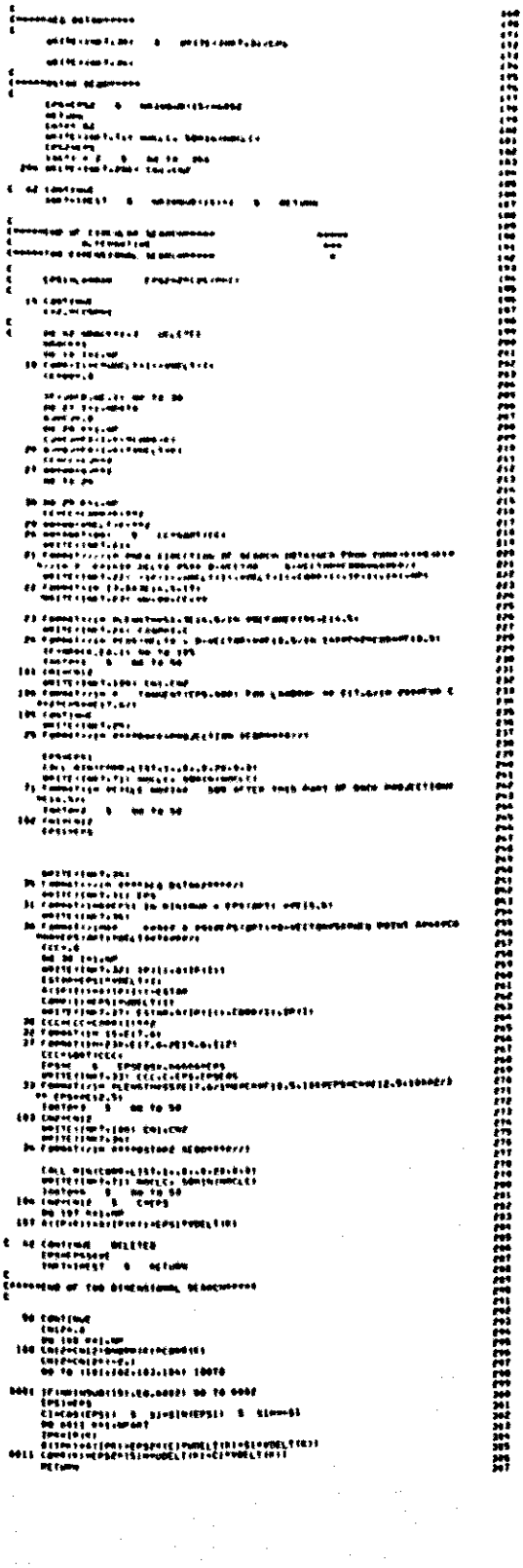
```
      WRITE(INK3,15)  $  WRITE(INK3,100) NOCLC  $  WRITE(INK3,61) EPSBE
   15 FORMAT(1H),39(#*#))
  100 FORMAT(1H0#***XEQ MIN***START OF CYCLE NO#I4#  ****#/)
   61 FORMAT(1H #EPS INITIAL=#E12.6)
      NSQ=IREQN=IREQP=EPSN=EPSP=TNSQP=.0
      IRNLOW=IRPLOW=0
C
      SQLOW=SQA   $   EPSLOW=TANLOW=.0
      SAVERN=RN   $   SAVERP=RP   $   SAVER=R
      EPST=EPS
      EPS=.0
      IF(ITAN.EQ.0) EPS=START
   10 ITO=1  $  GO TO 1000
C
 1000 IF(NSQ.LT.IX) GO TO 13  $  QEPS=EPS  $  IF(ITAN.GT.0) GO TO 3
   13 NSQ=NSQ+1
C
*
**
***
****

 2000       IF(LPX.GT.0) GO TO 3001
      IF(NRINSUB(15).GT.6000) GO TO 6001
C
      DO 12 I=1,NPDO
   12      B(IP(I))=A(IP(I))+EPS*CRCTN(I)

****
***
**
*
C
C-----IF BOUNDS DO 3000---ELSE GO TO 4000-----
C
 3000 IF((NBL.EQ.0.OR.NSQ.EQ.1).AND.ITAN.NE.0) GO TO 4000
C
C         KFIX COUNTS USE OF 0.5*(EPSLB+EPSUB), TO BE .LE. 20
C         KFIXC(K) IN COMMON COUNTS THIS FOR EACH PARAMETER
C         EPSDIF=EPSUB-EPSLB WHICH SHOULD BE .LE. .05*EPSUB
C         ITCOND=KFIX IF .GT. 0 THAN PARAMETER VECTOR FOR NPART WILL BE
C                               BROKEN DOWN UNLESS ON RISING BRANCH WITH
C                               POSITIVE SLOPE
C      NBREAK IS USED TO SAVE THE USE THAT IS MADE OF BREAK DOWN OPTION
C                THEN NO ITAN=0 OPTION IS HONOURED
C         KDELETE IS THE INDEX OF THE PARAMETER TO BE DELETED
C
      IF(ITAN.EQ.0.AND.NBL.EQ.0) GO TO 4000
C
C   --IF OUTSIDE RANGE, GO INSIDE-----
C
      INSIDE = 1
      DO 325 I=1,NBL   $   K=IBH(I)
      IF(B(K).LT.ALB(K).OR.AUB(K).LT.B(K)) GO TO 326
  325 CONTINUE
      IF(KFIX.GT.0) K=KDELETE
C
      IF(KFIX.EQ.0) GO TO 4000   $   GO TO 327
  326 IF(ITAN.EQ.0) GO TO 3999
      IF(KFIX.EQ.0.AND.LIST.NE.0) WRITE(INK3,306)
      IF(KFIX.EQ.0.AND.LIST.EQ.1) WRITE(3,4)
  306 FORMAT(//1H 4X#PAR NO#4X#EPS INSIDE#5X#+-LOWER BOUND----PAR.VALUE-
     *---UPPER BOUND-+#4X#EPS OUTSIDE#2X#INSIDE  ITER/PAR#5X#NEXT EPS#5X
     *#NO-OF-ENTRIES#)
      INSIDE=0
      KDELETE =K
      IF(KFIX.GT.0) GO TO 327
        EPSLB=EPSN   $   EPSUB=EPS   $   GO TO 328
  327 IF(INSIDE.EQ.1) EPSLB=EPS
      IF(INSIDE.EQ.0) EPSUB=EPS
  328 EPS=0.5*(EPSLB+EPSUB)  $  KFIX=KFIX+1   $   KFIXC(K)=KFIXC(K)+1
      EPSDIF=EPSUB-EPSLB
      IF(LIST.NE.0) WRITE(3,324) K,EPSLB,ALB(K),B(K),AUB(K),EPSUB,INSIDE
     *,KFIXC(K),EPS,KFIX
  324 FORMAT(1H I8,E17.6,E17.6,2E14.6,E17.6,I6,I9,E18.6,I10)
      IF(KFIX.LT.20.AND.EPSDIF.GT..05*EPSUB) GO TO 2000
      EPS=EPSLB  $  ITCOND=KFIX  $  KFIX=0  $  IF(LIST.EQ.1) WRITE(3,4)
    4 FORMAT(/)
      GO TO 2000
 3999 IF(EPS.GE..0) GO TO 3998
      EPS=EPS+STEP   $   GO TO 1000
 3998 IX=NSQ-1   $   ITCOND=1   $   GO TO 105
C
C-----4000 ROUTINE MINIMUM-----CALCUATION OF FUNCTION VALUES-----
C
C
 4000 CONTINUE
C
      NRINSUB(25)=NSQ

      CALL FNCTN(B,0,SQB,0,1,YCLCB,YOCB,0,0)
      CALL OFDA(B,HB,YCLCB,YOCB,0,0,NOWXHA)
      CALL NRMEQ(YOCB,BNORM,0,1,HB,0,0,N02)

           IF(I2000.GE.1) GO TO 2001
      IF(SQB.LT.SQLOW) EPSLOW=EPS
      IF(SQB.LT.SQLOW) SQLOW=SQB

C
C--------CALCULATION OF TANGENT TO (EPS,SQB)-RELATIONSHIP--------
C
      HNORM=HCRCT=TNSQB=.0
C
      S0=BNORM(NT)
      DO 205 I=1,NPART
      BN=(-2.)*BNORM(I)
      TNSQB=TNSQB+BN*CRCTN(I)   $   HNORM=HNORM+BN**2
  205 HCRCT=HCRCT+CRCTN(I)**2
      HNORM=SQRT(HNORM)
      HCRCT=SQRT(HCRCT)
C
      USETNB=TNSQB
      TN2=TNSQB/HCRCT
      TN3=TN2/HNORM
      IF(ITAN.EQ.2) USETNB=TN2  $  IF(ITAN.EQ.3) USETNB=TN3
      IF(SQB.LE.SQLOW) TANLOW=USETNB
C
      IF(ITCOND.GT.0.AND.USETNB.GT..0) ITCOND=0
C
      IF(NSQ.GT.1) GO TO 206
      IF(ABS(TNSQB).LT..1E-10) WRITE(3,203) TNSQB,NOCLC
      IF(ABS(TNSQB).LT..1E-10) GO TO 80
  203 FORMAT(//1H #TNSQB=#E14.5# .LT. .1E-10, GO TO 80 IN MIN. IN CYCLE#
     *I5/)
      RNRES=RN=+RN*USETNB   $   RPRES=RP=(-RP)*USETNB
      IF(RNRES.GT.(-.1E-08))  RNRES=RN=-.1E-08
      IF(RPRES.LT..1E-08) RPRES=RP=.1E-08
C
  206 GOTO (1,2,3,2,3),ITO
C
C
C
    1 CONTINUE
      IF(NSQ-1)105,71,105
C
   71 IF(INK3.EQ.7.AND.LIST.EQ.1) WRITE(3,14)
      IF(INK3.EQ.7.AND.LIST.EQ.1) WRITE(3,100) NOCLC
      IF(USETNB.LE.0.0.OR.ITAN.EQ.0) GO TO 120
      INEG=INEG+1
      DO 121 I=1,NPART
  121 CRCTN(I)=-CRCTN(I)
      WRITE(3,122) NOCLC,USETNB,EPS,SQB
      WRITE(3,129)
  129 FORMAT(/1H #IF THIS OCCURS TWO TIMES IN THE SAME CYCLE, EXECUTION
     *TERMINATES AT STOP 61 (MIN)#//)
  122 FORMAT(1H0#CORRECTION SIGN HAS BEEN CHANGED#19X#IN CYCLE#I5/1H #BE
     *CAUSE TAN=#E14.5#  AT EPS=#E14.5/1H 8X#SQB=#E14.5/)
      NSQ=0 $ IF(INEG.EQ.2)STOP 61 $ RN=SAVERN $ RP=SAVERP $ GO TO 1000
  120 IF(INK3.EQ.7) GO TO 102  $  VERSCHL=SQB-S0
      PQ=EPSBE*EOS
      WRITE(INK3,101) EOS,SQB,S0,VERSCHL,PQ  $  WRITE(INK3,103)
  101 FORMAT(1H #REDUCTION  =#E12.6#  X#9X#(SQB IN FNCTN=#E12.5,3X
     *#BNORM(NT) IN NRMEQ=#E12.5,3X#DIFFERENCE=#E12.5# )#/1H #EPS#8X#=#
     *E12.6)
  103 FORMAT(1H0#  I  #7X#A(I)#11X#CRCTN(I)#7X#EPS*CRCTN(I)#8X#TOTAAL#11
     *X#NORMAAL#9X#DELTA#/)
      DO 106 I=1,NPAR   $   Q1=PQ*CRCTN(I)   $   Q2=A(IP(I))+Q1
  106 WRITE(INK3,104) IP(I),A(IP(I)),CRCTN(I),Q1,Q2,ANORM(I),DELTA(I)
  104 FORMAT(1H I3,6E17.6)
      IF(ITAN.EQ.0) WRITE(3,123) START,STEP,ITIMES
  123 FORMAT(//1H #START AT EPS=#E14.6#, STEP LENGTH=#E14.6#, NO OF STEP
     *S=#I3)
      WRITE(INK3,14)  $  WRITE(INK3,108)  $  WRITE(INK3,11)
   14 FORMAT(/)
  108 FORMAT(1H 55(#==#))
   11 FORMAT(1H 29X#(#9X#T  A  N  G  E  N  T#16X#)#9X#L  E  N  G  T  H#)
      WRITE(INK3,111) $  WRITE(INK3,112)
  111 FORMAT(1H 6X#EPS#10X#SQA#7X#I#5X#ABS#10X#DIRECT.#8X#COS#8X#(#4X
     *# NORM(PHI)#5X#CORR.VECT#)
  112 FORMAT(1H 29X#I#44X#)#)
      WRITE(INK3,113) EPS,SQ,TNSQB,TN2,TN3,HNORM,HCRCT
  113 FORMAT(1H 2E14.6# )#3E14.6#  (#2E14.6)
      WRITE(INK3,112) $  WRITE(INK3,109)
  109 FORMAT(1H 55(#- #))
  102 IF(INK7.EQ.7.OR.LIST.EQ.0.OR.ITAN.EQ.0) GO TO 105
      WRITE(INK7,14)    $   WRITE(INK7,6)
    6 FORMAT(1H 18X#NEGATIVE SIDE#43X#POSITIVE SIDE#)
      WRITE(INK7,16)  $  WRITE(INK7,7)    $  WRITE(INK7,70)
   16 FORMAT(1H+52X#(#)
    7 FORMAT(1H 6X#EPS#25X#SQ#8X#REQ#15X#EPS#25X#SQ#8X#REQ#)
   70 FORMAT(1H+21X#TAN#53X#TAN#)
      WRITE(INK7,17)
   17 FORMAT(1H+52X#)#/)
C
  105 IF(ITAN.NE.0) GO TO 107
C
C---------PRINT (EPS,SQB)-RELATIONSHIP
C
      IF(ITCOND.GT.0) GO TO 116
      IF(NSQ.EQ.2) WRITE(INK3,125)
  125 FORMAT(1H+70X#SLOPE FROM 1ST TO 3RD POINT#12X   #SQA AT 2ND POINT#
     *)
      IF(NSQ.EQ.1) SQ1=SQ   $   IF(NSQ.EQ.2) SQ2=SQ
      IF(NSQ.LT.3) GO TO 126
      SQ3=SQ   $   CHORD=(SQ3-SQ1)/(2.*STEP)
      WRITE(INK3,127) CHORD
  127 FORMAT(1H+82XE14.5)
      SQ1=SQ2   $   SQ2=SQ3
  126 CONTINUE
      WRITE(INK3,124) NSQ,EPS,SQ,TNSQB,TN2,TN3   $   EPS=EPS+STEP
  124 FORMAT(1H I3,E12.5,E13.5# )#3E14.5#  (#)
      IF(NSQ.GT.1) WRITE(INK3,128) NSQ,SQ
  128 FORMAT(1H+100X,I10,E25.15)
      IF(NSQ.LE.IX) GO TO 1000
  116 WRITE(INK3,108)
      IF(ITCOND.GT.0) WRITE(3,114) K,B(K),NSQ
  114 FORMAT(/1H #PARAMETERS HAVE BEEN CONTROLLED#5X#B(#I3# )=#E14.6#,
     *# OUT OF RANGE AT NSQ=#I4//)
      NPWAS=NPART
      R=SAVER  $  RN=SAVERN  $  RP=SAVERP  $  INK3=I33  $  INK7=I77
      EPS=EPMIN(NOCLC) $  I2000=2 $  NOWXHA=-1 $  NBL=0 $  GO TO 2000
C
C
C---------UNTIL THE FIRST TIME A POSITIVE TANGENT IS ENCOUNTERED--------
C
  107 IF(USETNB) 20,21,21
   20 EPSN=EPS
      TNSQN=USETNB
      SQTNN=SQ
      IF(TNSQN.GT.RN) IREQN=1
      IF(SQB.LE.SQLOW) IRNLOW=IREQN
      IF(LIST) 22,22,23
   23 WRITE(3,220) EPSN,TNSQN,SQTNN,IREQN,NSQ
  220 FORMAT(1H 3E14.6,I5,I6)
   22 IF(ITCOND.GE.1) GO TO 67
      IF(EPS)25,25,26
   25 EPS=EPST
      GOTO 10

   26 EPS=(2.)*EPS
      IF(EPS-10.0) 10,27,27
   27 EPSP= 10.
      EPSN= 10.
      NSQ=NSQ-2
      QEPS=EPS
      GO TO 3
C
C---------TANGENT FIRST TIME POSITIVE--------
C
   21 EPSP=EPS
      TNSQP=USETNB
      SQTNP=SQ
      IF(TNSQP.LE.RP) IREQP=1
      IF(SQB.LE.SOLOW) IRPLOW=IREQP
      IF(LIST) 51,51,24
   24 WRITE(3,221) NSQ,EPSP,TNSQP,SQTNP,IREQP
  221 FORMAT(1H 47X,I6,3X,3E14.6,I5)
   51 EPS=EPSP-R*TNSOP*(EPSP-EPSN)/(TNSQP-TNSQN)
C
C--------CHECK BOTH CONDITIONS FOR POSITIVE AND NEGATIVE TANGENT--------
C
   42 IF(IREQN.EQ.1.AND.IREQP.EQ.1) GO TO 62
      IF((IREQN.EQ.1.OR.IREQP.EQ.1).AND.R.EQ.1.) GO TO 65
C
      ITO=2   $   GO TO 1000


C
    2 CONTINUE
      IF(USETNB) 40,40,41
C
C--------TANGENT NEGATIVE--------
C
   40 EPSN=EPS
      TNSQN=USETNB
      SQTNN=SQ
      IF(TNSQN-RN)33,33,50
   50 IREQN=1
      IF(SQB.LE.SQLOW) IRNLOW=IREQN
   33 IF(LIST) 34,34,32
   32 WRITE(3,220) EPSN,TNSQN,SQTNN,IREQN,NSQ
   34 IF(ITO.EQ.4) GO TO 30
      GO TO 51
C
C---------TANGENT POSITIVE--------
C
   41 EPSP=EPS
      TNSQP=USETNB
      SQTNP=SQ
      IF(TNSQP-RP) 43,43,53
   43 IREQP=1
      IF(SQB.LE.SQLOW) IRPLOW=IREQP
   53 IF (LIST) 30,30,52
   52 WRITE(3,221) NSQ,EPSP,TNSQP,SQTNP,IREQP
```

```
   30 EPS=EPSN-R*TNSON*(EPSP-EPSN)/(TNSOP-TNSON)
      IF(ITO.EQ.4) GO TO 66
      GOTO 42
C
C---------TERMINATION OF PROCEDURE OF FINDING THE MINIMUM---------
C
   62 OEPS=EPS=0.5*(EPSP+EPSN)
      ITO=3   $   GO TO 1000
   65 ITO=4   $   GO TO 1000
   66 ITO=5   $   GO TO 1000


    3 CONTINUE
C
C CHOOSE SECOND BRANCHE OR THE FIRST ONE IF THE SECOND HAS NO MINIMUM
C FINAL CHOICE IS NOW FOR LOWEST VALUE OF TAN IN STEAD OF THAT FOR SQB
C THIS WILL BE ACHIEVED BY JUMP PARAMETER IN ARGUMENT OF MIN GT 0

C
      IF(JUMP.EQ.0) GO TO 87   $   IF(EPSLOW.GT.EPS) GO TO 80
      IF(IANLOW.EQ.1) GO TO 67
      IF(NSQ.GE.ITIMES) GO TO 67   $   GO TO 88
   80 IF(IRPLOW.EQ.1) GO TO 67
      IF(NSQ.GE.ITIMES) GO TO 67   $   GO TO 88
C
C---------PROCEED WITH ROUTINE---------
C
   87 CONTINUE
      IF(OEPS.EQ.EPSLOW) GO TO 88
   67 EPS=EPSLOW   $   SQ=SQLOW   $   USETNB=TANLOW
      IF(EPS.NE..0) GO TO 88
      IF(ITCOND.GT.0) GO TO 88
C
C         EPS(OPT) CAN STILL BE ZERO IN CASE OF TWO BRANCHES
C         REPEAT PROCEDURE WITH SMALLER STEPS
C
      IFZERO=IFZERO+1
      EPSBE=0.1*HOWE
      WRITE(3,86) HOWE,NOCLC,EPS,IFZERO,SQ,NSQ,EPSBE
   86 FORMAT(////1H055X*  I*)1X*I*/1H 55X*  I*/1H 49(***)6X*  I*13X***9X
     **2*/1H ***5X*EPS=*E15.8,5X*NOCLC =*I4, 7X***6X*  I*/1H **  EPSMIN*
     **E15.8,5X*IFZERO=*I4,7X***6X*  I*15X**      **/1H ***5X*SQB=*E15.8
     *,5X*NSQ    =*I4,7X***6X*  I*TX***9X*3*/1H **  FUNCTION PROBABLY NOT
     * CONTINUOUS NEAR EPS=0  ***6X*  0*/1H ***3X*REPEAT SUB MIN WITH 0.1
     **EPS IF IFZERO S 5   ***6X*  I *   **/1H **  PROCEED WITH EPS =*
     *E12.5,15X***6X*  I    **/1H 49(***)6X*  I - - - - - - - - - - - - - - -
     **/1H 55X*  **17X***/1H 55X*LOWEST*5X*NOT ACCEPTED AS MINIMUM*/1H
     *65X*NO JUMP HAS BEEN PERFORMED*/)
      IF(IFZERO.GT.3) NCYCLS=NOCLC
      IF(IFZERO.GT.3) END=IREP
      IF(IFZERO.GT.3) GO TO 88
      HOWE=EPS=EPSBE   $   RN=SAVERN   $   RP=SAVERP   $   R=SAVER
      GO TO 8
C
C---------COLLECTION OF RESULTS---------
C
   88 SQMIN(NOCLC)=SQ
      EPMIN(NOCLC)=EPS
      NUMBR(NOCLC)=NSQ
      I2000=1   $   NOWAMA=-1   $   NOL=0   $   GO TO 2000
 2001 NOWAMA=0   $   IF(I2000.EQ.2) RETURN

      IF(INK7.EQ.7) GO TO 89
      WRITE(INK7,14)   $   WRITE(INK7,109)
   89 JNK3=INK3   $   IF(LIST.EQ.1.OR.LPX.GT.0) JNK3=3
      WRITE(JNK3,90) NSQ   $   WRITE(JNK3,91)
   90 FORMAT(1H029X****MINIMUM FOUND IN*I5*   ITERATIONS****/)
   91 FORMAT(1H 6X*EPS*11X*TAN*11X*SQ*12X*R*12X*RP*14X*USED TANGENT*
     *11X*SQB*)
      WRITE(JNK3,78) EPS,USETNB,SQ,R,RP,ITAN,SQ   $   WRITE(JNK3,100)
   78 FORMAT(1H 5E14.6,8X*NO*I6,E24.12)
C
      IF(((SQ.LE.SQA). AND.(SQ.LE.SQTNN).AND.(SQ.LE.SQTNP))
     *.OR.(RN.EQ.0.1*ANRES).OR.EPSP.GE.10.) GO TO 63
      IANLOW=IRPLOW=0
      IREON=IREOP=0   $   RN=0.1*ANRES   $   RP=0.1*APRES   $   GO TO 2
C
   63 RN=RNRES   $   RP=RPRES
C
      WRITE(INK3,82)
   82 FORMAT(/1H04X*K*9X*A(K)*11X*CORR=CRCTN*5X*EPS(OPT)*CORR*6X*B(K)*/)
        DO 83 I=1,NPAR
      CORR(I)=CRCTN(I)
      Q=B(IP(I))-A(IP(I))
      WRITE(INK3,84) IP(I),A(IP(I)),CORR(I),Q,B(IP(I))
      IF(INK3.EQ.3.AND.KFIXC(IP(I)).GT.0) WRITE(INK3,85) IP(I)
   83 CONTINUE
   84 FORMAT(1H I5,4E17.6)
   85 FORMAT(1H+80X,*CONTROLLED PARAMETER #(*I3*  )*)
      IF(JUMP.GT.0.AND.SQ.GT.SQA)WRITE(3,92)JUMP,NOCLC,SQ,SQA
   92 FORMAT(//1H 10X*JUMP=*I2*,  SQMIN ON 2ND BRANCH IN CYCLE*I3*  EQUA
     *LS*E14.6*,  SQA=*E14.6* - - - * * * <---*//)
C
    9 WRITE(INK3,110)
  110 FORMAT(1H0****MIN ZERO****/)
      RN=SAVERN   $   RP=SAVERP   $   NBREAK=3HNO
      IF(LETLIST.LE.1) GO TO 115   $   INK3=133   $   INK7=177

  115 IF(ITCOND.EQ.0) GO TO 305
C
C---------BREAK DOWN NPART---------
C       POLICY FOR BREAKING DOWN NPART OR NOT
C           SEE ALSO CALCULATION OF TANGENT
C           IF ON RISING BRANCH (TAN.GT.0) NO BREAK DOWN TAKES PLACE
C
C
C
C           NPARTX TO BE DEFINED IN NEXT SOLVE
C
      NBREAK=3HYES
      IFST=JP(KDELETE)+1
      IF(IFST.GT.NPART) GO TO 303
      DO 302 I=IFST,NPART
      JP(IP(I))=I-1
  302 IP(I-1)=IP(I)
  303 NPWAS=NPART   $   NPART=NPART-1
  307 IP(NPART+1)=KDELETE
      JP(KDELETE)=NPART+1
      IPX(NOCLC+1,KDELETE)=3H.
      EMINE=EPSUB-EPSLB
      WRITE(3,304) EPSLB,KFIXC(KDELETE),KDELETE,EPSUB,NSQ,EMINE,SQ,NPART
     *,NOCLC,KDELETE,KDELETE,B(KDELETE),NAME(KDELETE),KDELETE
  304 FORMAT(//1H *EPS INSIDE    =*E16.6,5X****  (ITER =*I3* X FOR PAR*I3
     */1H *EPS OUTSIDE  =*E14.6,5X****  NSQ  =*I3
     */1H *DIFFERENCE    =*E14.6,5X*****/1H 33X****  SQB  =*E16.10
     */1H *NPART NOW* *I4* IN CYCLE NO*I4*  ****/1H *PARAMETER NO*I3* HA
     *S BEEN DELETED ***  B(*I2*)=*E16.10,5X*NEW CONSTANT PARAMETER
     *------->   *A10*  =  NO*I3/)
      IF(EPS.EQ..0) EPS=EPSLB
      IF(NPART.EQ.0) CALL INITL

  305 CPTIME=SECOND(FS)   $   EXTIME( 9)=EXTIME( 9)+FS-ST
      RETURN
C
C---------POLYNOMIAL, HYPERBOLIC OR EXPONENTIAL EXTRAPOLATION---------
C
 3001 CALL LINYPEX(LPX,10,0,0,0,0,0)
      IF(INEG.EQ.0) GO TO 3000
      DO 3002 I=1,NPART
 3002 CRCTN(I)=-CRCTN(I)   $   GO TO 3000
```

```
C
C---------CIRCULAR SEARCH FOR PHI AND LAMBDA---------
C
 6001 CALL BACK(NPART,0,0,0,0)
      GO TO 3000
      END
*DECK AISB
      SUBROUTINE AISB

      REAL M   $   INTEGER END
      DIMENSION ASAVE(4),ZM(5,5)
*CALL DVAR
*CALL DFIX

      IF(NRINSUB(10).EQ.0) INXPCT=0
      CPTIME=SECOND(ST)   $   NRINSUB(10)=NRINSUB(10)+1
      IF(INK3.EQ.7) GO TO 15
      WRITE(INK3,6)
    6 FORMAT(1H0****NEW POINT****/)
      Q=SQB/SQA*100.
      WRITE(INK3,43) Q
   43 FORMAT(1H0*SQB=*F10.3,2X*PERCENT OF SQA*/)
C
      WRITE(3,50)
   50 FORMAT(1H 23X*PARAMETER VALUES*/1H 4X*K*5X*A=OLD*9X*B-A*11X*B=NEW*
     *9X*CORR.*9X*EPS*/)
      DO 51 I=1,NPAR
      IV=IP(I)
      Q1=B(IV)-A(IV)
      IF(CORR(I))60,61,60
   61 Q2=.0
      GOTO 51
   60 Q2=Q1/CORR(I)
   51 WRITE(3,52) IV,A(IV),Q1,B(IV),CORR(I),Q2
   52 FORMAT(1H I5,5E14.5)

   15 DO 72 I=1,NPAR
   72 HAHB(I)=(HB(I)-HA(I))/HA(I)
      IF(INK3.NE.3.AND.INK8.NE.3.AND.NOCLC.NE.2) GO TO 16
      IF(LSUMRY.LE.1) GO TO 16
      WRITE(3,201)
      IF(NRINSUB( 9).NE.0)WRITE(3,202) $ IF(NRINSUB(5).NE.0)WRITE(3,203)
      WRITE(3,204)
      IF(NRINSUB( 9).NE.0)WRITE(3,205) $ IF(NRINSUB(5).NE.0)WRITE(3,206)
      WRITE(3,207)
      DO 211 I=1,NPAR   $   WRITE(3,208) IP(I),HB(I)
      IF(NRINSUB( 9).NE.0)  WRITE(3,209) HA(I),HAHB(I)
      IF(NRINSUB(5).NE.0.AND.I.LE.NPART) WRITE(3,210) FFAANH(I),IP(I)
  211 CONTINUE
  201 FORMAT(/1H0*COURSE OF SCALE FACTORS *)
  202 FORMAT(1H+25X*---ACCORDING TO DIFFERENCES---*)
  203 FORMAT(1H+60X*---ACCORDING TO DIFFERENTIALS---*)
  204 FORMAT(1H0*  K*10X*HB*)
  205 FORMAT(1H+32X*HA*8X*(HB-HA)/HA*)
  206 FORMAT(1H+67X*D(HA)DA/HA*9X*K*)
  207 FORMAT(1H )
  208 FORMAT(1H I3,E16.5)
  209 FORMAT(1H+19X,E19.5,E14.5)
  210 FORMAT(1H+62X,E15.5,I10)
C
   16 DO 30 I=1,NDATA
      YCLCA(I)=YCLCB(I)
   30 YOCA(I)=YOCB(I)
      SQA=SQB
C
      DO 20 I=1,NT
      HA(I)=HB(I)
   20 ANORM(I)=BNORM(I)
      DO 10 I=1,NPAR
      IV=IP(I)
      A(IV)=B(IV)
   10 XA(NOCLC,IV)=A(IV)
      IF(NT.GT.NPAR) GO TO 21
      DO 22 I=NT,NPAR
      IV=IP(I)
   22 XA(NOCLC,IV)=A(IV)
   21 WRITE(INK3,5)
    5 FORMAT(1H0****A IS B****/)
      CPTIME=SECOND(FS)   $   EXTIME(10)=EXTIME(10)+FS-ST
      RETURN
      END
*DECK LISTING
      SUBROUTINE LISTING(INDALL)

      REAL M   $   INTEGER END
*CALL DIMLIST
      DIMENSION ISYMBOL(4),YPLOT(4),ISCALE(6)
      DIMENSION IFRMT(6),JFRMT(2),BOUND(3)
*CALL DVAR
*CALL DFIX

      CPTIME=SECOND(ST)   $   NRINSUB(11)=NRINSUB(11)+1
      IF(INDALL.EQ.0.OR.INDALL.EQ.50) WRITE(3,1)
    1 FORMAT(1H1)
      WRITE(3,20)
   20 FORMAT(///1H 31X*INITIAL PARAMETER VALUES*6X*NAME*/)
      WRITE(3,13)
   13 FORMAT(1H 28X,42(***))
      WRITE(3,12)
   12 FORMAT(1H 28X***40X***)
      WRITE(3,11) (IP(I),XA(1,IP(I)),NAME(IP(I)),I=1,NPAR)
   11 FORMAT(1H 28X***I5,E20.10,3X,A10*  **)
      WRITE(3,12)
      IF(NT.LE.NPAR)WRITE(3,11)(IP(I),XA(1,IP(I)),NAME(IP(I)),I=NT,NPAR)
      IF(NT.LE.NPAR) WRITE(3,12)
      WRITE(3,13)
      WRITE(3,14)
   14 FORMAT(//)
      WRITE(3,4)
    4 FORMAT(1H *- - - - - - - - - - - - - - - - -I- - - - - - - - - - - - - - - - -
     * - I - - - - - - - - - - - - - - - - - - - - - - - - -I- - - - - - - - - - -
     * - - - - - - - I*)
      WRITE(3,6)
    6 FORMAT(1H 4X*EPS*6X*CYCLE   MIN*T29*I*T60*I*44X*I*30X*I*/1H * TO O
     *BTAIN*5X*NO   FOUND*T29*I*6X*SQUARES TO BE EQUAL*T60*I*6X*L E N G
     *T H S   T O   B E   Z E R O   I   T O T A L   L E N G T H S   I*/1H
     * * NEXT CYCLE*9X*IN..X*T29*I*7X*SQA*11X*SQF*T60*I*6X*NORMAL*9X*DEL
     *TA*6X*V(SQA-SQF)  I*6X*NORMAL*9X*DELTA     I*)
      WRITE(3,4)
      INXX=0
      DO 15 I=1,NOCLC
      INXX=INXX+NUMBR(I)
      Q=SQMIN(I)-XSQF(I)
      IF(Q) 18,19,19
   18 Q=.0
      GO TO 15
   19 Q=SQRT(Q)
   15 WRITE(3,7)EPMIN(I),I,NUMBR(I),SQMIN(I),XSQF(I),XNRM(I),XDELT(I),Q
     *,XNORM(I),XDELTA(I)
    7 FORMAT(1H E12.5,I5,I7*   I*2E14.5*  I*3E14.5*  I*2E14.5*  I*)
      WRITE(3,4)
      WRITE(3,8) INXX
    8 FORMAT(1H 14X*TOTAL*I5)
C
C
      WRITE(3,3)   $   IF(IREP.EQ.1) EPTOT=.0
    3 FORMAT(/1H0)
      IF(NRINSUB(5).GT.0) WRITE(3,5)   $   WRITE(3,47)   $   WRITE(3,27)
    5 FORMAT(1H+*EPS(PREDICTED)*)
   47 FORMAT(1H+23X*EPS*8X*EPS(OPT)*)
   27 FORMAT(1H++48X*RESULT OF  FITTING  PROCEDURE*5X*STANDARD ERROR OF
```

```
     *ESTIMATE# 5X#      TOTAL COSINE    #/1H 48X#CYCLE-NO  MULT.CORR  M
     *ULT.COS#5X#NO NDATA  D.F. STAND.ERROR# 7X#CYCLE-NO#4X# COSINE#T17#
     *NO#T22#(OPTIMUM)#T37#TOTAAL#/)
      DO 60 I=1,NOCLC
      EPTOT=EPTOT+EPMIN(I)
      NDF=DF=NDATA-NPARTX(I)
      SERROR=SQRT(SQMIN(I)/DF)
      QCOS=SQMIN(I)-XSOF(I)  $  IF(QCOS.LT.0.0) QCOS=.0
      QCOS=SQRT(QCOS/SQMIN(I))
      WRITE(3,28)I,EPMIN(I),EPTOT,I,XCRLMT(I),XCOSMT(I),I,NDATA,NDF,SERR
     *OR,I,QCOS
      IF(NRINSUB(5).GT.0) WRITE(3,2) EPSNEW(I)
   60 CONTINUE
   28 FORMAT(1H 12X,I5,2E13.5,I10,F12.4,F11.4,I8,I5,I6,F13.5,I12,F14.6)
    2 FORMAT(1H+E12.5)
      WRITE(3,26)
   26 FORMAT(1H0+3(#*#)5X60(#*#)5X22(#*#))
      IF(INDALL.EQ.0) GO TO 65   $   IF(INDALL.EQ.50) GO TO 66
C
      WRITE(3,21)
      WRITE(3,34)
   34 FORMAT(/1H 13X#TABLEAU OF PARTIAL SOLUTIONS OUT OF CYCLE NO I#/)
      WRITE(3,35) (I,I=1,NPAR)
   35 FORMAT(1H #CYCLE PART#I4,30I4)
      WRITE(3,29)
   29 FORMAT(1H # NO   SOLN#/)
C
      DO 39 K=1,NPAR
   39 ITALLY(K)=0
      DO 36 I=1,NOCLC
      IX=1HX
      DO 38 J=1,NPAR
      IF(IPX(I,J).EQ.IX) ITALLY(J)=ITALLY(J)+1
   38 CONTINUE
   36 WRITE(3,37) I,NPARTX(I),(IPX(I,J),J=1,NPAR)
   37 FORMAT(1H I3,I6,4X,30A4)
      WRITE(3,25)(ITALLY(I),I=1,NPAR)
   25 FORMAT(//1H #OCCURRENCE#I4,30I4)
C
C---------------------OUTPUT PARAMETER VALUES-----
C
   66 WRITE(3,21)
   21 FORMAT(///)
      WRITE(3,22)
   22 FORMAT(1H 7X#     PARAMETER VALUES     #/)
      NFST=1        $    NB=8
   23 NLST=NPAR
      IF(NPAR.GT.NB)  NLST=NB
      WRITE(3,9) (NAME(IP(I)),I=NFST,NLST)
    9 FORMAT(/1H 3X,8(4X,A10))
      IF(NB.EQ.0) GO TO 80
      JFRMT(1)=9H(1H+,000X
      JFRMT(2)=7H,E14.5)
      WRITE(3,68)
   68 FORMAT(1H #BOUNDS#/1H #UPPER#)
      DO 69 K=NFST,NLST
      IF(AUB(IP(K)).GT..90E+85) GO TO 69
      ISKIP=14*(K-1)+3
      ENCODE(9,70,JFRMT(1)) ISKIP
   70 FORMAT(5H(1H+,,I3,1HX)
      WRITE(3,JFRMT) AUB(IP(K))
   69 CONTINUE
      WRITE(3,78)
   78 FORMAT(1H #LOWER#)
      DO 79 K=NFST,NLST
      IF(-.90E+85.GT.ALB(IP(K))) GO TO 79
      ISKIP=14*(K-1)+3
      ENCODE(9,70,JFRMT(1)) ISKIP
      WRITE(3,JFRMT) ALB(IP(K))
   79 CONTINUE
   80 CONTINUE
      WRITE(3,16) (IP(I),I=NFST,NLST)
   16 FORMAT(/1H #CYCLE#I8,7I14,/)
      DO 24 I=1,NOCLC
   24 WRITE(3,17) I,(XA(I,IP(J)),J=NFST,NLST)
   17 FORMAT(1H I3,8E14.5)
      IF(NPAR.LE.NB) GO TO 61
      NFST=NFST+8  $  NB=NB+8  $  WRITE(3,14)  $  GO TO  23
   61 CONTINUE
      IF(INDALL.EQ.50) GO TO 320
C
C--------------------- OUTPUT DIFFERENCES----------
C
      IF(NOCLC.EQ.1) GO TO 62
      DO 45 I=1,NPAR
   45 TOT21(I)=.0
      WRITE(3,21)  $  WRITE(3,40)
   40 FORMAT(1H 7X#    DIFFERENCES OF PARAMETER VALUES TO OBTAIN CYCLE N
     *O I#/)
      NFST=1        $    NB=8
   41 NLST=NPAR
      IF(NPAR.GT.NB)  NLST=NB
      WRITE(3,16) (IP(I),I=NFST,NLST)
      DO 42 I=2,NOCLC
      DO 43 J=1,NPAR
      JV=IP(J)    $    A2A1(J)=XA(I,JV)-XA(I-1,JV)
   43 TOT21(J)=TOT21(J)+A2A1(J)
   42 WRITE(3,17) I,(A2A1(J),J=NFST,NLST)
      WRITE(3,44)   (TOT21(J),J=NFST,NLST)
   44 FORMAT(/1H #TOTAL#E12.5,7E14.5)
      IF(NPAR.LE.NB) GO TO 62
      NFST=NFST+8  $  NB=NB+8  $  WRITE(3,14)  $  GO TO  41
   62 CONTINUE
C
C---------------------OUTPUT SCALE FACTORS---------
C
      WRITE(3,21)  $  WRITE(3,50)
   50 FORMAT(1H 7X#     SCALE FACTORS      #/)
      NFST=1        $    NB=8
   51 NLST=NPAR
      IF(NPAR.GT.NB)  NLST=NB
      WRITE(3,16) (IP(I),I=NFST,NLST)
      DO 52 I=1,NOCLC
   52 WRITE(3,17)  I,(XMA(I,IP(J)),J=NFST,NLST)
      IF(NPAR.LE.NB) GO TO 63
      NFST=NFST+8  $  NB=NB+8  $  WRITE(3,14)  $  GO TO  51
   63 CONTINUE

      IF(NRINSUB(5).EQ.0) GO TO 73
C
C---------------------OUTPUT GEODESIC CURVATURE
C
      WRITE(3,21)  $  WRITE(3,75)    $   WRITE(3,71)
   75 FORMAT(1H 7X#RADIUS OF GEODESIC CURVATURE IN DIRECTION OF SEARCH#
     */1H 7X#+#51(#-#)#+#)
   71 FORMAT(1H #CYCLE#9X#ABSOLUTE#6X#RELATIVE TO /SQA/# 9X#TOTAL COSINE
     *#6X#/SQA/#9X#/DELTA(NPART)/#5X#IN..X#/)
      NOCLC1=NOCLC-1
      DO 72 I=1,NOCLC1
      RGEOCV=RGEOCSQ=9999.999
      IF(XGEO(I).EQ..0) GO TO 72
      RGEOCV=1./XGEO(I)     $    SQM=SQRT(SQMIN(I))
      RGEOCSQ=RGEOCV/SQM
   72 WRITE(3,74) I,RGEOCV,RGEOCSQ,ACOSQ(I),SQM,XDELT(I),NUMBR(I)
   73 CONTINUE
   74 FORMAT(1H I4,4X,F13.3,F17.3,7X,F18.6,E18.5,7X,E12.5,I11)
C
C-------------------OUTPUT COSINES OF NORMAL------
C
   65 CONTINUE
      WRITE(3,21)  $  WRITE(3,30)
   30 FORMAT(1H 7X#PARTIAL COSINES#/)
      NFST=1        $    NB=8
   31 NLST=NPAR
      IF(NPAR.GT.NB)  NLST=NB
      WRITE(3,16) (IP(I),I=NFST,NLST)
      DO 32 I=1,NOCLC
   32 WRITE(3,33) I,(XCOSN(I,IP(J)),J=NFST,NLST)
      WRITE(3,67) NOCLC,(XCOSN(NOCLC,IP(J)),J=NFST,NLST)
   67 FORMAT(/1H I3,E15.4,7E14.4)
   33 FORMAT(1H I3,F12.5,7F14.5)
      IF(NPAR.LE.NB) GO TO 64
      NFST=NFST+8  $  NB=NB+8  $  WRITE(3,14)  $  GO TO  31
   64 CONTINUE
      IF(INDALL.EQ.0) GO TO 320
C
C---------------------OUTPUT FINAL PARAMETER VALUES--
C
      DO 114 I=1,NPAR
      TVALUE(I)=.0  $  IF(STDEV(I).EQ.0.0) GO TO 114
      TVALUE(I)=A(IP(I))/STDEV(I)
  114 CONTINUE
      WRITE(3,10)
   10 FORMAT(/1H1,3X#NAME#10X#FINAL  PARAMETER  VALUES#10X#STANDARD DEVI
     *ATION#27X#T-VALUE#/)
      WRITE(3,113)
  113 FORMAT(1H 10(#*#),4X,30(#*#) ,9X,30(#*#),9X,30(#*#))
      WRITE(3,112)
  112 FORMAT(1H 14X#*#28X#*#9X#*#28X#*#9X#*#28X#*#)
      WRITE(3,111) (NAME(IP(I)),IP(I),A(IP(I)),IP(I),STDEV(I),IP(I),
     *TVALUE(I),I=1,NPAR)
  111 FORMAT(1H A10,4X#*#I5,E20.10#    *#9X#*#I7,E15.5,8X#*#9X#*#I7,F13.2
     *,8X#*#)
      WRITE(3,112)
      WRITE(3,113)
      WRITE(3,115) SQMIN(NOCLC),SERROR,QCOS,XCRLMT(NOCLC)
  115 FORMAT(/1H #MULT.CORR.#7X#SQA=#E19.10,14X#STAND.ERROR=#E16.8,12X#T
     *OTAL COSINE=#E14.6/1H F8.5,//1H 54X#LINEARIZED IN TANGENT PLANE#)
      WRITE(3,312)$WRITE(3,313)$WRITE(3,314)$WRITE(3,315)$WRITE(3,314)
  312 FORMAT(  1H 38X#*#13(#-#)# 95 0/0   CONFIDENCE   INTERVAL #22(#-#)
     *#*#/)
  313 FORMAT(1H 10(#*#)3X,109(#*#))
  314 FORMAT(1H 13X,#*#107X#*#)
  315 FORMAT(1H 13X,#*  K#15X#LOWER BOUND#24X#MEAN VALUE#23X#UPPER BOUND
     *#8X#K *#)
      IFRMT(1)= 8H(1H+T21,
      IFRMT(2)=10H 9X,F20.9,
      IFRMT(3)=10HT52#-#T55,
      IFRMT(5)=10HT86#-#T89,
      IFRMT(6)=10H 9X,F20.9)
      DO 300 I=1,NPAR   $   IV=IP(I)    $   WRITE(3,311) IV,IV
      IF(STDEV(I)) 300,300,302
  311 FORMAT(1H 13X,#*#,I3,99X,I4# *#)
  302 WRITE(3,316) NAME(IV)
  316 FORMAT(1H+A10)
      ALNSTD=ALOG10(STDEV(I))  $  IF(ALNSTD.LT.0.0) ALNSTD=ALNSTD-1.
      NLOG=ALNSTD  $  TEN=10.**(1-NLOG)
      IFMT=1-NLOG  $  IF(IFMT.LT.0) IFMT=0  $  IF(IFMT.GT.9) IFMT=9

      BOUND(1)=A(IP(I))-2.*STDEV(I)
      BOUND(2)=A(IP(I))
      BOUND(3)=A(IP(I))+2.*STDEV(I)
      DO 304 J2=1,3  $  BOUND(J2)=TEN*BOUND(J2)
                        IF(BOUND(J2)) 305,305,306
  305                   BOUND(J2)=IFIX(BOUND(J2)-0.5)
                        GO TO 304
  306                   BOUND(J2)=IFIX(BOUND(J2)+0.5)
  304                   BOUND(J2)=BOUND(J2)/TEN

      ENCODE(10,309,IFRMT(2))IFMT,IFMT
  309 FORMAT(1H +I1,6HX,F20.,I1,1H,)
      ENCODE(10,310,IFRMT(6)) IFMT,IFMT
  310 FORMAT(1H +I1,6HX,F20.,I1,1H))
      IFRMT(4)=IFRMT(2)

      WRITE(3,IFRMT) (BOUND(J3),J3=1,3)
  300 CONTINUE
      WRITE(3,314)  $  WRITE(3,313)
C
  320 IF(NYS2.EQ.4H  **) MAXCYCL=1000
      CPTIME=SECOND(FS)  $  EXTIME(11)=EXTIME(11)+FS-ST
      RETURN

C*****     *****     *****     *****     *****     *****     *****     *****
      ENTRY PLOT
C---------------------PLOT CURVE--------------------
C
      CPTIME=SECOND(ST)    $   NRINSUB(13)=NRINSUB(13)+1
      IMAP=1
      KB=1H  $  ISYMBOL(1)=1H*  $  ISYMBOL(2)=1HO
      RANGE=TOP(IMAP)-BOTTOM(IMAP) $ SCALE=100./RANGE $ UNIT=RANGE/5.
      BTO=BOTTOM(IMAP)-UNIT
      DO 205 I=1,6
      BTO=BTO+UNIT
  205 ISCALE(I)=BTO

      WRITE(3,201)  $  WRITE(3,202)
  201 FORMAT(1H1//)
  202 FORMAT(1H #SCALE  VALUE#T26#,#6(19X#,#))

      WRITE(3,203) (ISCALE(I),I=1,6)
  203 FORMAT(1H I6,5I20))
      WRITE(3,204)  $  WRITE(3,204)  $  WRITE(3,204)
  204 FORMAT(1H 4X#,#5(19X#,#))
      WRITE(3,206)
  206 FORMAT(1H # NR +#101# . . . . +#)
     * #  NR#5X#YOBS=*#7X#YCALC=O#
     *)

      IPLOT=1  $  LAATST=NDATA+11
      DO 250 I=1,LAATST
      J=I-(I/10)*10           $  IF(J.EQ.0) 210,211
  210 WRITE(3,212)            $ GO TO 220
  212 FORMAT(1H 4X#+#101(# . . . . +#))
  211 J=I-(I/5)*5             $ IF(J.EQ.0) 207,208
  207 WRITE(3,209)            $ GO TO 220
  209 FORMAT(1H 4X#+#5(19X#+#))
  208 WRITE(3,204)
  220 IF(IPLOT.GT.1) GO TO 250
      YPLOT(1)=-X(I,1)  $  YPLOT(2)=-YCLCA(I)
      WRITE(3,214) I,I,YPLOT(1),YPLOT(2)
  214 FORMAT(1H+I3,T107,I3,F11.0,F13.2)

      DO 251 J=1,2
      YPLOT(J)=(YPLOT(J)-BOTTOM(IMAP))*SCALE
      IF(YPLOT(J).LT.0.0)  YPLOT(J)=YPLOT(J)-1.0
      YPLOT(J)=YPLOT(J)+0.5
      NBLANC=YPLOT(J)+6.
      IF(NBLANC.LT.3) NBLANC=3  $  IF(NBLANC.GT.105) NBLANC=105
      WRITE(3,254) (KB,L,JK=1,NBLANC),ISYMBOL(J)
  254 FORMAT(1H+110A1)
  251 CONTINUE
      IF(I.EQ.NDATA) IPLOT=2
  250 CONTINUE

      WRITE(3,255)
  255 FORMAT(//1H ****END OF PLOT****)
      CPTIME=SECOND(FS)   $   EXTIME(13)=EXTIME(13)+FS-ST
      RETURN
```





```
      CALL FNCTN(XXA,0,SQSAVE,0,1,YCALC,YDIF,0,0)

      AA=-.1
      IF(NTYPE.EQ.5) NDO=NOCLC
      IF(NTYPE.EQ.6) NDO=21

      DO 40 K=1,NDO
      IF(NAUT.EQ.0) GO TO 46
      IF(NTYPE.EQ.5) GO TO 39
      AA=AA+.1
      DO 38 I=1,NPAR
   38 XXA(I)=AA*XA(NOCLC,I)+(1.-AA)*XA(1,I)
      IF(K.EQ.11.OR.K.EQ.12) WRITE(INK7,86)
      GO TO 47

   39 DO 41 I=1,NPAR
   41 XXA(I)=XA(K,I)
      GO TO 47

   46 AA=AA+.1
      DO 48 I=1,NPAR
   48 XXA(I)=AA*XA(NOCLC,I)+(1.-AA)*A(I)
      IF(K.EQ.11.OR.K.EQ.12) WRITE(INK7,86)

   47 CONTINUE

      CALL FNCTN(XXA,0,SQA,0,1,YCLCA,YOCA,0,0)

      S1=S2=S12=.0
      DO 43 I=1,NDATA
      YCCA=YCALC(I)-YCLCA(I)
      S1=S1+YDIF(I)**2
      S2=S2+YCCA**2
   43 S12=S12+YDIF(I)*YCCA
      DH=S12/S1
      HEIGHT=.0
      DO 44 I=1,NDATA
   44 HEIGHT=HEIGHT+((1.+DH)*YDIF(I))**2
      SHEIGHT=SQRT(HEIGHT)
      ABOVE=SQRT(SQSAVE)
      BELOW=ABOVE-SHEIGHT
      BELOW2=BELOW**2
      LENGTH=S2-BELOW2
      SLENGTH=SQRT(LENGTH)
      PROC=100.*BELOW/ABOVE
      IF(LENGTH.NE..0) GO TO 52
      RATIO=.0   $   GO TO 68
   52 RATIO=100.*SLENGTH/ABOVE

   68 BB=AA-1.
      IF(NTYPE.EQ.5) WRITE(INK7,55) K,SQA,SQSAVE,SLENGTH,BELOW,
     *RATIO,PROC,K
      IF(NTYPE.EQ.6) WRITE(INK7,58)BB,SQA,SQSAVE,SLENGTH,BELOW,
     *RATIO,PROC,BB
   40 CONTINUE
   55 FORMAT(1H I4,1X2E14.5,E17.5,E15.5,2F14.2,I12)
   58 FORMAT(1H F5.2,2E14.5,E17.5,E15.5,2F14.2,F13.2)
      NRINSUB(16)=0
      IF(NTYPE.EQ.5) GO TO 50
      WRITE(INK7,91)
   91 FORMAT(//1H 23X#USE HAS BEEN MADE OF THE FOLLOWING VALUES#/1H #
     *K#8X#NAME#11X#INITIAL POINT#5X#TERMINAL POINT#10X#DIFFERENCE#6X#K#
     */)
      DO 92 K=1,NPAR
      IF(NAUT.EQ.1) AIT=XA(NOCLC,K)-XA(1,K)
      IF(NAUT.EQ.0) AIT=XA(NOCLC,K)-A(K)
      IF(NAUT.EQ.1) WRITE(INK7,93) K,NAME(K),XA(1,K),XA(NOCLC,K),AIT,K
      IF(NAUT.EQ.0) WRITE(INK7,93) K,NAME(K),A(K),XA(NOCLC,K),AIT,K
   92 CONTINUE
   93 FORMAT(1H I4,5XA10,2X2E19.8,E20.5,I7)
      GO TO 50

    4 IF(N.GT.1) RETURN
      WRITE(3,63) SQMIN(NOCLC),NOCLC
   63 FORMAT(/1H #SQA=#E14.5# IN CYCLE NO.#I3/)
      CALL FNCTN(A,0,SQA,0,1,YCLCA,YOCA,0,0)
      DO 62 I=1,NDATA
   62 X(I,1)=2.*YCLCA(I)-X(I,1)
      DO 65 K=1,NPAR
   65 A(K)=XA(1,K)
      IREP=END-1
      GO TO 50

    3 DO 70 I=1,NDATA
       XY=.0
       DO 71 J=1,NPART
   71 XY=XY+FA(I,J)*DELTA(J)
   70 X(I,1)=2.*(XY+YCLCA(I))-YOBS(I)
      GO TO 50

   50 CONTINUE
       INK7=JNK7
      CPTIME=SECOND(FS)   $   EXTIME(16)=EXTIME(16)+FS-ST
      INK7=IREST   $   RETURN
      END
*DECK LINYPEX
      SUBROUTINE LINYPEX(LME,N1,N2,N3,NCRIT,LIST)
      INTEGER END

*CALL DIMLIPEX
      DIMENSION U(5,5),AIVEC(5)
*CALL DVAR
*CALL DFIX
C
C     FOR APPLICATION USE END.GE.2
C     THE CORR(I) VECTORS USED IN 1200 AND 1300 ARE FIRST DERIVATIVES OF
C     THE PARAMETER FUNCTIONS WITH RESPECT TO THE STEP FACTOR.
C     THESE VECTORS ARE TESTED IN SUBROUTINE MIN ON ZERO LENGTH.
C     FOR NEW CONVERGENCE SIMULATION FORMULAS IT IS RECALLED THAT EPS,
C     SQA, AND A(K) MUST BE DEFINED BEFORE A RETURN TO NLV TAKES PLACE
C
      IF(LME.GE.10) GO TO 1099
      IF(END.GE.2) GO TO 7   $   WRITE(3,8) END
    8 FORMAT(//1H #END=#I3# THIS SHOULD BE ≥ 2.#/1H #REDEFINE END IN MAI
     *N PROGRAM NLV.#//1H #STOP 70 (LINYPEX)#)
      STOP 70
    7 CONTINUE
       INK=7  $  IF(INK7.EQ.3.OR.LIST.EQ.1) INK=3  $  WRITE(3,1)
    1 FORMAT(//1H #***XEQ LINYPEX CONVERGENCE SIMULATION***#)
      IF(LME.EQ.1) GO TO 1100   $   IF(NCYCLS.GE.N3) GO TO 9
      NN=N3
    6 WRITE(3,2) NCYCLS,NN   $   STOP 71
    2 FORMAT(/1H #NCYCLES=#I3# BUT IT SHOULD BE#I3//1H #INCREASE THE VAL
     *UE OF NCYCLES IN THE MAIN PROGRAM NLV#//1H #STOP 71 (LINYPEX)#)
C
    9 E1=EPMIN(N1)   $   E2=.0   $   I1=N1+1
      DO 3 I=I1,N2
    3 E2=E2+EPMIN(I)   $   E2=E2+E1   $   E21=E2-E1
      E3=.0   $   I1=N2+1
      DO 4 I=I1,N3
    4 E3=E3+EPMIN(I)   $   E3=E3+E2   $   E32=E3-E2
      IF(E21.NE..0.AND.E32.NE..0) GO TO 1000
      WRITE(3,5) N1,E1,N2,E2,N3,E3,E21,E32   $   STOP 72
    5 FORMAT(/1H # N1#8X#E1#6X#N2#8X#E2#6X#N3#8X#E3#7X#E21#12X#E32#/1H
     *3(I3,E15.5),2(E15.5)# POINTS ARE COLLINEAR SO E21 AND/OR E32 = 0#
     *//1H #STOP 72 (LINYPEX)#)
 1000 GO TO (1100,1200,1300),LME
 1099 ILME=LME-10   $   GO TO (1199,1299,1399),ILME
C
C
C            4TH-DEGREE SIMULATION OF CONVERGENCE TO FINAL PARAMETER
C            VALUES WITH EQUAL STEPS BASED ON N1 AND N2
C
 1100 CONTINUE
      WRITE(3,101)
  101 FORMAT(1H+50X#+---POLYNOMIAL B(NI)=A+B*NI+C*NI**2+D*NI**3+E*NI**4-
     *--+#/1H 61X#+-----WHERE NI=CYCLE NUMBER---+#/)
      N21=N2-N1   $   N5=4*N21+N1   $   N40=N5-N21   $   N30=N40-N21
      IF(NCYCLS.LT.N5) NN=N5
      IF(NCYCLS.LT.N5) GO TO 6
      U(1,1)=U(1,2)=U(1,4)=U(1,5)=U(2,3)=U(4,3)=.0
      U(5,1)=U(5,5)=1.  $  U(3,1)=U(3,5)=-1.
      U(2,1)=U(4,5)=2.  $  U(4,1)=U(2,5)=-2.
      U(4,2)=4.  $  U(4,4)=U(5,2)=U(5,4)=-4.
      U(2,4)=U(3,2)=U(3,4)=16.  $  U(2,2)=-16.
      U(1,3)=24.  $  U(3,3)=-30.  $  U(5,3)=6.


      WRITE(3,124) N1,N2,N30,N40,N5
  124 FORMAT(/1H 10X#USED PARAMETER VALUES ACCORDING TO CYCLE NUMBER NI#
     */1H 3X#K#I11,4I20)
      DO 115 I=1,NPART
      DO 114 K=1,5
      NNN=N1+(K-1)*N21
  114 AIVEC(K)=XA(NNN,IP(I))
      WRITE(INK,121)  IP(I),(AIVEC(K),K=1,5)
      AA(I)=BB(I)=CC(I)=DD(I)=EE(I)=.0
      DO 116 J=1,5
      AA(I)=AA(I)+U(1,J)*AIVEC(J)
      BB(I)=BB(I)+U(2,J)*AIVEC(J)
      CC(I)=CC(I)+U(3,J)*AIVEC(J)
      DD(I)=DD(I)+U(4,J)*AIVEC(J)
  116 EE(I)=EE(I)+U(5,J)*AIVEC(J)
      AA(I)=AA(I)/24.  $  BB(I)=BB(I)/24.  $  CC(I)=CC(I)/24.  $  DD(I)=
     *DD(I)/24.  $  EE(I)=EE(I)/24.
  115 CONTINUE
      IF(INK.EQ.7) GO TO 122
      WRITE(INK,120) N5
  120 FORMAT(/1H #BACKSOLUTION,NO#I3,13X# NEXT FOUR EQUIDISTANT POINTS
     *ARE#/)
      DO 117 I=1,NPART
      DO 118 K=1,5   $   FN=FLOAT(K*1)
  118 AIVEC(K)=AA(I)+FN*(BB(I)+FN*(CC(I)+FN*(DD(I)+FN*EE(I))))
  117 WRITE(INK,121) IP(I),(AIVEC(K),K=1,5)
  121 FORMAT(1H I4,E16.5,4E20.5)
      WRITE(INK,131) (I,I=1,4)
      WRITE(INK,132) (IP(I),AA(I),BB(I),CC(I),DD(I),EE(I),I=1,NPART)
  131 FORMAT(/1H #POLYNOMIAL PARAMETERS OF DEGREE#/1H 14X#0#,4I20)
  132 FORMAT(1H I4,E18.7,4E20.7)
  122 EPS=1.   $   SQA=SQMIN(N5)
      DO 130 I=1,NPAR
  130 A(I)=XA(N5,I)
      WRITE(INK,123) N5,SQA,EPS
  123 FORMAT(/1H #SQA=SQMIN(#I3# )=#E15.5#   NEW EPS VALUE =#E19.5/)
      GO TO 399
C
 1199 DO 151 I=1,NPART
      IJ=IP(I)   $   FN=2.+EPS
      B(IJ)=   AA(I)+FN*(BB(I)+FN*(CC(I)+FN*(DD(I)+FN*EE(I))))
  151 CORR(I)=    BB(I)+FN*(2.*CC(I)+FN*(3.*DD(I)+FN*4.*EE(I)))
      RETURN
C
C             HYPERBOLIC SIMULATION
C
 1200 WRITE(3,201)   $   WRITE(3,200)
      WRITE(3,202) N1,N2,N3,N1,N2,N3

  201 FORMAT(1H+50X#+---HYPERBOLIC B(N)=A(LIM)+C/(-B+TEPS(N))---+#)
  200 FORMAT(1H 40X#+---WHERE TEPS(N) IS THE SUM OF N EPS VALUES OUT OF
     *N FITTING CYCLES---+#/)
  202 FORMAT(/1H #  K#21X#TEPS(MINIMA)#44X#VALUES OF PARAMETERS USED#/1H
     * 4X#CYCLE=#I5,2I15,22X#CYCLE=#I5,2I15)
C
      DO 215 I=1,NPART
      A1=XA(N1,IP(I))  $  A2=XA(N2,IP(I))  $  A3=XA(N3,IP(I))
      A32=A3-A2   $   A21=A2-A1
      IF(INK.EQ.7) GO TO 203
      WRITE(INK,204) IP(I),E1,E2,E3,A1,A2,A3
  204 FORMAT(1H I3,2X,3E15.5,18X,3E15.5)
  203 IF(A32.EQ..0.OR.A21.EQ..0) GO TO 216
      BN=E21*A32-E32*A21
      IF(ABS(A32/E32).GE.ABS(A21/E21)) GO TO 216
      BB(I)=E21*A32*E3-E32*A21*E1
      BB(I)=BB(I)/BN
      CC(I)=-A32*(E3-BB(I))*(E2-BB(I))/E32
      IF(CC(I).EQ..0) GO TO 216
      AA(I)=A3-CC(I)/(E3-BB(I))
      GO TO 215
  216 AA(I)=A3+DELTA(I)   $   BB(I)=.0   $   CC(I)=-DELTA(I)*E3
  215 CONTINUE
      IF(INK.EQ.7) GO TO 220
      WRITE(INK,206)   $   WRITE(INK,208) N3
  206 FORMAT(//1H #  K      LAST A(K) USED#12X#HYPERBOLIC PARAMETERS#  2#
     *X#BACK SOLUTION B(K)#)
  208 FORMAT(1H 27X#A(LIM)#11X#B#14X#C#3X,I15,10X#NEXT TWO ON DISTANCE N
     *3-N2#)
      DO 230 I=1,NPART
      AAA=AA(I)+CC(I)/(-BB(I)+E3)
      BBB=AA(I)+CC(I)/(-BB(I)+E3+E32)
      CCC=AA(I)+CC(I)/(-BB(I)+E3+2.*E32)
  230 WRITE(INK,207) IP(I),XA(N3,IP(I)),AA(I),BB(I),CC(I),AAA,BBB,CCC
  207 FORMAT(1H I3,2X,4E15.5, 3X,3E15.5)
  220 EPS=E32  $   SQA=SQMIN(N3)
      DO 231 I=1,NPAR
  231 A(I)=XA(N3,I)
      WRITE(INK,123) N3,SQA,EPS
      GO TO 399
```

```
C                                                                       153
 1299 DO 251 I=1,NPART                                                  154
      IJ=IP(I)                                                          155
      B(I,J)=CC(I)/(E3+EPS-BB(I))+AA(I)                                 156
  251 CORR(I)=-CC(I)/((E3+EPS-BB(I))**2)                                157
      RETURN                                                            158
C                                                                       159
C             EXPONENTIAL SIMULATION                                    160
C                                                                       161
 1300 WRITE(3,301)   $   WRITE(3,200)                                  162
      WRITE(3,202) N1,N2,N3,N1+N2+N3                                    163
  301 FORMAT(1H+50X,----EXPONENTIAL B(N)=A1LIN)+C*EXP(-B*TEPS(N))-----*) 164
C                                                                       165
      NCRIT=NCRIT-1   $   DO 315 I=1,NPART                              166
      ITEL(I)=0                                                         167
      A1=XA(N1,IP(I))   $   A2=XA(N2,IP(I))   $   A3=XA(N3,IP(I))     168
      A32=A3-A2   $   A21=A2-A1   $   A31=A3-A1                         169
      IF(INK.EQ.7) GO TO 303                                            170
      WRITE(INK,204) IP(I),E1,E2,E3,A1,A2,A3                            171
  303 IF(A32.EQ.0.OR.A21.EQ.0) GO TO 316                                172
      IF(ABS(A32/E32).GE.ABS(A21/E21)) GO TO 316                        173
C                                                                       174
C             ITERATIVE SOLUTION                                        175
C                                                                       176
      B1=B1=9999999999.                                                 177
C                                                                       178
  325 BT=EXP(-B1*E32)                                                   179
      BT=ALOG((A31-A21*BT)/A32)                                         180
      B2=BT/E2)                                                         181
      ITEL(I)=ITEL(I)+1                                                 182
      POWER=10.**IFIX(ALOG10(B2))                                       183
      IF(POWER.LT..01) POWER=POWER-1.                                   184
      IB2=IFIX((B2/POWER)*10.**NCRIT)                                   185
      IF(IB2-IB1.EQ.0.OR.ITEL(I).EQ.25) GO TO 327                       186
      B1=B2   $   IB1=IB2                                               187
      GO TO 325                                                         188
C                                                                       189
  327 BB(I)=B2                                                          190
      CN=EXP(-BB(I)*EJ)-EXP(-BB(I)*E2)                                  191
      CC(I)=A32/CN                                                      192
      AA(I)=A3-CC(I))*EXP(-BB(I)*EJ)                                    193
      GO TO 315                                                         194
  316 AA(I)=A3+DELTA(I)   $   BB(I)=1.   $   CC(I)=-DELTA(I)*EXP(E3)    195
  315 CONTINUE                                                          196
C                                                                       197
      IF(INK.EQ.7) GO TO 320                                            198
      WRITE(INK,306)   $   WRITE(INK,208) N3   $   WRITE(INK,319) NCRIT 199
  306 FORMAT(//1H #   K      LAST A(K) USED#,11X#EXPONENTIAL PARAMETERS# 29 200
     *X#BACK SOLUTION B(K)*#)                                           201
  319 FORMAT(1H+,115X,#ITERATIONS (NCRIT=#I2#)#)                        202
      DO 330 I=1,NPART                                                  203
      AAA=AA(I)+CC(I)*EXP(-BB(I)*E3)                                    204
      BBB=AA(I)+CC(I)*EXP(-BB(I)*(E3+E32))                              205
      CCC=AA(I)+CC(I)*EXP(-BB(I)*(E3+2.*E32))                           206
  330 WRITE(INK,307) IP(I),XA(N3,IP(I)),AA(I),BB(I),CC(I),AAA,BBB,CCC,  207
     *ITEL(I)                                                           208
  307 FORMAT(1H I3,2X,4E15.9,3X,3E18.5,I10)                             209
  320 EPS=E32   $   SOA=SOMIN(N3)                                       210
      DO 331 I=1,NPAR                                                   211
  331 A(I)=XA(N3,I)                                                     212
      WRITE(INK,123) N3,SOA,EPS                                         213
      GO TO 399                                                         214
C                                                                       215
 1399 DO 351 I=1,NPART                                                  216
      IJ=IP(I)                                                          217
      AUX=CC(I)*EXP(-BB(I)*(E3+EPS))                                    218
      B(I,J)=AUX+AA(I)                                                  219
  351 CORR(I)=-BB(1)*AUX                                                220
      RETURN                                                            221
C                                                                       222
C                                                                       223
  399 WRITE(INK,400)                                                    224
  400 FORMAT(//1H ***LIMYPEX XQO****/)                                  225
      RETURN                                                            226
      END                                                               227
```

```
CREATION RUN              DECK LIST AS WRITTEN. IF NEWPL        UPDATE V1.2        01/17/75  13.46.38.

DECKS ARE LISTED IN THE ORDER OF THEIR OCCURRENCE ON A NEW PROGRAM LIBRARY IF ONE IS CREATED BY THIS UPDATE

        YANKSSS   INITL    READ     FNCTN    OFDA     D2FDA    NRNEQ    SOLVE
        MONA      MIN      AISB     LISTING  BLOCK    BACK     IBACK    LIMYPEX
        A84/8208* OVAR     DFIX     DIMINITL DIMFNCTN DIMDFDA  DIMD2FDA DIMNRNEQ
        DIMSOLVE  DIMMONA  DIMMIN   DIMLIST  DIMBLOCK  DIMBACK  DIMIBACK DIMLIPEX
        EXAMPLE   XINITL   NDIMRD   RDDATA   RTDATA    NDIMFN   NFNCTN   NFNCTN
        NDIMDF    NDFDA    NDIMD2   ND2FDA
```

THIS UPDATE REQUIRED 336800 WORDS OF CORE.

1.4 Condition function dependent parts

1.4.1 statements

```
          CONDECKS TO BE CALLED BY ALL SUBPROGRAMS
          CONDECKS TO BE INSERTED IN MAIN PROGRAM NLV

*DECK A04/029B*
*CONDECK DVAR
      COMMON A( 4),B( 4),NAMB( 4),NAME( 4),CORR( 4),DELTA( 4),STDEV( 4)      1
      COMMON MA( 5),MB( 5),ANORM( 5),BNORM( 5),IP( 5),JP( 5)                 2
      COMMON YCLCA( 20),YCLCB( 20),YOCA( 20),YOCB( 20),YOBS( 20)             3
      COMMON KFIXC( 4),ALB( 4),AUB( 4),IB( 4),FA( 20, 5),FAZ( 5)             4
      COMMON X( 20, 2),XA(15, 4),XMA(15, 4),XCOSN(15, 4),IPX(15, 4)          5
      COMMON IPAR(  1),H( 5, 5),XM( 5, 5),FAA( 20,10),FBFAA( 4, 4)           6
      COMMON FAFAA( 4, 4),FFAANN( 4),FSS( 30),CURVTOT( 4),CURVGED( 4)        7
*CONDECK DFIX
      COMMON EPMIN(15),SQMIN(15),XSQF(15),XDELT(15),XNORM(15),NUMBR(15)      8
      COMMON ENRM(15),XDELTA(15),XALFA(15),XBETA(15),XGAMMA(15),XGEO(15)     9
      COMMON NPARTX(15),XCOSO(15),XCOSMT(15),XCRLMT(15),EPSNEW(16),IC(6)    10
      COMMON NPART,NPAR,NT,MPAR,NDATA,MAXDAT,MAXVEC,NB,BOTTOM(5),TOP(5)     11
      COMMON XYO,XYC,SQA,SQB,EPSEST,NRINSUB(25),EXTIME(25),CPAR(25),NO2     12
      COMMON EPS,EQS,RN,RP,R,COSCRIT,NOYES1,NOYES2,NOYES3,NYS1,NYS2,NYS3    13
      COMMON NOCLC,NCYCLES,NCYCLS,NC,MAXCYCL,IREP,END,LSUMRY,REC1,REC2      14
      COMMON INK3,INK7,INK8,NOUTRO,NOUTFN,NOUTDF,NOUTD2F,NOUTFL,METHD,CK    15
      COMMON TA(5)                                                          16

          CONDECKS TO BE CALLED BY SINGLE SUBPROGRAMS

*CONDECK DIMINITL
      DIMENSION ISTAR( 4)
*CONDECK DIMFNCTN
    . DIMENSION C( 4),YCLC( 20),YOC( 20)
*CONDECK DIMDFDA
      DIMENSION C( 4),M( 5),YOC( 20),YCLC(20),ORDERED( 4),JPERM( 4)
*CONDECK DIMD2FDA
      DIMENSION C( 4),FAA2( 4),M( 5),IL(10),YCLC( 20),YOC( 20)
*CONDECK DIMNAMEQ
      DIMENSION M( 5),CNORM( 5),COSN( 5),YOC( 20)
*CONDECK DIMSOLVE
      DIMENSION C( 4),SOLN( 4),E( 4),QQ( 4),CN( 5),COSINF( 5),D( 5)
*CONDECK DIMNOWA
      DIMENSION D( 4),E( 4),ISTAR( 4),FCALC( 20),FOC( 20),TJFSS( 4)
      DIMENSION DCURVE( 4)
*CONDECK DIMMIN
      DIMENSION KS( 4),FKS( 20), FJS( 20), TJFSS( 4),DCURVE( 4)
      DIMENSION CRCTN( 4),IBN( 4)
*CONDECK DIMLIST
      DIMENSION AZAL( 4),TOTZL( 4),ITALLY( 4),TVALUE( 4)
*CONDECK DIMBLOCK
      DIMENSION IPB(5, 4),JPB(5, 4),ISTER( 4)
*CONDECK DIMBACK
      DIMENSION VDELT( 4),WDELT( 4),G( 5),YBA( 20)
*CONDECK DIMTRACK
      DIMENSION XXA( 4),YCALC( 20),YDIF( 20)
*CONDECK DIMLIPEX
      DIMENSION AA( 4),BB( 4),CC( 4),DD( 4),EE( 4),ITEL( 4)
```

```
          CONDECKS TO BE CALLED BY SINGLE SUBPROGRAMS

*DECK EXAMPLE
*CONDECK XINITL
  200 FORMAT(1H+18X*        SOIL MOISTURE RETENTION CURVE         *)
      WRITE(3,201)
  201 FORMAT(1H *                          NONE*)
  100 FORMAT(/1H *SUBJECT*/1H *********
     */1H *SOIL MOISTURE RETENTION CURVE, JANUARY 1975*
     */)
  101 FORMAT(/1H *COMMENTS*/1H **********
     */1H *COMMON DECK A04/029B,*
     */1H0*VARIABLES   Y = MOISTURE CONTENT IN PERCENTAGES*
     */1H *            X = LOGARITHM OF THE MOISTURE TENSION*
     */1H *FORMULA BY FINK AND JACKSON (1973) SOIL SC. VOL.116 NO.4 P.24
     *9*/1H *APPLIED TO DATA FROM ICW-LAB. WAGENINGEN*
     */)
  102 FORMAT(/1H *FORMULA*/1H *********
     */1H *               -C                              *
     */1H * X = A+B*LN((Y/D)   -1)*
     *//1H *WHICH IS SOLVED FOR Y BY THE FORMULA*
     *//1H *             (X-A)/B       -1/C*
     */1H *Y = D * ( E             + 1 )*
     */1H0*PARAMETERS TO BE FITTED*
     */1H *  C(1) = D  =  PORE SPACE*
     */1H *  C(2) = A  =  LOCATION*
     */1H *  C(3) = B  =  SCALE *
     */1H *  C(4) = C  =  EXPONENT*
     */)
      NAME(1)=10HPORE SPACE
      NAME(2)=10HLOCATION A
      NAME(3)=10HSCALE    B
      NAME(4)=10HEXPONENT C
*CONDECK NDIMRD
C-----RESERVED FOR DIMENSION-----
*CONDECK RDDATA
      READ(2,2) (YOBS(J),I=1,9)
    2 FORMAT(10F8.5)
      X(1,2)=0.4   $   X(2,2)=1.0   $   X(3,2)=1.5   $   X(4,2)=2.0
      X(5,2)=2.3   $   X(6,2)=2.7   $   X(7,2)=3.4   $   X(8,2)=4.2
      X(9,2)=6.0   $   NDATA=9
*CONDECK WTDATA
      WRITE(3,15)
   15 FORMAT(/1H010X*DATA READ IN SUBROUTINE READ*/
     */1H 10X*I*6X*YOBS*9X*X*
     */1H 19X*V*9X*PF*
     */)
      WRITE(3,33)
   33 FORMAT(1H )
      DO 200 I=1,LAST1
  200 WRITE(3,202) I,YOBS(I),X(I,2)
  202 FORMAT(1H I11,F11.2,F11.2)
      WRITE(3,20)
   20 FORMAT(1H ***/1H ***/1H ***)
      N=NDATA
      WRITE(3,202) N,YOBS(N),X(N,2)
*CONDECK NDIMFN
C-----RESERVED FOR DIMENSION-----
*CONDECK NFNCTN
      DO 200 I=1,NDATA
      Q=(X(I,2)-C(2))/C(3)      $  E=EXP(Q)
      T=-1./C(4)                $  E1=E+1.
      E1T=E1**T
      YCLC(I)=C(1)*E1T
  200 CONTINUE
*CONDECK WFNCTN
      CONTINUE
*CONDECK NDIMDF
C-----RESERVED FOR DIMENSION-----
*CONDECK NDFDA
      K1=JP(1)   $  K2=JP(2)   $  K3=JP(3)   $  K4=JP(4)
      DO 200 I=1,NDATA
      Q=(X(I,2)-C(2))/C(3)      $  E=EXP(Q)
      T=-1./C(4)                $  E1=E+1.
      E1T=E1**T                 $  E1T1=E1T/E1
      FA(I,K1)=E1T
      FA(I,K2)=C(1)/C(3)/C(4)*E*E1T1
      FA(I,K3)=C(1)/C(3)/C(4)*Q*E*E1T1
      FA(I,K4)=C(1)/C(4)/C(4)*ALOG(E1)*E1T
  200 CONTINUE
*CONDECK NDIMD2
C-----RESERVED FOR DIMENSION-----
*CONDECK ND2FDA
      K1=IL(1)   $  K2=IL(2)   $  K3=IL(3)   $  K4=IL(4)   $ K5=IL(5)
      K6=IL(6)   $  K7=IL(7)   $  K8=IL(8)   $  K9=IL(9)   $  K10=IL(10)
      DO 200 I=1,NDATA
      Q=(X(I,2)-C(2))/C(3)      $  E=EXP(Q)
      T=-1./C(4)  $  T1=T-1.    $  E1=E+1.
      E1T=E1**T                 $  E1T1=E1T/E1
      FAA(I,K1)=0.              $  QC=C(1)/C(3)/C(3)/C(4)
      FAA(I,K2)=1./C(3)/C(4)*E*E1T1
      FAA(I,K3)=Q*FAA(I,K2)
      FAA(I,K4)=1./C(4)/C(4)*ALOG(E1)*E1T
      FAA(I,K5)=-QC*E1T1*((-1./C(4)-1.)*E*E/E1+E)
      FAA(I,K6)=Q*FAA(I,K5)-QC*E*E1T1
      FAA(I,K7)=C(1)/C(3)/C(4)/C(4)*E*E1T1*(ALOG(E1)/C(4)-1.)
      FAA(I,K8)=-QC*E1T1*E*Q*(E*Q*T1/E1+Q+2.)
      FAA(I,K9)=Q*C(1)/C(3)/C(4)*E*E1T1*((ALOG(E1)/C(4)-1.)/C(4))
      FAA(I,K10)=C(1)*E1T*ALOG(E1)*(ALOG(E1)-2.*C(4))/C(4)**4
  200 CONTINUE
END OF RECORD

          STARTING VALUES FOR THE PARAMETERS

      A(1)=34.8  $  A(2)=1.31  $  A(3)=.2746  $  A(4)=3.489

          FIRST PARAMETER REDEFINED IN MAIN PROGRAM NLV BY

      CALL READ (1,1,NOUTRD)
      A(1)= YOBS(1)+.1                                                      1
      IREP = 3

          DATA TO DEMONSTRATE FAST CONVERGENCE

 453     434     410     333     276     232     115     74     26      11-1

          DATA TO DEMONSTRATE SLOW CONVERGENCE

 383     361     348     323     290     241     172     114     35     811-2
```

1.4.2 default output

```
$$$$$$$  $$    $$      $$   $$$$$$   $$$$$$  $$$$$$$  $$$$$$$
$$    $$ $$    $$     $$$  $$    $$ $$    $$ $$    $$ $$    $$
$$    $$ $$    $$    $$$$  $$    $$ $$    $$ $$    $$ $$    $$
$$    $$ $$$$$$$$   $$ $$  $$    $$ $$    $$ $$    $$ $$$$$$$
$$    $$ $$    $$  $$  $$  $$    $$ $$    $$ $$    $$ $$    $$
$$$$$$$  $$    $$ $$$$$$$$  $$$$$$$ $$    $$ $$    $$ $$    $$
$$  $$   $$    $$      $$        $$ $$    $$ $$    $$ $$    $$
$$   $$  $$    $$      $$  $$    $$ $$    $$ $$    $$ $$    $$
$$    $$ $$    $$      $$   $$$$$$   $$$$$$  $$$$$$$  $$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$                                                             $
$                                                             $
$                 SOIL MOISTURE RETENTION CURVE               $
$                                                             $
$                                                             $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


*                                                              *
***         CONTROL PARAMETERS USED IN FUNCTION, IF ANY      ***
*                                                              *


         NONE

                      VALUES OF SYSTEM PARAMETERS    DEFAULT   /   USED IN THIS JOB
=============================================================================================================
                                                                             IC(1)=10LIBLANKI   /
NCYCLES=10  /  10       NDATA  =5HCOUNT  /   X         EPS    = .001   /  .001000   IC(2)=10L          /
END     = 1  /   1                                     EPSEST= .001   /  .001000    IC(3)=10L          /
                        NOUTRD =0(NDATA) /   0                                      IC(4)=10L          /
INK3    = 7  /   7      NOUTFN =    1    /   1         EOS    = .66666 /  .666660   IC(5)=10L          /
INK7    = 7  /   7      NOUTDF =    1    /   1                                      IC(6)=10L          /
INK8    = 7  /   7      NOUTD2F=    1    /   1         AN     = .1     /  .100000
                        NOUTFL =5HNDATA  /   X         AP     = .1     /  .100000
                                                                            REC1 =   0.0    /      0.0
LSUMRY  = 1  /   1                                                           REC2 =   0.0    /      0.0
                        COSCRIT= 1000.    / .1000E+04  R      =1.      / 1.000000
=============================================================================================================

IF NDATA = X    DATA   WILL BE COUNTED IN READ
IF NOUTRD= 0    NOUTRD WILL AUTOMATICALLY BE EQUALLED TO MIN(NDATA,50)       MATRIX N02 NOT USED     IN02= 0 / 0 )
IF NOUTFL= X    NOUTFL WILL AUTOMATICALLY BE EQUALLED TO MIN(NDATA,50)

NO BOUNDS HAVE BEEN DEFINED BEFORE CALL HEADING

NO BLOCKS HAVE BEEN DEFINED


                                                                            USED FOR
      SUBJECT                          DATE          TIME        CP-SEC
                                                                         COMPILATION/LOADING
NONLINEAR PARAMETER OPTIMIZATION       01/17/75      12.46.27.    2.665

SUBJECT
*******
SOIL MOISTURE RETENTION CURVE, JANUARY 1975

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

COMMENTS
********
COMMON DECK A04/0208.

VARIABLES   Y = MOISTURE CONTENT IN PERCENTAGES
            X = LOGARITHM OF THE MOISTURE TENSION
FORMULA BY FINK AND JACKSON (1973) SOIL SC. VOL.116 NO.4 P.249
APPLIED TO DATA FROM ICW-LAB, WAGENINGEN


FORMULA
*******
                   -C
 X = A+B*LN((Y/D)    -1)

WHICH IS SOLVED FOR Y BY THE FORMULA

          (X-A)/B       -1/C
Y = D * ( E        + 1 )

PARAMETERS TO BE FITTED
  C(1) = D  =  PORE SPACE
  C(2) = A  =  LOCATION
  C(3) = B  =  SCALE
  C(4) = C  =  EXPONENT

PERMUTATION OF THE FIRST NPAR PARAMETERS

 K=   1   2   3   4
IP=   1   2   3   4
    ================


TOTAL NUMBER OF PARAMETERS    NPAR =   4
TOTAL NUMBER TO BE SOLVED     NPAR =   4
TOTAL NUMBER TO BE FITTED     NPART=   4

TOTAL NUMBER OF DATA          NDATA=   X

NUMBER OF DATA WILL BE COUNTED IN SUBROUTINE READ IF NDATA=X

***INITL XEQD***

***XEQ READ***


          DATA READ IN SUBROUTINE READ

          I        YOBS          X
                     Y          PF


          1        45.30        .40
          2        43.40       1.00
          3        41.00       1.50
          4        33.30       2.00
          5        27.60       2.30
          6        23.20       2.70
          7        11.50       3.40
          8         7.40       4.20
          9         2.40       6.00
*
*
*
          9         2.40       6.00
```

```
+-----------------------------------------------+     +-------------------------------------------------------------+
    INITIAL VALUES PARAMETERS  AND NAME                                  PERMUTED               NAME
                                                                         AND PARTIAL (*)
                                                       TO BE FITTED

 1        .34800E+02      PORE SPACE     1                   1          .34800E+02  *      PORE SPACE      1
 2        .13100E+01      LOCATION A     2                   2          .13100E+01  *      LOCATION A      2
 3        .27460E+00      SCALE    B     3                   3          .27460E+00  *      SCALE    B      3
 4        .34890E+01      EXPONENT C     4                   4          .34890E+01  *      EXPONENT C      4
+-----------------------------------------------+     +-------------------------------------------------------------+

***READ XEQ***
************************************************     **************************************************************

*************START OF CYCLE NO   1  ***



        COS-MATRIX FOR   4  PARAMETERS AND PARTIAL COSINES COS(F0,FK)

              1             2            3         4
      PORE SPACE   LOCATION A   SCALE    B   EXPONENT C
  1      1.00000
  2       .73067      1.00000
  3       .32608       .74548      1.00000
  4       .54673       .89157       .95855     1.00000
   NORM   .53113       .88493       .95845      .99370


CORRELATION MATRIX OF ESTIMATES, OBTAINED FROM THE INVERSE OF H, FOR NPART=  4

              1             2            3         4
      PORE SPACE   LOCATION A   SCALE    B   EXPONENT C
  1      1.00000
  2       .24854      1.00000
  3       .72973       .74150      1.00000
  4      -.67771      -.83224      -.98406     1.00000

DIAGONAL  .18519E+01   .12134E-01   .76968E-02   .19776E+01


          LINEAR REDUCTION OF SQB BY APPLICATION OF DELTA CORRECTIONS TO L PARAMETERS

 K           A             REDUCED SQB       REDUCTION        NAME       K    L      REDUCTION/DELTA

                          .56468838E+03
 1     .45400000E+02      .40533392E+03    .15927446E+03  PORE SPACE   1    1         .63406E+03
 2     .13100000E+01      .10631131E+03    .29902261E+03  LOCATION A   2    2         .39222E+04
 3     .27460000E+00      .74600904E+01    .98851217E+02  SCALE    B   3    3         .16122E+04
 4     .34890000E+01      .62798688E+01    .11802217E+01  EXPONENT C   4    4         .77253E+00


***************************************************************************************************************************
*** NOW A *** AFTER CYCLE NO   1  ***          X(I,1)= YA(I) + 1.00000 * YOC(I)          NOWA NO=  1  ***          MULT.CORR.=  .961195  ***
                                                                                    *** END=  1  ***
**********
I  SUMMARY OF SOLUTIONS
I  SOLVED**
**********        *
I     1    .25E+00  .13E+01 -.35E+01  .76E+01
I     2    .76E-01  .18E+00  .65E+00
I     3    .61E-01  .16E+00
I     4    .15E+01
**********


                    RESULT OF SOLVING NRMEQS                                          LENGTH OF VECTORS

                              YOBS
                               *                                                   /YOBS/= .90685E+02
                              **
                             * *
           SQA= .56461E+03 *   * SQF= .62799E+01               MULT.COSINE = .97503          /SQA/= .23761E+02
                            *  *
                           *   *
                    ******************* TANGENT PLANE                               /YCALC/= .75771E+02
                             A   F
                       SQAF= .55833E+03
                  FCALC**2= .55833E+03

              TOTAL COSINE = .99442
                                                                                 STANDARD ERROR= .1062644112E+02
                                                                                      SQA      = .5646083793E+03
                                                                                 NPARTX(  1 ) =    4

**** COS(NORMAL,DELTA) FOR NPART =   .1365078   ****   TAN(EPS=0,SQA,1) =   -.11167E+04   **** EPS(PREDICTED) =      .8334329E+00  ****
                                                                                          **** (NO2 = 0 GIVES =      .1000000E+01) ****
                                     PARTIAL   PARTIAL
 K          A(K)          USED       COS A     COS F      DELTA-VECTOR        NORMAL          MA*DELTA       MA*NORM         MA(K)        K

 1     .45400000000E+02    *        .53113    .00000    .25119840E+00    .21062901E+02    .41924E+00    .35153E+02    .16690E+01    1
 2     .13100000000E+01    *        .88493    .00000    .76237523E-01    .81556065E+03    .29569E+01    .31632E+05    .38786E+02    2
 3     .27460000000E+00    *        .95845    .00000    .61315276E-01    .24973377E+04    .67236E+01    .27385E+06    .10966E+03    3
 4     .34890000000E+01    *        .99378    .00000    .15277276E+01    .22107100E+03    .14303E+02    .20697E+04    .93628E+01    4

***END OF CYCLE NO   1  ***
***************************************************************************************************************************
```

*************LAST CYCLE, NO 7 ***

COS-MATRIX FOR 4 PARAMETERS AND PARTIAL COSINES COS(F0,FK)

	1 PORE SPACE	2 LOCATION A	3 SCALE B	4 EXPONENT C
1	1.00000			
2	.73628	1.00000		
3	.28658	.70369	1.00000	
4	.56725	.89585	.93525	1.00000
NORM	.00002	.00002	.00001	.00002

CORRELATION MATRIX OF ESTIMATES, OBTAINED FROM THE INVERSE OF N, FOR NPART= 4

	1 PORE SPACE	2 LOCATION A	3 SCALE B	4 EXPONENT C
1	1.00000			
2	.31614	1.00000		
3	.74744	.77386	1.00000	
4	-.69255	-.86582	-.98038	1.00000
DIAGONAL	.13740E+01	.19272E-01	.11743E-01	.16980E+01

LINEAR REDUCTION OF SQB BY APPLICATION OF DELTA CORRECTIONS TO L PARAMETERS

K	A	REDUCED SQB	REDUCTION	NAME	K	L	REDUCTION/DELTA
		.59948760E+01					
1	.45443522E+02	.59948760E+01	.26546161E-08	PORE SPACE	1	1	-.11297E-02
2	.17608409E+01	.59948760E+01	.25892177E-10	LOCATION A	2	2	-.58050E-05
3	.37405762E+00	.59948760E+01	.67302608E-10	SCALE B	3	3	-.18761E-04
4	.34944376E+01	.59948760E+01	.12543637E-08	EXPONENT C	4	4	.27180E-04

****

*** HOW A *** AFTER CYCLE NO 7 *** X(I,1)= YA(I) + 1.00000 * YOC(I) HOWA NO= 1 *** MULT.CORR.= .998574 ***
*** END= 1 ***

```
**********
I  SUMMARY OF SOLUTIONS
I  SOLVED**
**********     *
I     1   -.23E-05  .26E-04  .23E-04  .26E-04
I     2   -.45E-05 -.20E-06  .23E-06
I     3   -.36E-05  .18E-06
I     4    .46E-04
**********
```

```
              RESULT OF SOLVING NRNEQS                          LENGTH OF VECTORS

                          YOBS                                  /YOBS/= .90655E+02
                            *
                           **
                          * *
    SQA= .59949E+01 *      * SQF= .59949E+01       MULT.COSINE = .99964      /SQA/= .24484E+01
                        A    *
                       *     *
          ******************** TANGENT PLANE                    /YCALC/= .90622E+02
                       A     F
                   SQAF= .40021E-08
                FCALC**2= .40021E-08

        TOTAL COSINE = .00003                          STANDARD ERROR= .1094977262E+01
                                                             SQA      = .5994876018E+01
                                                             NPARTX( 7 ) =   4
```

**** COS(NORMAL,DELTA) FOR NPART = .0379987 **** TAN(EPS=0,SQA,1) = -.80042E-08 **** EPS(PREDICTED) = .1830449E+01) ****
**** (NO2 = 0 GIVES = .9999939E+00) ****

K	A(K)	USED	PARTIAL COS A	PARTIAL COS F	DELTA-VECTOR	NORMAL	HA*DELTA	HA*NORM	HA(K)	K
1	.4544352232E+02	*	.00002	-.00000	-.23498833E-05	.10274583E-03	-.46860E-05	.20489E-03	.19942E+01	1
2	.1760840947E+01	*	.00002	-.00000	-.44603086E-05	.13543869E-02	-.14600E-03	.44332E-01	.32732E+02	2
3	.3740576162E+00	*	.00001	-.00000	-.35873505E-05	.17723657E-02	-.26967E-03	.13323E+00	.75173E+02	3
4	.3494437595E+01	*	.00002	0.00000	.46150583E-04	.36061625E-03	.42483E-03	.33196E-02	.92054E+01	4

***END OF CYCLE NO 7 ***

****

************** ******
SUMMARY NO 1 END= 1

K	INITIAL VALUE A(K)	DIFFERENCE B(K)-A(K)	FINAL VALUE B(K)	NAME	K	B/A	CONTROLLED	K
1	.454000E+02	.435223E-01	.454435E+02	PORE SPACE	1	.1001E+01	0	1
2	.131000E+01	.450841E+00	.176084E+01	LOCATION A	2	.1344E+01	0	2
3	.274600E+00	.994576E-01	.374058E+00	SCALE B	3	.1362E+01	0	3
4	.348900E+01	.543759E-02	.349444E+01	EXPONENT C	4	.1002E+01	0	4

TOTAL NUMBER OF USED DATA NDATA= 9
TOTAL NUMBER OF FITTED PARAMETERS NPART= 4

FUNCTION EVALUATION WITH FINAL PARAMETER VALUES (IK)

ANALYSIS OF OBSERVED AND CALCULATED FUNCTION VALUES M.E.=SUM/NDATA S.E.=STANDARD ERROR X1=YA+BETHA*YOC V=X1-YCALC T=V/S.E.

BETHA= 1.0000 95 0/0 CONFIDENCE INTERVAL FOR F(B) REMAINDER X1-YCALC ANALYSIS

	AZ	X1= Y USED	Y LOWER BOUND	YCALC	Y UPPER BOUND		V	T	S.E(F(B))	
1	.40000E+00	.45300E+02	.43172E+02	.45197E+02	.47843E+02	-.205E+00	.19287E+00	.18	.9677E+00	1
2	.10000E+01	.43400E+02	.42513E+02	.43873E+02	.45232E+02	-.394E+00	-.47263E+00	-.43	.8798E+00	2
3	.15000E+01	.41000E+02	.38769E+02	.40481E+02	.42194E+02	-.786E+00	.91883E+00	.47	.8563E+00	3
4	.20000E+01	.33300E+02	.32152E+02	.33524E+02	.34896E+02	-.169E+01	-.22368E+00	-.20	.6860E+00	4
5	.23000E+01	.27600E+02	.26914E+02	.28310E+02	.29706E+02	-.211E+01	-.71012E+00	-.65	.6979E+00	5
6	.27000E+01	.23200E+02>U0	.22802E+02	.21664E+02	.20545E+02	-.232E+01	.15364E+01	1.40	.6998E+00	6
7	.34000E+01	.11500E+02<L0	.11574E+02	.12922E+02	.14269E+02	-.218E+01	-.14215E+01	-1.30	.6737E+00	7
8	.42000E+01	.74000E+01	.55256E+01	.70287E+01	.85307E+01	-.172E+01	.37135E+00	.34	.7510E+00	8
9	.60000E+01	.24000E+01	.85367E+00	.17742E+01	.26947E+01	-.726E+00	.62582E+00	.57	.4803E+00	9

NDATA= 9 *---- 2 (= 22.2 0/0) DATA OUTSIDE INTERVAL + = .32653E+01
NPART= 4 - = -.26279E+01
D.F. = 5 SUM= .61744E+00
TO BE NEAR ZERO S.E.= .10949E+01
M.E.= .46382E-01 S.E.(M.E.)= .38699E+00 RATIO= .1271 <---------- MEAN/V/= .67479E+00

SECOND DERIVATIVE MATRIX NO2 NOT USED IN 6/2=(N-NO2)

ESTIMATE F(B) = YCALC
2ND ORDER TERM = FAA=INV(N)*VAR/2
= REMAINDER
STAND.ERROR F(B) = SQRT(FAA=INV(N)*VAR)

ANALYSIS OF T=V/S.E. (EXPECTATIONS ARE USED IN HIGHER MOMENTS) NDATA= 9

	EXPECTATION	ESTIMATE	STAND.ERROR	(ES-EA)/SE	CHARACTERISTICS
MEAN	0	.42359E-01	.33279E+00	.13	ZERO
VARIANCE	1	.10900E+01	.50000E+00	0.60	UNITY
SKEWNESS;GAMMA(1)	0	.24779E+00	.71714E+00	.35	NORMAL
KURTOSIS;GAMMA(2)	0	.65217E+00	.13997E+01	.47	NORMAL

	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
HISTOGRAM OF T	) 0)	0)	0)	0)	0)	1)	1)	2)	3)	1)	1)	0)	0)	0)	0)	0)	0)	0)
RELATIVE TO 100 0/0	(0.0 (	0.0 (	0.0 (	0.0 (	11.1 (	11.1 (	22.2 (	33.3 (	11.1 (	11.1 (	0.0 (	0.0 (	0.0 (	0.0 (	0.0 (	0.0 (	0.0 (	0.0 (
CLASS LIMITS	-3.0	-2.5	-2.0	-1.5	-1.0	-0.5	-0+	+0.5	+1.0	+1.5	+2.0	+2.5	+3.0					
NORMAL DISTRIBUTION	(0.1 (	0.5 (	1.7 (	4.4 (	9.2 (	15.0 (	19.1 (	19.1 (	15.0 (	9.2 (	4.4 (	1.7 (	0.5 (	0.1 (				
FOR NDATA= 9	) 0.)	0.)	0.)	0.)	1.)	1.)	2.)	2.)	1.)	1.)	0.)	0.)	0.)	0.)				

KOLMOGOROV-SMIRNOV ONE-SAMPLE 95 0/0 TWO-TAILED TEST FOR CUMULATIVE DISTRIBUTION OF T=V/S.E.. NDATA= 9. CRITICAL VALUE= 43.2

UPPER BOUND	43.20	43.20	43.20	43.20	43.20	54.31	65.42	87.64	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
DISTR. OF T	0.00	0.00	0.00	0.00	0.00	11.11	22.22	44.44	77.78	88.89	100.00	100.00	100.00	100.00	100.00	100.00
LOWER BOUND	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.24	34.58	45.69	56.80	56.80	56.80	56.80	56.80	56.80
NORMAL DISTRIB.	.00	.13	.62	2.28	6.68	15.87	30.85	50.00	69.15	84.14	93.32	97.72	99.38	99.86	100.00	

OUTSIDE INTERVAL = 0

NEGATIVE T VALUES = 44.4 0/0
POSITIVE T VALUES = 55.6 0/0

*****COLLECTION OF RESULTS AFTER 7 CYCLES *** SUMRY NO 1 *** TOTAL NO OF SUMRIES TO BE PRODUCED 1 *** DATE 01/17/75 *****

INITIAL PARAMETER VALUES NAME

1	.4548000000E+02	PORE SPACE	
2	.1310000000E+01	LOCATION A	
3	.2746000000E+00	SCALE B	
4	.3489000000E+01	EXPONENT C	

EPS TO OBTAIN NEXT CYCLE	CYCLE NO	MIN FOUND IN=.X	SQUARES TO BE EQUAL SOA	SOF	LENGTHS TO BE ZERO NORMAL	DELTA	V(SQA-SQF)	TOTAL LENGTHS NORMAL	DELTA
0.	1	0	.5044E+03	.6279E+01	.26365E+04	.15913E+01	.23629E+02	.26365E+04	.15913E+01
.18753E+01	2	16	.2596E+02	.6340E+01	.91814E+02	.27493E+01	.44304E+01	.9181+E+02	.27493E+01
.48845E+00	3	8	.1384E+02	.6642E+01	.96418E+02	.53282E+00	.27939E+01	.9641BE+02	.53282E+00
.18094E+01	4	7	.6025E+01	.5995E+01	.35979E+01	.13349E+00	.17091E+00	.35979E+01	.13349E+00
.91605E+00	5	6	.5996E+01	.5994E+01	.24856E+01	.22799E-01	.41029E-01	.24856E+01	.22799E-01
.16482E+01	6	6	.5994E+01	.5994E+01	.28423E-01	.16281E-02	.13245E-02	.20423E-01	.16281E-02
.10659E+01	7	6	.5994E+01	.5994E+01	.22619E-02	.45644E-04	.63263E-04	.22619E-02	.46564E-04
	TOTAL	49							

EPS(PREDICTED) NO	EPS (OPTIMUM)	EPS(OPT) TOTAAL	RESULT OF CYCLE-NO	FITTING MULT.CORR	PROCEDURE MULT.COS	STANDARD ERROR OF ESTIMATE NO	NDATA	D.F.	STAND.ERROR	TOTAL COSINE CYCLE-NO	COSINE
0. 1	0.	0.	1	.9612	.9758	1	9	5	18.62844	1	.994623
.83343E+00 2	.10753E+01	.10753E+01	2	.9958	.9986	2	9	5	2.27890	2	.869385
.93021E+00 3	.48845E+00	.15638E+01	3	.9976	.9994	3	9	5	1.66425	3	.798777
.99498E+00 4	.10094E+01	.25732E+01	4	.9986	.9996	4	9	5	1.09773	4	.869628
.16666E+01 5	.91605E+00	.34892E+01	5	.9986	.9996	5	9	5	1.09313	5	.816755
.97870E+00 6	.10482E+01	.45374E+01	6	.9986	.9996	6	9	5	1.09498	6	.880541
.18686E+01 7	.10659E+01	.56033E+01	7	.9986	.9996	7	9	5	1.09498	7	.880926

PARAMETER VALUES

CYCLE	PORE SPACE 1	LOCATION A 2	SCALE B 3	EXPONENT C 4
1	.45480E+02	.13108E+01	.27460E+00	.34890E+01
2	.45670E+02	.13920E+01	.34053E+00	.51318E+01
3	.45651E+02	.15871E+01	.38221E+00	.38040E+01
4	.45438E+02	.17793E+01	.38632E+00	.33491E+01
5	.45447E+02	.17632E+01	.37579E+00	.34695E+01
6	.45465E+02	.17810E+01	.37416E+00	.34938E+01
7	.45444E+02	.17608E+01	.37408E+00	.34944E+01

TABLEAU OF PARTIAL SOLUTIONS OUT OF CYCLE NO 1

CYCLE NO	PART SOLN	1	2	3	4
1	4	X	X	X	X
2	4	X	X	X	X
3	4	X	X	X	X
4	4	X	X	X	X
5	4	X	X	X	X
6	4	X	X	X	X
7	4	X	X	X	X
OCCURRENCE		7	7	7	7

DIFFERENCES OF PARAMETER VALUES TO OBTAIN CYCLE NO 1

CYCLE	1	2	3	4
2	.27012E+00	.61979E-01	.65933E-01	.16428E+01
3	-.19134E+01	.19515E+00	.41677E-01	-.13278E+01
4	-.21311E+00	.19218E+00	.41113E-02	-.45486E+00
5	.95452E-02	-.16147E-01	-.10528E-01	.12037E+00
6	-.28819E-02	-.21638E-02	-.16311E-02	.23568E-01
7	-.10173E-02	-.15792E-03	-.10384E-03	.13933E-02
TOTAL	.43522E-01	.45084E+00	.99458E-01	.54376E-02

SCALE FACTORS

CYCLE	1	2	3	4
1	.16690E+01	.38786E+02	.13946E+03	.93620E+01
2	.19420E+01	.29762E+02	.96496E+02	.68104E+01
3	.19419E+01	.31490E+02	.78972E+02	.88702E+01
4	.19924E+01	.32581E+02	.71937E+02	.95958E+01
5	.19933E+01	.32731E+02	.74699E+02	.92715E+01
6	.19941E+01	.32732E+02	.75146E+02	.92092E+01
7	.19942E+01	.32732E+02	.75173E+02	.92056E+01

RADIUS OF GEODESIC CURVATURE IN DIRECTION OF SEARCH

CYCLE	ABSOLUTE	RELATIVE TO /SQA/	TOTAL COSINE	/SQA/	/DELTA(PART)/	IN++K
1	54.779	2.365	.994423	.23761E+02	.25513E+01	0
2	1.616	.356	.869385	.58960E+01	.27493E+01	16
3	19.639	5.277	.756777	.37214E+01	.53282E+00	8
4	.338	.138	.669628	.24546E+01	.13349E+00	7
5	1.750	.715	.616755	.24488E+01	.22799E-01	6
6	.249	.101	.600541	.24484E+01	.16281E-02	6

PARTIAL COSINES

CYCLE	1	2	3	4
1	.53113	.08493	.95845	.99378
2	.35538	.38705	-.14338	-.02468
3	.52422	.62952	.20639	.38231
4	.63742	.00051	-.02034	-.00428
5	.81228	.01224	.01284	.01479
6	-.08036	-.00025	-.00801	-.00009
7	.96002	.00002	.00801	.00802
7	.2104E-04	.1690E-04	.9629E-05	.1600E-04

NAME		FINAL PARAMETER VALUES		STANDARD DEVIATION		T-VALUE
PORE SPACE	1	.4544352232E+02	1	.12835E+01	1	35.41
LOCATION A	2	.1760849947E+01	2	.3520IE+00	2	11.52
SCALE B	3	.3760576162E+00	3	.11866E+00	3	3.15
EXPONENT C	4	.3494437595E+01	4	.1420BE+01	4	2.45

MULT.CORR= .99857 SQA= .5994876018E+01 STAND.ERROR= .1894977IE+01 TOTAL COSINE= .2583768-04

LINEARIZED IN TANGENT PLANE

95 0/0 CONFIDENCE INTERVAL

	K	LOWER BOUND		MEAN VALUE		UPPER BOUND	K
PORE SPACE	1	42.9	-	45.4	-	48.0	1
LOCATION A	2	1.46	-	1.76	-	2.06	2
SCALE B	3	.14	-	.37	-	.61	3
EXPONENT C	4	.6	-	3.5	-	6.3	4

NUMBER OF DATA 9

COS CRITERIUM FULFILLED WITH 3 ZEROS FOR / NO YES

*** NPART= 4 **

TOTAL COSINE **

NPAR = 4 ** / IN CYCLE NO 7 OUT OF 10 CYCLES

STATISTICS

FROM DATE 01/17/75 TIME 12.46.27. CP-SEC.USED 2.665

TILL DATE 01/17/75 TIME 12.46.30. CP-SEC.USED 4.237

USED FOR THIS JOB CP-SEC. 1.572

TOTAL NO OF ENTRIES IN FNCTN DFDA D2FDA SOLVE MIN

62 62 7 7 6

DEFINED NO OF SUMMARIES. PRODUCED. CYCLES/SUMRY. PRODUCED IN LAST ONE. TOTAL NO OF CYCLES

1 1 10 7 7

REQUIRED CONDITION FULFILLED
PROCESS TERMINATED AT-STOP 88-
IN RLV FROM HOUR ENTRY SUMRY

END OF JOB

1.5 Instructions to apply main program NLV

```
*) TO DISTINGUISH BETWEEN EXISTING AND SUPPLEMENTARY STATEMENTS THE LATTER ARE
   NUMBERED IN THE 73RD COLUMN. DEFAULT VALUES THAT SHOULD BE CHANGED FOR A
   PROPER EXECUTION OF THE MODIFICATION ARE GIVEN IN THE HEADING  AND ARE MARKED
   WITH 0 IN THE 73RD COLUMN.
   PROGRAM TEXTS ARE GIVEN BETWEEN FULL *** AND --- LINES. A SERIES OF EXISTING
   AND UNALTERED CARDS IN THE MAIN PROGRAM NLV IS DENOTED BY ... IN THE MARGIN.
   IN MODIFICATION EXAMPLES THE FIRST AND THE LAST LINE ARE EXISTING UNALTERED
   STATEMENTS OF THE DEFAULT DECK.
   FOR INFORMATIVE USE A STOP STATEMENT CAN BE INSERTED AT THE DESIRED POINT OF
   THE MAIN PROGRAM NLV. USE OF LSUMRY=0 IS SUGGESTED FOR USE IN DIFFERENT RUNS
   OF MODIFICATIONS THAT ARE ENTERED AFTER THE 40 CONTINUE STATEMENT.

MODIFICATION 1. TEST OF THE FORMULAS FOR FIRST AND SECOND DERIVATIVES
1.1      CURVE FITTING WITH ROUNDED CALCULATED VALUES USED AS OBSERVATIONS
*******************************************************************************
      CALL READ (1,1,NOUTRD)
      LSUMRY=1  $  NCYCLES=5  $  CALL FNCTN(A,1,SQA,1,0,YCLCA,YOCA,0,0)  1
      DO 85 I=1,NDATA    $    ALOG10Y=ALOG10(ABS(YCLCA(I)))             2
      IF(ALOG10Y.LT..0) ALOG10Y=ALOG10Y-1.    $    IPOWER=2-IFIX(ALOG10Y) 3
   85 YOBS(I)=FLOAT(IFIX(YCLCA(I)*10.**IPOWER))/10.**IPOWER             4
      IREP = 0
...
      CALL MIN(DELTA,0,1.,0.,0,20,0,0)
      IF(NC.EQ.1)                                                       5
     *CALL MIN(DELTA,0,0,-.2*EPMIN(NOCLC),.1*EPMIN(NOCLC),22,0,0)       6
      CALL AISB
-------------------------------------------------------------------------------

MODIFICATION 2. PROPERTIES OF CONDITION FUNCTION AND STARTING POINT
2.1      DETERMINATION OF A NEW STARTING POINT BY UNIVARIATE SEARCH FOR THE FIRST
         NPART PERMUTED PARAMETERS
*******************************************************************************
      CALL READ (1,1,NOUTRD)
      CALL NEWZERO(10,6,6)                                              1
      CALL PUNCH(0)                                                     2
      STOP                                                              3
      IREP = 0
-------------------------------------------------------------------------------
   THE STOP STATEMENT IS OPTIONAL

2.2      CALCULATION OF THE BEST PARAMETER COMBINATION BY COMBINATORIAL SEARCH
      LSUMRY=0                                                          0
*******************************************************************************
      CALL HOWA
      CALL COMBIN           $             IF(NC.LT.3) GO TO 20          1
                 IF(NOCLC.GE.NCYCLS) GOTO 30   $   NOCLC=NOCLC+1
-------------------------------------------------------------------------------
   NUMBER OF CYCLES AUTOMATICALLY RESTRICTED TO 4

MODIFICATION 3. SELECTIVE USE OF PARAMETERS
3.1      AUTOMATICAL ORDERING OF PARAMETERS BY MEANS OF THEIR PARTIAL COSINES
   USE DEFAULT LSUMRY=1                                                 0
*******************************************************************************
      CALL DFDA(A,HA,YCLCA,YOCA,1,1,NOUTDF)
      CALL ORDER(A,HA,YCLCA,YOCA,1,1,NOUTOF)                            1
      CALL D2FDA(A,SQA,HA,YCLCA,YOCA,1,NOUTD2F)
-------------------------------------------------------------------------------

3.2  USE OF TWO BLOCKS WITH PERMUTED AND GROUPED PARAMETERS IN THE FIRST BLOCK
         MAXIMUM NUMBER OF BLOCKS THAT CAN BE USED IS 5
   USE LSUMRY=1 (DEFAULT) FOR INTERMEDIATE RESULTS.                     0
*******************************************************************************
C     BLOCK 1
      K=0                                                                    0
      K=K+1   $   IP(K)= 4                         $   JP(IP(K))=K
      K=K+1   $   IP(K)= 2                         $   JP(IP(K))=K
                            NPART = K                                        1
      K=K+1   $   IP(K)= 1                         $   JP(IP(K))=K
                            NPAR  = K                                        2
      K=K+1   $   IP(K)= 3                         $   JP(IP(K))=K
                            NPAR  = K                                        3
      CALL BLOCK(1,0,0,.0)                                              1
C     BLOCK 2
      K=0                                                               3    0
      K=K+1   $   IP(K)= 1                         $   JP(IP(K))=K      4
      K=K+1   $   IP(K)= 2                         $   JP(IP(K))=K      5
      K=K+1   $   IP(K)= 3                         $   JP(IP(K))=K      6
      K=K+1   $   IP(K)= 4                         $   JP(IP(K))=K      7
                            NPART = K                                   8    1
                            NPAR  = K                                   9    2
                            NPAR  = K                                   10   3
      CALL BLOCK(2,0,0,.0)                                              11
*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ****
...
   20           NC=NC+1
      CALL BLOCK(0,2,3,10.)                                             12
      CALL FNCTN (A,1,SQA,1,0,YCLCA,YOCA,1,NOUTFN)
-------------------------------------------------------------------------------
   ALSO CAN BE USED E.G.   CALL BLOCK(0,2,NCYCLES*END,COSCRIT)          12

MODIFICATION 4. APPLICATION OF ALGORITHMS
4.1      CURVE FITTING ACCORDING TO STEEPEST DESCENT
*******************************************************************************
                 IF(NOCLC.GE.NCYCLS) GOTO 30   $   NOCLC=NOCLC+1
      CALL MIN(ANORM,0,1, .0, .0,20,0,0)                                1
      CALL AISB
-------------------------------------------------------------------------------

MODIFICATION 5. SELECTED PATHS
5.1A     CONTROLLED APPROACH ALONG THE FIRST YOCA-VECTOR WITH TRACK(1,NPART,...)
         CONTROLLED APPROACH ALONG EACH  NEW YOCA-VECTOR WITH TRACK(2,NPART,...)
*******************************************************************************
                NC=0                          $   NCYCLS=NCYCLES
      CALL TRACK(1,NPART,.2,4,5,0,1)                                    1
   20           NC=NC+1
...
   30 CONTINUE
                IF( IREP .LT. END ) GO TO 10                            2
      CALL SUMRY,RETURNS(40)
                IF( IREP .LT. END ) GO TO 10
...
   40 CONTINUE
                IF( IREP .LT. END ) GO TO 10                            3
      STOP 55
-------------------------------------------------------------------------------

5.1B     CONTROLLED APPROACH WITH REPETITION OF THE PROCEDURE ON THE LAST INTERVA
         THEIR NUMBER BEING DEFINED BY THE VARIABLE LAG
*******************************************************************************
      IREP = 0
      IDO=1    $    IDONE=0                                             1
   10 IREP = IREP+1  $        I3=3            $   IF(LSUMRY.EQ.0) I3=7
...
                NC=0                          $   NCYCLS=NCYCLES
      MSTEP=10  $  CALL TRACK(1,NPART,.0,MSTEP, 5,0,1) $ IDONE=IDONE+1  2
   20           NC=NC+1
...
   30 CONTINUE
      LAG=2  $  IF(IDONE.GT.MSTEP-LAG.AND.IDO.EQ.1)IDO=NRINSUB(16)=0    3
                IF( IREP .LT. END ) GO TO 10                            4
      CALL SUMRY,RETURNS(40)
...
   40 CONTINUE
                IF( IREP .LT. END ) GO TO 10                            5
      STOP 55
-------------------------------------------------------------------------------

MODIFICATION 6. SIMULATION OF REVERSED TYPE OF CONVERGENCE
6.1   APPROACH WITH OBSERVATION VECTOR REFLECTED IN EACH NEW TANGENT PLANE
*******************************************************************************
                IF(NOCLC.GE.NCYCLS) GOTO 30   $   NOCLC=NOCLC+1
      CALL TRACK(3,NPART,.0,0,0,0,0)                                    1
      CALL MIN(DELTA,0,1,.0,.0,20,0,0)
-------------------------------------------------------------------------------

6.2 APPROACH WITH OBSERVATION VECTOR REFLECTED IN THE TERMINAL POINT
*******************************************************************************
   40 CONTINUE
      CALL TRACK(4,NPART,.0,0,0,0,0)                                    1
                IF( IREP .LT. END ) GO TO 10                            2
      STOP 55
-------------------------------------------------------------------------------

MODIFICATION 7. INTERMEDIATE INFORMATION ON THE FITTING PROCESS
7.1      LISTING OF THE (EPS,SQA)-RELATIONSHIP WITH EQUAL STEPS ABOUT THE MINIMUM
*******************************************************************************
      CALL MIN(DELTA,0,1,.0,.0,20,0,0)
      CALL MIN(DELTA,0,0,-.2*EPMIN(NOCLC),.1*EPMIN(NOCLC),22,0,0)       1
      CALL AISB
-------------------------------------------------------------------------------

7.2      CALCULATION OF CURVATURE ALONG EACH PARAMETRIC CURVE SEPARATELY
*******************************************************************************
      CALL MIN(DELTA,0,1,.0,.0,20,0,0)
      DO 80 K=1,NPART  $   DO 81 KL=1,NPAR                              1
   81 CORR(KL)=.0            $            CORR(K)=1.                    2
      CALL MIN( CORR,1,0,.0,.0,0,0,0)                                   3
   80 CONTINUE                                                          4
      STOP
      CALL AISB
-------------------------------------------------------------------------------

MODIFICATION 8. SPEEDING UP CONVERGENCE USING FIRST DERIVATIVES
8.1     CORRECTION FOR NON LINEARITY BY SCALE FACTOR WEIGHTS BASED ON DIFFERENCES
*******************************************************************************
                 IF(NOCLC.GE.NCYCLS) GOTO 30   $   NOCLC=NOCLC+1
      EPS=EPS1   $   CALL MIN(DELTA,0,1,.0,.0,20,0)    $   EPS1=EPS     1
      DO 55 I=1,NPAR                                                    2
   55 CORR(I)=CORR(I)*HA(I)/HB(I)                                       3
      EPS=EPS2   $   CALL MIN(CORR,0,1,.0,.0,20,0)     $   EPS2=EPS     4
      CALL AISB
-------------------------------------------------------------------------------

8.2A     CORRECTION FOR CURVATURE BY BACK PROJECTION
   TWO DIMENSIONAL SEARCH IN PARAMETER SPACE
*******************************************************************************
      CALL MIN(DELTA,0,1,.0,.0,20,0,0)
      CALL BACK(NPART,0,0,0,5)                                          1
      CALL AISB
-------------------------------------------------------------------------------
   TWO DIMENSIONAL SEARCH IN OBSERVATION SPACE IS PERFORMED BY
      CALL BACK(NPART,0,1,0,5)                                          1

8.2B     CORRECTION FOR CURVATURE BY BACK PROJECTION
   CIRCULAR SEARCH IN PARAMETER SPACE
*******************************************************************************
      CALL MIN(DELTA,0,1,.0,.0,20,0,0)
      CALL BACK(NPART,0,0,1,5)     $   EPSSAVE=EPS                      1
      IF(NRINSUB(15).EQ.5000) GO TO 5000                                2
      CALL MIN(CORR,0,1,.0,.0,20,0,0)                                   3
      CALL MIN(CORR,0,0,.0,3.1415926535/16.,32,0,0)                     OPTIONAL
      CALL B1 (0,0,0,0,0)                                               4
      CALL MIN(CORR,0,1,.0,.0,20,0,0)                                   5
      CALL B2 (0,0,0,0,0)           $   EPS=EPSSAVE                     6
 5000 CONTINUE                                                          7
      CALL AISB
-------------------------------------------------------------------------------
   THE WRITE OPTION GIVES THE (EPS,SQA)-RELATIONSHIP FROM 0 TO 2*PI
      CALL BACK(NPART,1,0,1,5)                                          1
   CIRCULAR SEARCH IN OBSERVATION SPACE IS PERFORMED BY
      CALL BACK(NPART,0,1,1,5)                                          1

8.3     SIMULATED APPROACH BY POLYNOMIAL, HYPERBOLIC OR EXPONENTIAL EXTRAPOLATIN
      END=2                                                             0
*******************************************************************************
      CALL SUMRY,RETURNS(40)
      CALL LIHYPEX(1,3,4,5,0,1)                                         1
      CALL MIN(CORR,0,1,.0,.0,20,1,0)                                   2
      CALL AISB                                                         3
                IF( IREP .LT. END ) GO TO 10
-------------------------------------------------------------------------------

MODIFICATION 9. SPEEDING UP CONVERGENCE USING SECOND DERIVATIVES
9.1      USE OF APPROXIMATED STEP FACTOR BY SECOND DERIVATIVES
         REDUCTION OF NUMBER OF ITERATIONS IN SUBROUTINE MIN
*******************************************************************************
                IF(NOCLC.GE.NCYCLS) GOTO 30   $   NOCLC=NOCLC+1
      NO=20    $    IF(EPSEST.GT..5*EPS) NO=4                           1
                    IF(EPSEST.GT..5*EPS) EPS=EPSEST                     2
      CALL MIN(DELTA,0,1,.0,.0,NO,0,0)                                  3
      CALL AISB
-------------------------------------------------------------------------------

9.2  CORRECTION FOR NON LINEARITY BY SCALE FACTOR WEIGHTS BASED ON DIFFERENTIALS
*******************************************************************************
                 IF(NOCLC.GE.NCYCLS) GOTO 30   $   NOCLC=NOCLC+1
      EPSHA=EPS   $   IF(EPSEST.GT..5*EPS) EPSHA=EPS=EPSEST             1
      DO 60 K=1,NPAR                                                    2
   60 DELTA(K)=DELTA(K)/(1.+EPSHA*FFAANH(K)**2)                         3
      CALL MIN(DELTA,0,1,.0,.0,20,0,0)                                  4
      CALL AISB
-------------------------------------------------------------------------------
   BEFORE ENTERING THE DO LOOP EPSHA=1. COULD BE USED

MODIFICATION 10. INFORMATION ON RESULTS USING FIRST DERIVATIVES
10.1A NUMERICAL INVESTIGATION OF CURVATURE ALONG FITTING ORDER TRACK(5,NPART,..)
      NUMERICAL INVESTIGATION OF CURVATURE  FOR   EQUAL  STEPS TRACK(6,NPART,..)
*******************************************************************************
   40 CONTINUE
      CALL TRACK(5,NPART,.0,.0,0,1,0)                                   1
      CALL TRACK(6,NPART,.0,.0,0,1,0)                                   1
      STOP 55
-------------------------------------------------------------------------------

10.1B    INVESTIGATION OF THE FITTING SURFACE IN THE DIRECTION OF EACH PARAMETER
         SEPARATELY IN COMBINATION WITH MODIFICATION 10.1A
*******************************************************************************
   40 CONTINUE
      CALL TRACK(5,NPART,.0,.0,0,1,0)                                   1
      CALL TRACK(6,NPART,.0,.0,0,1,0)                                   2
      DO 55 K=1,NPART    $    DO 65 L=1,NPAR                            3
   65 A(L)=XA(NOCLC,L)                                                  4
      A(IP(K))=XA(1,IP(K))                                              5
   55 CALL TRACK(6,NPART,.0,0,0,0,0)                                    6
      STOP 55
-------------------------------------------------------------------------------
   USE CAN BE MADE OF PREDEFINED PARAMETER VALUES. IN THIS EXAMPLE THE INITIAL
   VALUE IS RECALLED BY THE 5TH INSERTED CARD. A USEFUL ALTERNATIVE IS
     A(IP(K))=A(IP(K))-2.*STDEV(K)                                      5

   FINAL REMARKS.
   ALTHOUGH EACH OF THESE MODIFICATIONS HAS BEEN TESTED CAREFULLY TO PERFORM
   ITS SPECIFIC ALGORITHM PROPERLY ARBITRARY COMBINATIONS OF MODIFICATIONS MIGHT
   PRODUCE UNPREDICTABLE RESULTS.
```

2 Examples

2.1 Specimen nonlinear condition function

Condition function $y = x^\theta$
Vector of observed variables $x = (x_1,x_2) = (y,x)$
Vector of parameters $\theta = (\theta)$
Observation matrix $X = \begin{bmatrix} 0.5 & 1 \\ 10.0 & e \end{bmatrix}$
Position vector $f = (1,e^\theta)^T$
First derivative of condition function $f_\theta = x^\theta \ln x$
Jacobian $J = \begin{bmatrix} 0 \\ e^\theta \end{bmatrix}$, scale factor $h = (e^\theta)$
Matrix of normal equations $J^TJ = M = (e^{2\theta}) = (7.38906)$
Inverse matrix of normal equations $M^{-1} = (e^{-2\theta}) = (0.13534)$
Initial parameter vector $\theta^{(0)} = (1)$
Difference vector $x - f(\theta^{(0)}) = f_0 = \begin{bmatrix} 0.5 - 1 \\ 10.0 - 2.718 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 7.282 \end{bmatrix}$
Sum of squares at initial point $f_0^Tf_0 = S = 53.273$
Normal vector $J^Tf_0 = N = (19.794)$ and $Jd = 7.282$
Differential correction $M^{-1}N = d = (2.679)$
Updated value for θ becomes $\theta^{(1)} = (1 + 2.679) = (3.679)$
Exact solution $\theta^* = (2.30258)$

Used to demonstrate application of arc length in Section 2.7.3. See also Figure 7.

2.2 Nonlinear condition function, fast convergence

A function analogous to the one given by Fink & Jackson (1973) to describe sigmoidal adsorption isotherms, applied to observations on the soil moisture tension (its common logarithm being denoted by x) and the moisture content (y) of soil samples appeared to be a good condition function to demonstrate properties of optimization processes. Examples are given for two series of observations carried out by the laboratory of the Institute for Land and Water Management Research, Wageningen. The first series (Table 18) gives an example of an optimization process with fast convergence, the second series, see Appendix 2.3, gives an example of an optimization process with slow convergence.

Table 18. Data to demonstrate fast convergence.

i	1	2	3	4	5	6	7	8	9
x	0.4	1.0	1.5	2.0	2.3	2.7	3.4	4.2	6.0
y	45.3	43.4	41.0	33.3	27.6	23.2	11.5	7.4	2.4

The condition function reads

$$y = D\{e^{(x-A)/B} + 1\}^{-1/C}$$

where the asymptotes are given by $y = 0$ and $y = D$.
Vector of observable variables $x = (y,x)$
Vector of parameters $\theta = (D,A,B,C)^T$
Initial parameter $\theta^{(0)} = (34.8, 1.31, 0.2746, 3.489)^T$
Initial parameter values obtained as an average for several samples. After the data are read the parameter θ_1 is redefined by $\theta_1 = y^{[1]} + 0.1$
Sum of squares at initial point $S(\theta^{(0)}) = 564.61$
Sum of squares at final point $S(\theta^{(f)}) = 6.00$
Final parameter $\theta^{(f)} = (45.44, 1.761, 0.3741, 3.494)^T$

Results obtained in this example are used in Sections 9.3.4, 9.4.4, 11.4 and Appendices 1.4.1 and 1.4.2.

2.3 Nonlinear condition function, slow convergence

Condition function and initial parameter as given in Appendix 2.2. Data as given in Table 19.

Sum of squares at initial point $S(\theta^{(0)}) = 976.40$
Sum of squares at final point $S(\theta^{(f)}) = 1.83$
Final parameter $\theta^{(f)} = (38.31, 2.1277, 0.5474, 3.047)^T$

Results obtained in this example are used in Sections 5.5, 6.5, 7.4.3, 9.4.4, and 10.6.

Table 19. Data to demonstrate slow convergence.

i	1	2	3	4	5	6	7	8	9
x	0.4	1.0	1.5	2.0	2.3	2.7	3.4	4.2	6.0
y	38.3	36.1	34.8	32.3	29.0	24.1	17.2	11.4	3.5

List of symbolic FORTRAN names

Main symbols used throughout the subprograms are given. The dimension of arrays is obtained by p = total number of parameters, $q = p + 1$, $c = pq/2$, m = total number of observed variables, v = total number of observations, t = predefined number of fitting cycles ($t = 15$), $t_1 = t + 1$.

Name	Dimension	Symbol	Interpretation
A	$p \times 1$	$\theta^{(0)}$	Initial parameter in each cycle
ALB	$p \times 1$		Lower bound of parameter values
ANORM	$q \times 1$	$N(\theta^{(0)})$	Normal at initial point A, extended with sum of squares SQA
AUB	$p \times 1$		Upper bound of parameter values
B	$p \times 1$	$\theta^{(1)}$	New parameter in direction of search
BNORM	$q \times 1$	$N(\theta^{(1)})$	Normal at point B, extended with sum of squares SQB
CORR	$p \times 1$	s	Arbitrary direction of search
CURVGEO	$p \times 1$	κ_g	Geodesic curvature
CURVTOT	$p \times 1$	κ	Total curvature
DELTA	$p \times 1$	d	Differential corrections
END			Number of repetitions of a predefined number of cycles
EPMIN	$t \times 1$	$\lambda^{*(n)}$	Optimal step factor in nth cycle
EPS		λ	Step factor
EPSNEW	$t_1 \times 1$	$\lambda^{(0)}$	Predicted step factor
EQS		λ_r	Reduction factor to step factor
FA	$v \times q$	J	Jacobian matrix of first derivatives extended with difference vector
FAA	$v \times c$	f_{kl}	Vectors of second derivatives
FAFAA	$p \times p$	M_{12}	Scalar products of vectors of first and second derivatives
FA2	$q \times 1$	(f_k, f_k)	Square of length of first derivatives extended with sum of squares (f_0, f_0)
FSS	$v \times 1$	f_{ss}	Curvature vector
F0FAA	$p \times p$	N_{02}	Scalar products of difference vector and vectors of second derivatives

Name	Dimension	Symbol	Interpretation
HA	$q \times 1$	$h(\theta^{(0)})$	Scale factors at initial point A, extended with square root of SQA
HB	$q \times 1$	$h(\theta^{(1)})$	Scale factors at point B, extended with square root of SQB
IB	$p \times 1$		Index to save bounded parameters
IP	$q \times 1$		Index to save parameter permutation
IPX	$t \times p$		Index to save parameter permutation for summary
IREP			Index to save number of repetitions of predefined number of cycles actually being performed
JP	$q \times 1$		Index to save original parameter ordening
M	$q \times q$	M	Matrix of normal equations (extended)
MAXDAT		v	Total number of data
MAXVEC		m	Total number of observed variables
MPAR		p	Total number of parameters
MT		$p+1$	MPAR+1
NB			Total number of bounded parameters
NC		n	Cycle number being performed
NCYCLES			Number of cycles to be performed
NCYCLS			Number of cycles to be performed
NDATA		v	Total number of data to be performed
NOCLC		n	Cycle number being performed
NPAR			Number of to be fitted and informative parameters
NPART		p	Number of to be fitted parameters
NPARTX	$t \times 1$	p	Number of to be fitted parameters saved for summary
NT		$p+1$	NPAR+1
SQA		$S(\theta^{(0)})$	Sum of squares at initial point A
SQB		$S(\theta^{(1)})$	Sum of squares at point B
SQMIN		$S^{*(n)}$	Subminimum of sum of squares in direction of search
STDEV	$p \times 1$		Linearized standard deviation of parameter estimates in tangent plane
X	$v \times m$	X	Observation matrix
XA	$t \times p$	θ	Initial and optimal parameter values saved for summary
XCOSMT	$t \times 1$	$\cos \alpha$	Multiple cosine saved for summary
XCOSN	$t \times p$	c	Partial cosines saved for summary
XCOSQ	$t \times 1$	$\cos \beta$	Total cosine saved for summary

Name	Dimension	Symbol	Interpretation
XCRLMT	$t \times 1$		Multiple correlation saved for summary
XDELT	$t \times 1$	$\| d \|$	Length of vector of differential corrections for NPART parameters saved for summary
XDELTA	$t \times 1$		Length of vector of differential corrections for NPAR parameters saved for summary
XGEO	$t \times 1$	$\varkappa_g$	Geodesic curvature in direction of search saved for summary
XHA	$t \times p$	h	Scale factors saved for summary
XM	$q \times q$	M, M^{-1}	Matrix to save matrix of normal equations and its inverse
XNORM	$t \times 1$		Length of normal for NPAR parameters saved for summary
XNRM	$t \times 1$	$\| N \|$	Length of normal for NPART parameters saved for summary
XSQF	$t \times 1$	$({}^A f_0, {}^A f_0)$	Sum of squares to tangent plane saved for summary
YCLC	$v \times 1$	y	Calculated function values
YCLCA	$v \times 1$	$f(\theta^{(0)})$	Condition function evaluated at initial point A
YCLCB	$v \times 1$	$f(\theta^{(1)})$	Condition function evaluated at point B
YOBS	$v \times 1$	x	Observed function values
YOCA	$v \times 1$	$f_0(\theta^{(0)})$	Difference vector at initial point A
YOCB	$v \times 1$	$f_0(\theta^{(1)})$	Difference vector at point B

List of symbols

Symbols defined or introduced in chapters or sections mentioned between parentheses are used in a restricted number of sections. The other symbols are used throughout the book. Symbols falling outside the main line of argument are defined in the text only. Dimension symbols are also explained in this list.

Symbol	Dimensions	Chapter Section	Interpretation
A		2.4	Tangent plane
A		2.4	Left superscript denoting quantities with respect to the tangent plane as ${}^A\theta$, ${}^Af_0$ and ${}^AS$
a		(10)	Constant of parameter function
B	$p \times p$	3.4	Metric matrix whose inverse acts on the vector of steepest descent $-g$
b	$p \times 1$	2.9	Auxiliary vector in tangent space
b	$p \times 1$	6.2	Vector of back projection
b		(10)	Constant of parameter function
C	$p \times p$	2.2	Cosine matrix whose elements, $\cos(f_k, f_l)$, are the cosines of the angle between the direction vectors
C		2.9	Contour curve for constant value of the response
c	$p \times 1$	2.2	Cosine vector whose components, $\cos(f_0, f_k)$, are the partial cosines
c_j		(3.1)	Constraint to jth variable
c_1, c_2		(4)	Constant function values
c		(6.3)	Algorithm parameter equal to $2 \cos \phi$ in back projection method
c		(10)	Constant of parameter function
d	$p \times 1$	(2.4)	Differential correction vector; the solution of the normal equations
d		(10)	Constant of parameter function
ds		2.7	Arc element
E^p		1.3	p-dimensional Euclidean parameter space
E^p		1.3	p-dimensional Euclidean tangent space

Symbol	Dimensions	Chapter Section	Interpretation
E^{p+1}		2.2	$(p+1)$-dimensional Euclidean response space
E^{v}		1.3	v-dimensional Euclidean observation space
e		(10)	Constant of parameter function
F	$v \times 1$	2.2	Vector of implicit condition functions F
F		1.2	Implicit condition function
${}^{1}F, {}^{2}F$		(4)	Left superscript denoting implicit condition functions of different form
f	$v \times 1$	2.1	Vector of explicit condition functions f
f_k	$v \times 1$	2.1	Direction vector, tangent to kth parametric curve on the fitting surface
f_s	$v \times 1$	2.8	Direction vector with arc length taken as parameter
f_{ss}	$v \times 1$	2.8	Curvature vector
f_0	$v \times 1$	2.1	Difference vector in the observation space, equal to $x - y$, for differences between observed and calculated function values
f		2.1	Explicit condition function
f_k		2.1	Explicit condition function differentiated with respect to the kth parameter
${}^{1}f, {}^{2}f, {}^{j}f$		(4)	Left superscript denoting explicit condition function of different form
G	$p \times p$	2.5	Hessian matrix whose elements, $\partial^2 S/\partial\theta_k \partial\theta_l$, are second derivatives of the objective function
G		(4)	Implicit condition function
g	$p \times 1$	2.2	Gradient vector
$-g$	$p \times 1$	2.2	Vector of steepest descent
g		2.8	Subscript denoting quantities related to geodesic curvature, e.g. κ_g, $(f_s)_g$ and $(f_{ss})_g$
g		(4)	Explicit condition function
H	$p \times p$	3.4	Matrix to be updated in subsequent cycles; approximation to G^{-1}
h	$p \times 1$	2.2	Vector of scale factors h_k
h_k		2.2	Scale factor; length of direction vector f_k
I		2.1	Identity matrix; dimensions being defined by operation rules of matrix algebra
i		1.2	General superscript for observations, e.g. $x^{(i)}$
i		(11)	General superscript for steps in controlled approach, e.g. $x^{(i)}$

Symbol	Dimension	Chapter Section	Interpretation
i		(10.1)	General subscript for equidistant steps, e.g. λ_i
$\boldsymbol{J}$	$v \times p$	2.2	Jacobian matrix, $\boldsymbol{J}_k^{[i]}$, consisting of direction vectors $\boldsymbol{f}_k$
$\boldsymbol{J}_\theta^f$	$v \times p$	2.2	Jacobian matrix, $\partial f^{[i]}/\partial\theta_k$; abbreviated to $\boldsymbol{J}$
$\boldsymbol{J}_\lambda^\theta$	$p \times q$	2.5	Jacobian matrix $\partial\theta_k/\partial\lambda_j$
$\boldsymbol{J}_\theta^F$	$v \times p$	4.3	Jacobian matrix $\partial F^{[i]}/\partial\theta_k$ for implicit condition function F
$\boldsymbol{J}_2$	$v \times \binom{p+1}{2}$	2.5	Matrix of second derivaties $f_{kl}^{[i]}$ where $k \leqq l$
$\boldsymbol{Jb}$	$v \times 1$	(6)	Vector in tangent plane obtained by back projection; linear combination of direction vectors
$\boldsymbol{Jd}$	$v \times 1$	2.4	Total tangent; linear combination of direction vectors according to the solution of the normal equations
$\boldsymbol{j}$	$v \times 1$	(4.9)	Choice vector to save information on the sequence of executed statements for alternative condition functions
j		1.2	General subscript for variables
j		(1.2)	Subscript assigned to quantities that refer to a condition function solved for the jth variable, e.g. $x_j, f_j, y_j, f_{0j}, S_j$
j		(4)	Ibidem
j		2.5	General subscript for algorithm parameters
j		(4)	Left superscript for general index of condition functions of different form, e.g. ${}^jf$
j		(10)	General superscript for fitting cycles, e.g. $\theta_k^{(j)}$
$\boldsymbol{K}$	$v \times p$	2.8	Matrix derived from the Jacobian for the determination of the curvature in an arbitrary direction of search
k		1.2	General first choice subscript for parameters, e.g. θ_k
k		2.1	General first choice subscript denoting differentiation with respect to the kth parameter, e.g. f_k
$\boldsymbol{l}$	$v \times 1$	2.3	Position vector of the tangent plane
l		2.1	General second choice subscript for parameters, e.g. θ_l
l		2.1	General second choice subscript denoting differentiation with respect to the lth parameter, e.g. f_l

Symbol	Dimension	Chapter Section	Interpretation
M	$p \times p$	2.4	Matrix of normal equations with elements (f_k, f_l); the square of the Jacobian J
M_{12}	$p \times p$	2.5	Matrix with elements (f_k, f_{kl})
M_{22}	$\binom{p+1}{2} \times \binom{p+1}{2}$	2.5	Matrix which is the square of the matrix J_2
m		1.2	Total number of real variables of the condition function
N	$p \times 1$	2.4	Normal vector with components (f_0, f_k); the right-hand side of the normal equations
N	$p \times 1$	(6.2)	Normal vector redefined for back projection method
N_{02}	$p \times p$	2.5	Matrix with elements (f_0, f_{kl})
N		(11)	Number of steps in controlled approach
n		1.4	General superscript for fitting cycles e.g. $\theta^{(n)}$
$o(\lambda^2)$		(5.3)	Remainder of Taylor expansion of scale factors representing terms of second degree and higher
$o(\lambda^3)$		(2.6)	Remainder of Taylor expansion of the objective function representing terms of third degree and higher
p		1.2	Total number of real parameters of the condition function
q		2.5	Total number of algorithm parameters
R		(9.3)	Reduction factor used in finding the subminimum
r		2.8	Subscript denoting quantities related to remainder curvature, e.g. κ_r, $(f_s)_r$ and $(f_{ss})_r$
r, r_1, r_2		(4)	System parameter to control sequential use of sequential condition functions
r		(9)	Subscript for reduction factor to step factor, viz. λ_r
S		1.2	Objective function (f_0, f_0); sum of squares; response
s	$p \times 1$	1.4	Arbitrary vector of direction of search
s		2.7	Arc length
T		(9.3)	Measure of slope to $(\lambda, S(\lambda))$-curve
t		1.4	Superscript denoting fitting cycle in which stopping criteria are fulfilled, e.g, $\theta^{(t)}$
t		(9.3)	Reduced measure of slope ρT
t_1, t_2		(10.4)	Auxiliary variable in exponential extrapolation

Symbol	Dimension	Chapter Section	Interpretation
U		(2.2)	Arbitrary matrix
u		(2.1)	Arbitrary vector; (also in Sections 2.9 and 3.5)
u		(2.1)	Arbitrary variable
u		(4)	Arbitrary function of the parameters
v		(2.1)	Arbitrary vector; (also in Section 3.5)
v		(2.1)	Arbitrary variable
v		(4)	Arbitrary function of the parameters
w	$p \times 1$	3.5	Vector of weights; weights to be applied to s; defined in Chapter 5
w		(4)	Arbitrary function of the parameters
w		(4.8)	Auxiliary variable for sequential condition functions
X	$v \times m$	1.2	Observation matrix with elements $x_j^{[i]}$
x	$1 \times m$	1.2	Vector of real variables x_j
x	$v \times 1$	2.1	Observation vector with components $x^{[i]}$
$x^{(i)}$	$v \times 1$	(11)	ith intermediate observation vector used in controlled approach
x_j	$v \times 1$	(2.1)	Observation vector; observed values for y_j
x		1.2	Observable real variable of the condition function
y	$v \times 1$	2.2	Vector of calculated values $y^{[i]}$; position vector of the fitting surface
y_j	$v \times 1$	(2.1)	Vector of calculated values $y_j^{[i]}$
y_j		(2.1)	Variable for which the implicit condition function F is solved
z	$p \times 1$	(6.4)	Basis vector used in back projection method
z		(4.5)	Arbitrary function of the parameters
z		(4.6)	Auxiliary variable for sequential condition functions; (also in Section 4.8)
α		2.4	Angle between observation vector x and position vector y
$\cos \alpha$		2.4	Multiple cosine; cos (x,y)
α		(3.5)	Algorithm parameter; (also in Section 6.3.1)
β		2.4	Angle between f_0 and Jd
$\cos \beta$		2.4	Total cosine; cos (f_0,Jd)
β		(3.5)	Algorithm parameter; (also in Section 6.3.1)
β		(11)	Reduction factor of difference vector f_0 used in controlled approach

Symbol	Dimension	Chapter Section	Interpretation
Δ	$p \times 1$	2.3	Difference vector in the parameter space
δ	$p \times 1$	(7.4)	Stopping criterion vector for partial cosines
δ	$v \times 1$	2.3	Remainder vector of Taylor expansion of position vector with components $\delta^{[i]}$
δ		(7.4)	Stopping criterion
θ	$p \times 1$	1.2	Parameter vector of real parameters θ_k
θ		1.2	Real parameter of condition function
κ		2.8	Total curvature
κ_g		2.8	Geodesic curvature
κ_r		2.8	Remainder curvature
λ	$q \times 1$	2.5	Vector of algorithm parameters λ_j
λ		1.4	Algorithm parameter: step factor in direction of search
λ_i		(12.1)	Equidistant points in direction of search
λ_r		9.4	Reduction factor to step factor
ν		1.2	Total number of observations
ρ		9.3	Reduction factor applied to measure of slope T
ϕ		6.3	Angle in tangent subplane between Jd and Jb; algorithm parameter in back projection method
ϕ		(9.3)	Angle between g and s
ψ		6.4	Algorithm parameter in back projection method

Special notation

Symbol	Dimension	Chapter Section	Interpretation
D		2.2	Superscript, adds the kth component of the superscripted vector to the (k,k)th element of a zero matrix
d		2.2	Superscript, adds the inverse of the kth component of the superscripted vector to the (k,k)th element of a zero matrix
d			Differential operator
e			Base of system of natural logarithms
ln			Natural logarithm
$\min(a,b)$			The minimum of the numbers a and b
T		1.2	Superscript denoting transposition of a vector or a matrix
∂_k		2.1	General component of the operator ∇
$\in$			Element of
∇	$p \times 1$	2.1	Vector differential operator for differentiation with respect to condition function parameters
∇'	$q \times 1$	2.5	Vector differential operator for differentiation with respect to algorithm parameters
[ ]			Superscript; brackets denoting number of observation, e.g. $x^{[i]}$
( )			Superscript; parentheses denoting number of fitting cycle, e.g. $\theta^{(n)}$
$:=$			Is defined by
$=:$			Which defines
*			Superscript denoting quantities related to the minimum response
*(n)			Superscript denoting quantities related to the minimum response in the nth cycle or to the nth subminimum
$\| \ \|$			Euclidean norm
$\|$			Under the condition. Given

Symbol	Dimension	Chapter Section	Interpretation
$'$			Prime, denoting the derivative with respect to the innermost argument
$'$			Prime, used in superscripts defined in the text
$-$		(9.3)	Subscript assigned to quantities left to the (sub)minimum
$+$		(9.3)	Subscript assigned to quantities right to the (sub)minimum
$\mathfrak{Z}$		2.1	Matrix (vector) whose elements (components) are equal to 3; dimensions being defined by operation rules of matrix algebra
$\mathfrak{1}_k$		2.1	Unit vector in direction of kth coordinate axis of an orthogonal reference system; dimensions being defined by operation rules of matrix algebra
$k = 1(1)p$		1.3	The general subscript k ranges from 1 through p, with increments ( ) equal to 1