United Kingdom Atomic Energy Authority

RESEARCH GROUP

Report

A FORTRAN SUBROUTINE FOR SOLVING SYSTEMS OF NON-LINEAR ALGEBRAIC EQUATIONS

M. J. D. POWELL

Theoretical Physics Division,

Atomic Energy Research Establishment,

Harwell, Berkshire

1968

Available from H. M. Stationery Office
EIGHT SHILLINGS NET

1. Introduction

The method of this report will be justified in a companion paper (Powell, 1969), so now we state briefly the reasons for adding yet another algorithm to the various methods for solving numerically the system of non-linear equations

$$f_k(x_1,x_2,\dots,x_n) = 0, \quad k=1,2,\dots,n. \tag{1}$$

The new algorithm occurred from a consideration of existing algorithms, and primarily it resulted from asking when certain useful algorithms would fail. In particular the class of methods based on the Newton iteration (see Ostrowski, 1966, for instance) can fail if the Jacobian matrix becomes singular, and this is the main difficulty that we try to overcome.

Throughout this report we will use the vector notation $\underset{\sim}{x}$ to indicate the vector of unknowns $(x_1,x_2,\dots,x_n)$.

In the classical Newton iteration we have a guess $\underset{\sim}{x}$ of the solution of the system (1), and we calculate the elements of the Jacobian matrix

$$J_{kj} = \frac{\partial}{\partial x_j} f_k(\underset{\sim}{x}) \tag{2}$$

at the guess. Next we obtain a correction vector $\underset{\sim}{\delta}$ by solving the system of linear equations

$$\sum_{j=1}^{n} J_{kj}\, \delta_j = - f_k(\underset{\sim}{x}), \quad k=1,2,\dots,n, \tag{3}$$

and we complete the iteration by replacing the vector $\underset{\sim}{x}$ by the vector $(\underset{\sim}{x}+\underset{\sim}{\delta})$. The success of the Newton iteration is due to the fact that the correction is calculated so that, if the Jacobian is non-singular at the solution and if the functions $f_k(\underset{\sim}{x})$ are twice differentiable, then near the solution

$$f_k(\underset{\sim}{x}+\underset{\sim}{\delta}) = O\,(||\underset{\sim}{\delta}||^2), \quad k=1,2,\dots,n. \tag{4}$$

Thus the Newton iteration converges rapidly if the guess is a sufficiently good one. However it is well known that divergence often results if a close estimate of the solution is not available.

In the new algorithm we retain the fast convergence of the Newton method, but we modify the iteration so that it is progressive even if the guess $\underset{\sim}{x}$ is far from the solution. Other useful attempts to meet this objective have been made by Barnes (1965), Broyden (1965), Fletcher (1968), Haselgrove (1961), Levenberg (1944), Marquardt (1963), Powell (1965) and other authors, but we claim that the present one has certain advantages.

The main observation of Haselgrove's (1961) paper (which is also noted by many other authors) is that it is often worthwhile to use the correction $\underset{\sim}{\delta}$, calculated from equation (3), as a search direction in the space of the variables. Specifically Haselgrove's iteration replaces $\underset{\sim}{x}$ by $(\underset{\sim}{x}+\lambda\underset{\sim}{\delta})$, where the value of the parameter λ is calculated by a search process, which tries to make the estimate $(\underset{\sim}{x}+\lambda\underset{\sim}{\delta})$ better than the estimate $\underset{\sim}{x}$, the criterion for success being the inequality

$$F(\underset{\sim}{x}+\lambda\underset{\sim}{\delta}) < F(\underset{\sim}{x}) \,, \tag{5}$$

where $F(\underset{\sim}{x})$ is the sum of squares of residuals

$$F(\underset{\sim}{x}) = \sum_{k=1}^{n} \left[f_k(x) \right]^2 . \tag{6}$$

A successful value of λ can be obtained if (i) the functions are differentiable, (ii) the Jacobian is non-singular, and (iii) a solution has not been reached, because, if the left hand side of the inequality (5) is regarded as a function of λ, we find, using equation (3), the result

$$\begin{aligned}\left[\frac{\partial}{\partial\lambda} F(\underset{\sim}{x}+\lambda\underset{\sim}{\delta})\right]_{\lambda=0} &= 2 \sum_{j=1}^{n} \delta_j \left\{ \sum_{k=1}^{n} J_{kj}\, f_k(\underset{\sim}{x}) \right\} \\ &= -2 \sum_{k=1}^{n} \left[f_k(\underset{\sim}{x}) \right]^2 \\ &< 0 \,. \end{aligned} \tag{7}$$

Therefore by calculating the value of λ at each iteration, the sum of squares $F(\underset{\sim}{x})$ can usually be made to decrease monotonically, and so we hope to reduce the left hand sides of the equations to values that are very close to zero. However this hope cannot always be realised.

A different method (which we prefer) for modifying the Newton iteration, so that it converges from a poor initial estimate of the required vector $\underset{\sim}{x}$, is suggested by Levenberg (1944) and Marquardt (1963). It is derived by introducing a parameter λ^* into the "normal least squares" formulation of the equations (3):

$$\sum_{j=1}^{n} \left\{ \sum_{k=1}^{n} J_{ki} J_{kj} \right\} \delta_j = - \sum_{k=1}^{n} J_{ki} f_k(\underset{\sim}{x}), \quad i=1,2,\ldots,n. \tag{8}$$

Specifically they obtain a correction vector $\underset{\sim}{\delta}^*$ by solving the set of linear equations

$$\sum_{j=1}^{n} \left\{ \lambda^* I_{ij} + \sum_{k=1}^{n} J_{ki} J_{kj} \right\} \delta_j^* = - \sum_{k=1}^{n} J_{ki} f_k(\underset{\sim}{x}), \quad i=1,2,\ldots,n, \tag{9}$$

where I is the unit matrix. We note that the systems (8) and (9) are identical in the case $\lambda^*=0$. However when, as is usual, λ^* is positive, then $\underset{\sim}{\delta}^*$ is different from $\underset{\sim}{\delta}$, and when λ^* is very large, $\underset{\sim}{\delta}^*$ is approximately equal to the gradient vector of the sum of squares (6) multiplied by the small negative number $-1/2\lambda^*$. The Levenberg/Marquardt iteration changes an estimate $\underset{\sim}{x}$ to the estimate $(\underset{\sim}{x}+\underset{\sim}{\delta}^*)$, the length of the correction being regulated by the value of λ^*. It can be shown that the inequality

$$F(\underset{\sim}{x}+\underset{\sim}{\delta}^*) < F(\underset{\sim}{x}) \tag{10}$$

is obtained if λ^* is sufficiently large, provided that the functions $f_k(\underset{\sim}{x})$ have continuous first derivatives, and that the components of the gradient of $F(\underset{\sim}{x})$ are not all equal to zero at the initial estimate $\underset{\sim}{x}$ of the iteration.

We have described the two ideas for trying to obtain convergence, because we prefer the method due to Levenberg and Marquardt, although most practical algorithms are based on Haselgrove's approach. Our reason is that Haselgrove's idea fails more often, for equation (3) is not valid whenever the Jacobian matrix becomes singular, while the other methods breaks down only at a stationary point of $F(\underset{\sim}{x})$.

However a singular Jacobian need not spoil Haselgrove's technique, because equation (3) is used just to define the direction of the correction to be applied to $\underset{\sim}{x}$, the length of the correction being calculated to obtain the inequality (5). When J tends to singularity, the direction of $\underset{\sim}{\delta}$, defined by equation (3), usually tends to that of an eigenvector of J whose eigenvalue is zero, so it is often possible to identify it. But along such a direction the initial gradient of $F(\underset{\sim}{x})$ is zero, so there need not be a value of λ that satsifies the condition (5).

That singular Jacobians are not uncommon is shown by the equations

$$\left.\begin{aligned} f_1(\underset{\sim}{x}) &= x_1 - 1 = 0 \\ f_2(\underset{\sim}{x}) &= x_1 x_2 - 1 = 0 \end{aligned}\right\}. \qquad (11)$$

In this case the determinant of J is equal to x_1, which is positive at the solution (1,1). Therefore for all initial estimates (x_1,x_2) for which $x_1 < 0$, the determinant must be zero at some point of a successful path to the solution, so we may find a singular Jacobian.

The companion paper (Powell, 1969) includes a two-equation example showing failure of Haselgrove's idea, for (x_1,x_2)converges to a point at which the gradient of $F(\underset{\sim}{x})$ is not equal to zero.

Because of the above discussion, the algorithm of this paper tends to take steps along the steepest descent direction of $F(\underset{\sim}{x})$, if it seems that the classical Newton iteration diverges. The actual method used to define the correction to $\underset{\sim}{x}$ at each iteration is different from the Levenberg/Marquardt technique, because our method requires less computation when derivatives are approximated numerically.

The steepest descent qualities of the iteration yield some reassuring theorems on convergence (Powell, 1969), but we are content to reach a stationary point of $F(\underset{\sim}{x})$, even if it is not a solution to the equations. We take this view because sometimes the algorithm will be applied to systems of equations that have no solution, and in this case we must finish iterating at some stage. Therefore the process finishes if the gradient of $F(\underset{\sim}{x})$ becomes very small. A consequence is that on some awkward problems the method will converge to a point at which the equations are not satisfied, although there is a solution, so we are admitting that we are unable to solve the familiar difficulty of recognising whether or not a point of convergence is a global minimum of $F(\underset{\sim}{x})$. One recourse, if the subroutine fails to obtain a solution to the equations, is to try different initial estimates of $\underset{\sim}{x}$.

To approximate derivatives numerically, the algorithm uses one of the class of methods described by Broyden (1965,1967). Not only is the Jacobian matrix J, see equation (2), approximated, but also we keep an estimate of J^{-1}, so that we can solve equation (3) in only of order n^2 computer operations. This saving of work is consistent with the fact that Broyden's scheme requires only $O(n^2)$ operations to revise the estimates to J and to J^{-1}. Thus we obtain an important advantage over some other algorithms, for the direct solution of equation (3) or (9) requires of order n^3 computer operations.

Another important property of the new algorithm is that, unlike earlier methods by the author, it does not search for best points along straight lines in the space of the variables $\underset{\sim}{x}$. Instead we heed the advice of Broyden (1965), and usually the functions $f_k(\underset{\sim}{x})$ $(k=1,2,\ldots,n)$ are calculated for only one value of $\underset{\sim}{x}$ on each iteration.

The sections of the report are arranged so that it is easy for the reader, who just wishes to make use of the subroutine, to omit the details of the method. Indeed the information that he requires, such as a description of the programme parameters, is given in Section 2. We summarize the method of calculation in Section 3, and give details in the subsequent five sections: Section 4 describes the choice of the correction vector $\underset{\sim}{\delta}$, having both Newton iteration and steepest descent characteristics, Section 5 discusses the adjustment of a step-length parameter Δ, Section 6 gives the formulae that are used to revise the numerical approximations to the Jacobian matrix and its inverse, Section 7 describes a necessary device which ensures that successive corrections $\underset{\sim}{\delta}$ are not linearly dependent, and Section 8 assembles the various remaining details. In an appendix there is a Fortran listing of the algorithm, and a test programme is included as well to assist those who wish to try the method. Other numerical examples appear in Section 9, and a summary of the results concludes the paper.

2. The parameters of the subroutine

The name of the subroutine and its parameters are:

SUBROUTINE NS01A (N,X,F,AJINV,DSTEP,DMAX,ACC,MAXFUN,IPRINT,W).

The name NS01A is chosen to conform with the other names of the programmes in the Harwell Subroutine Library. Before calling the routine the user must assign values to the parameters N, X(1), X(2),...,X(N),DSTEP,DMAX,ACC,MAXFUN, and IPRINT, and we now specify the purpose of each of these quantities.

N is just the number of equations, and it must be greater than one.

X is a one-dimensional array for the variables of the equations, and initially {X(1),X(2),...,X(N)} must be set to an estimate of the solution. This estimate is refined during the execution of NS01A, so that when the subroutine finishes it is usually set to the calculated value of the solution. However, if it happens that execution of the subroutine is terminated because an error condition is found, then {X(1),X(2),...,X(N)} is set to the best estimate of the solution, according to the value of $F(\underset{\sim}{x})$, defined in equation (6). These error conditions are specified below.

DSTEP must be set to a number that is a moderate step-length to use to approximate first derivatives of the functions by differences between function values. For instance we suppose that

$$\frac{\partial f_1(\underset{\sim}{x})}{\partial x_1} \approx \left[f_1(x_1 + \text{DSTEP}, x_2, \ldots, x_n) - f_1(\underset{\sim}{x}) \right] \Big/ \text{DSTEP}. \qquad (12)$$

Note that the one increment DSTEP is used for all the variables, so it is necessary for the user to choose the variables so that their magnitudes are similar. We have deliberately introduced this requirement because there are other good reasons for having all the variables of the same size.

DMAX must be set to a generous estimate of the "distance" of the solution from the initial guess of $(x_1,x_2,\ldots,x_n)$; we use the Euclidean metric, so the distance between $\underset{\sim}{x}$ and $\underset{\sim}{y}$ is

$$d(\underset{\sim}{x},\underset{\sim}{y}) = \left[\sum_{i=1}^{n} (x_i - y_i)^2 \right]^{\frac{1}{2}}. \qquad (13)$$

DMAX is used in two separate ways. Firstly it is arranged that the change in the vector $\underset{\sim}{x}$ at each iteration does not exceed DMAX. Secondly there is an error return giving the diagnostic "Error return from NS01A because a nearby stationary point of $F(\underset{\sim}{x})$ is predicted" if it happens that the gradient of $F(\underset{\sim}{x})$ become so small that it is predicted that steps that are much larger than DMAX are needed. DMAX must be greater than DSTEP.

ACC specifies the accuracy that is required in the solution. A normal return to the calling programme occurs when a vector $\underset{\sim}{x}$ is found such that we have the inequality

$$\sum_{k=1}^{n} \left[f_k(\underset{\sim}{x}) \right]^2 \leqslant \text{ACC}. \tag{14}$$

Note that this convergence criterion is such that it is sensible to scale the functions $f_k(\underset{\sim}{x})$ $(k=1,2,\ldots,n)$ to have similar magnitudes. There are other good reasons for defining the equations so that they have comparable left hand sides.

The parameter MAXFUN is included to ensure that the execution of NS01A will finish. The number of times that the left hand sides of the equations are worked out is counted, and if this count attains the value MAXFUN, there is an error return following the diagnostic "Error return from NS01A because there have been MAXFUN calls of CALFUN". Experiments show that often the subroutine requires fewer than 10*N evaluations of the left hand sides, but MAXFUN should be set to a greater number, unless the total amount of calculation needs to be limited by a conservative value of MAXFUN.

The parameter IPRINT must have the value zero or one. If it is zero, there is no printed output from the subroutine, except for a message if an error condition is found. However, if IPRINT=1, then, each time the functions $f_k(\underset{\sim}{x})$ $(k=1,2,\ldots,n)$ are calculated, the values of the functions and of the variables $x_1,x_2,\ldots,x_n$ are printed. Examples of this output are given in the appendix.

The other parameters of NS01A are F, AJINV and W, and they are all the names of arrays. The numbers in the arrays will be changed during the execution of the subroutine.

The array F is one-dimensional, and it must contain at least N elements. Usually F(k) is set to a calculated value of $f_k(\underset{\sim}{x})$ $(k=1,2,\ldots,n)$, but the array also serves as working space during some of the operations of the subroutine. When the subroutine finishes, the array F contains the values of the functions that are obtained for the vector of variables $(x_1,x_2,\dots,x_n)$ that is present in the array X.

AJINV is an N $\times$ N two dimensional array, and when the subroutine finishes it contains the elements of an approximation to the matrix J^{-1}, where J is the Jacobian (2). We include it in the parameter list in case an estimate is required of the accuracy of the solution $\underset{\sim}{x}$: equation (3) suggests that after a normal return from the subroutine the error in x_j is approximately equal to

$$-\sum_{k=1}^{n} \text{AJINV}(j,k)\ F(k),\ j=1,2,\dots,n. \tag{15}$$

Also equation (3) suggests that if the functions $f_k(\underset{\sim}{x})$ are changed by small amounts $\eta_k(k=1,2,\dots,n)$, then the change in the required value of x_j is approximately

$$-\sum_{k=1}^{n} \text{AJINV}(j,k)\ \eta_k,\ j=1,2,\dots,n. \tag{16}$$

Thus one can estimate the sensitivity of $\underset{\sim}{x}$ to any uncertainty in the specification of $f_k(\underset{\sim}{x})$, $k=1,2,\dots,n$.

The last array, W, must be one-dimensional, and its elements are used by the subroutine for working space. The number of elements required is $n(2n+5)$. It happens that, when NS01A finishes, the most recent approximation to the Jacobian matrix is present at the beginning of the array:

$$J_{kj} \approx W(k[n-1] + j),\quad j=1,2,\dots,n;\quad k=1,2,\dots,n. \tag{17}$$

The functions $f_k(\underset{\sim}{x})$ $(k=1,2,\dots,n)$, defining the system of nonlinear equations, must be defined by another Fortran subroutine, called CALFUN. It has three parameters

SUBROUTINE CALFUN (N,X,F),

and the names of these parameters accord with those of NS01A, so N is the number of equations, and X and F are one-dimensional arrays. CALFUN is called whenever NS01A requires the functions $f_k(\underset{\sim}{x})$ to be calculated for some vector $\underset{\sim}{x}$, and the components of $\underset{\sim}{x}$ are given in the array X. CALFUN must set the components of F to the function values

$$F(k) = f_k(X(1),\ X(2), \ldots, X(N)), \quad k=1,2,\ldots,N. \tag{18}$$

An example of the subroutine CALFUN is given in the appendix.

We have mentioned that sometimes there are error returns from NSO1A, and we have already given two instances when they occur. In two other situations the execution of the subroutine is terminated before the required accuracy is obtained. The first is when, in spite of the convergence theorems and the various strategies of the algorithm, the subroutine is persistently unsuccessful in its attempts to decrease the sum of squares (6). Specifically if $F(\underset{\sim}{x})$ fails to decrease on n+4 consecutive iterations when a decrease is predicted, and if on each of these occasions the vector $\underset{\sim}{x}$ is within the distance DSTEP of the most successful vector of variables, then there is an error return after the diagnostic "Error return from NSO1A because N+4 calls of CALFUN failed to improve the residuals". It may happen because the value of DSTEP is too large (we explain this remark in Section 8), or because of programming mistakes, or because the rounding errors of the computer are so large that the required accuracy, given in expression (14), cannot be obtained.

These conditions can also cause the other error return, which is indicated by the diagnostic "Error return from NSO1A because $F(\underset{\sim}{x})$ failed to decrease using a new Jacobian". It happens when a completely new Jacobian has just been obtained, by differencing along the coordinate directions in the space of the variables (see Section 8). In this case we expect to have a reliable prediction of the behaviour of $F(\underset{\sim}{x})$, provided that the distance of $\underset{\sim}{x}$ from the point at which the Jacobian was calculated does not exceed DSTEP. Therefore if this hope is not realised, and we find that $F(\underset{\sim}{x})$ does not become smaller although a decrease is predicted, the error return is made.

The reader who does not use the A.E.R.E. operating system must also note that a subroutine for inverting a matrix is required by NSO1A. The Fortran listing in the appendix uses another Harwell library programme, named MBO1B, the 92nd instruction being

CALL MBO1B (AJINV, N, N).

The effect of this statement is to replace the elements of an $N \times N$ matrix by the elements of its inverse. Specifically, when MBO1B is called, AJINV(i,j) is set to J_{ij}, and when the execution of MBO1B is finished, the element AJINV(i,j) is set to J^{-1}_{ij} $(i,j = 1,2,\ldots,n)$.

3. <u>An outline of the algorithm</u>

The next six sections specify the formulae that are used by the algorithm, and they justify the decisions of the Fortran subroutine, which is listed in the appendix. This section begins the description by summarising an iteration, in order to identify the four main parts of the calculation, which are considered separately in Sections 4,5,6 and 7. Note that this introductory summary ignores some important points, because its purpose is just to provide a simple coherent picture of the method, so that the reader can relate the subsequent details to the general strategy.

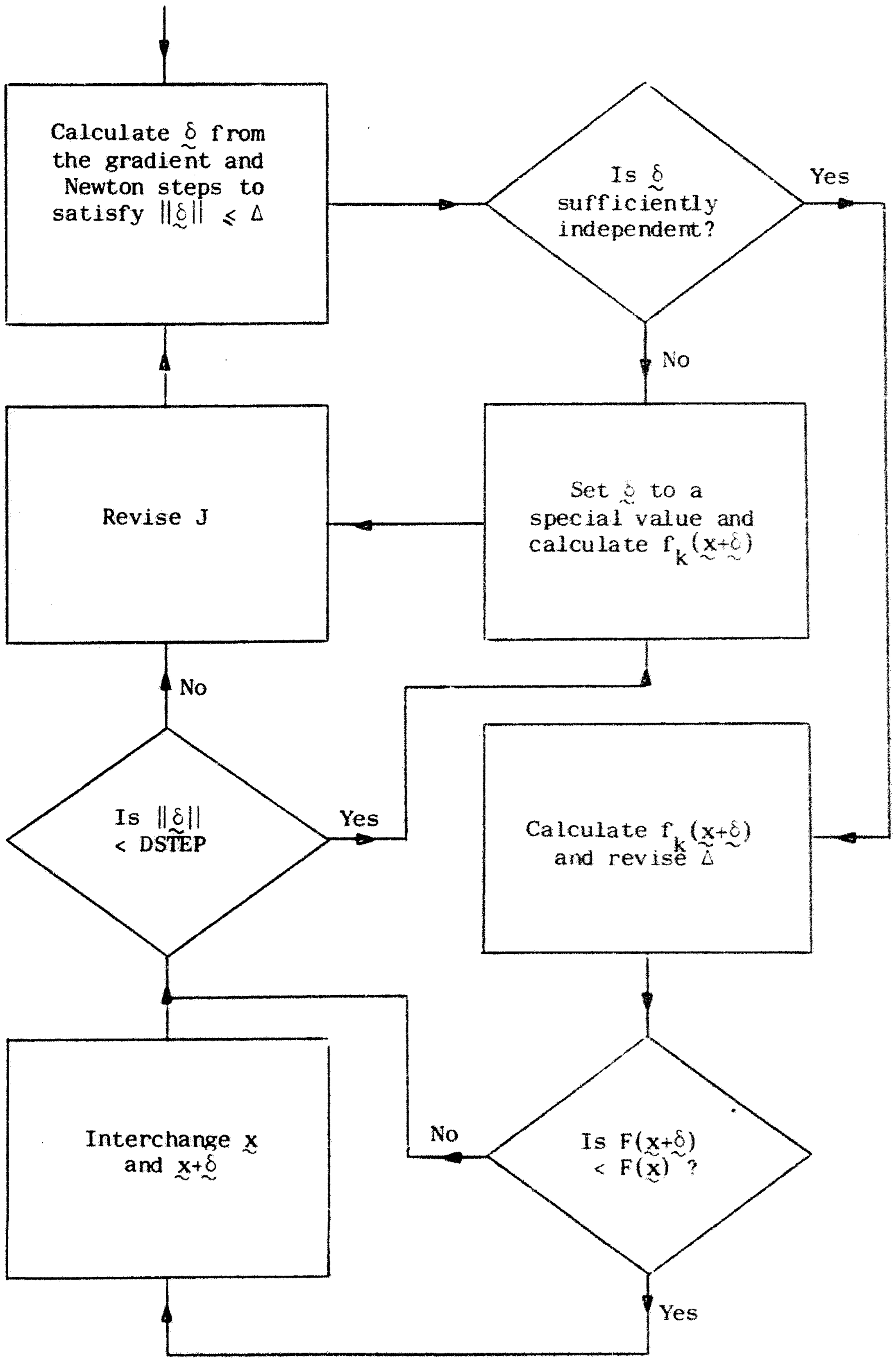
To begin an iteration of the method the following data is required: (i) a vector of variables $\underset{\sim}{x}$, which is an estimate of the solution of the equations, and the corresponding function values $f_k(\underset{\sim}{x})$ $(k=1,2,\ldots,n)$, (ii) an approximation to the Jacobian matrix (the Jacobian is defined by equation (2), but now we use the notation J for the approximation), (iii) the matrix J^{-1}, (iv) a matrix Ω of n directions in the space of the variables, and an associated vector of integers $\underset{\sim}{\omega}$, and (v) a step length Δ. The calculation of an iteration is outlined in Figure 1.

The figure shows that the first operation of an iteration is to calculate a correction $\underset{\sim}{\delta}$ to apply to the approximation $\underset{\sim}{x}$. This calculation is described in Section 4, so for the present we note that it is a compromise between the Newton iteration (see equation (3)) and the method of steepest descents applied to the sum of squares $F(\underset{\sim}{x})$. The balance between these two methods is governed by the step length Δ, so that if Δ is sufficiently large the correction $\underset{\sim}{\delta}$ is the pure Newton step; thus fast ultimate convergence to the solution can be obtained. For very small values of Δ, the vector $\underset{\sim}{\delta}$ is exactly a multiple of the predicted gradient of $F(\underset{\sim}{x})$, and in all cases the correction is such that the sum of squares $F(\underset{\sim}{x}+\underset{\sim}{\delta})$ is predicted to be less than $F(\underset{\sim}{x})$. However this prediction may not be realised. It can be wrong because the non-linear behaviour of the functions $f_k(\underset{\sim}{x})$ has already caused J to deviate from the actual Jacobian at $\underset{\sim}{x}$, and, even if J is exact, it can be wrong because $f_k(\underset{\sim}{x})$ does not vary linearly between $\underset{\sim}{x}$ and $(\underset{\sim}{x}+\underset{\sim}{\delta})$.

We will introduce the criterion which decides whether $\underset{\sim}{\delta}$ is "sufficiently independent" at the end of this section; it depends on the elements of $\underset{\sim}{\omega}$ and of Ω. Usually we find that $\underset{\sim}{\delta}$ passes the test, in which case the next step of the flow diagram is to calculate $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$, $k=1,2,\ldots,n$, and to revise Δ.

Figure 1

Summary of an iteration



Calculate $\underset{\sim}{\delta}$ from the gradient and Newton steps to satisfy $\|\underset{\sim}{\delta}\| \leqslant \Delta$
Is $\underset{\sim}{\delta}$ sufficiently independent?
Yes
No
Set $\underset{\sim}{\delta}$ to a special value and calculate $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$
Revise J
No
Is $\|\underset{\sim}{\delta}\|$ < DSTEP
Yes
Calculate $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$ and revise Δ
Interchange $\underset{\sim}{x}$ and $\underset{\sim}{x}+\underset{\sim}{\delta}$
No
Is $F(\underset{\sim}{x}+\underset{\sim}{\delta})$ < $F(\underset{\sim}{x})$?
Yes

The method for revising Δ, described in Section 5, is intended to find such a small step-length that each iteration is successful in obtaining the inequality

$$F(\underset{\sim}{x}+\underset{\sim}{\delta}) < F(\underset{\sim}{x}). \tag{19}$$

Therefore Δ is reduced if the condition (19) is not satisfied, except that we do not let Δ become less than the subroutine parameter DSTEP, which we defined in Section 2. Also we are prepared to increase Δ, because an extravagant number of iterations is required if Δ is too small. The criterion for increasing Δ depends on the accuracy of the approximations

$$f_k(\underset{\sim}{x}+\underset{\sim}{\delta}) - f_k(\underset{\sim}{x}) \approx \sum_{j=1}^{n} J_{kj}\, \delta_j, \quad k=1,2,\ldots,n, \tag{20}$$

for usually the estimates (20) are close only if the size of Δ is conservative.

The decision of the flow diagram to interchange $\underset{\sim}{x}$ and $(\underset{\sim}{x}+\underset{\sim}{\delta})$, if the inequality (19) holds, provides the best approximation to the solution of the equations for the next iteration (of course the function values $f_k(\underset{\sim}{x})$ and $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$, $k=1,2,\ldots,n$, are interchanged as well). However an important point to notice is that the interchange does not take place if $\underset{\sim}{\delta}$ is set to a special value in order to provide a "sufficiently independent" correction to $\underset{\sim}{x}$. The reason for this rather contentious decision is that, if we were prepared to make the interchange after using a special value of $\underset{\sim}{\delta}$, then a certain theorem (Powell, 1969) would not apply to the algorithm. The theorem states that, given exact arithmetic, the method of the subroutine causes all the vectors $\underset{\sim}{x}$ to be within a finite distance of the initial guess of the solution, so, even if the only solution to the equations is at infinity, we expect that the algorithm will not cause the components of $\underset{\sim}{x}$ to become too large for the computer. We prefer not to invalidate this theorem.

The method for revising the Jacobian approximation J depends on the vector $\underset{\sim}{\delta}$, and on the differences

$$\gamma_k = f_k(\underset{\sim}{x}+\underset{\sim}{\delta}) - f_k(\underset{\sim}{x}), \quad k=1,2,\ldots,n. \tag{21}$$

These differences are liable to be dominated by computer rounding errors if $\underset{\sim}{\delta}$ is too small, so we include the precaution of setting $\underset{\sim}{\delta}$ to a special value if $||\underset{\sim}{\delta}|| <$ DSTEP. Only on this occasion is CALFUN called twice during an iteration.

The details of the calculation of the new Jacobian approximation, J^* say, are given in Section 6. We want to satisfy the equations

$$\sum_{j=1}^{n} J^*_{kj}\,\delta_j = \gamma_k, \quad k=1,2,\ldots,n, \tag{22}$$

because they would hold if J^* were exact and the functions $f_k(\underset{\sim}{x})$ were linear, so usually we apply the formula (Broyden, 1965)

$$J^* = J + (\underset{\sim}{\gamma} - J\underset{\sim}{\delta})\,\underset{\sim}{\delta}^T/||\underset{\sim}{\delta}||^2 . \tag{23}$$

However equation (23) can cause J^* to be singular, and we require the elements of $(J^*)^{-1}$. Therefore sometimes we apply a part of the full correction, in order to force non-singularity.

The correction (23) has a special property, which is the reason for the introduction of the device (depending on Ω and $\underset{\sim}{\omega}$) to ensure "sufficient independence" of the successive displacements $\underset{\sim}{\delta}$. The property is that the results of applying both the old and the new Jacobian approximations to any vector that is orthogonal to $\underset{\sim}{\delta}$ are the same. Therefore, if it happens that all the vectors $\underset{\sim}{\delta}$ are linearly dependent, there is some non-zero vector $\underset{\sim}{\eta}$ (orthogonal to all the vectors $\underset{\sim}{\delta}$) such that $J\underset{\sim}{\eta}$ is the same for all the Jacobian approximations. This is unsatisfactory because the non-linearity of the equations causes the true Jacobian to change with $\underset{\sim}{x}$, and probably the true value of $J\underset{\sim}{\eta}$ changes as well. Therefore, if necessary, we deliberately introduce extra displacements to ensure that most sets of $(2n+1)$ successive vectors $\underset{\sim}{\delta}$ span the full space of the variables. The number $(2n+1)$ was chosen intuitively, and numerical experiments show that the extra programming that is needed to obtain "sufficient independence" is worthwhile.

The details of the test for independence are described in Section 7. The matrix Ω and the vector $\underset{\sim}{\omega}$ contain information about the corrections $\underset{\sim}{\delta}$ that were used in recent iterations, from which we can find out the extent of linear dependence in the successive vectors $\underset{\sim}{\delta}$. In accordance with the flow diagram of Figure 1, this information is used to test the vector $\underset{\sim}{\delta}$, calculated by the method of Section 4, for "sufficient independence", and if the test fails, then we change $\underset{\sim}{\delta}$ so that it becomes practically orthogonal to the most recent 2n correction vectors $\underset{\sim}{\delta}$, for the resultant revision of J is probably overdue. Like the method for changing J, this test and the revision of Ω and $\underset{\sim}{\omega}$ require of order n^2 computer operations.

Section 8 includes details of the procedures for starting and finishing the algorithm.

4. The calculation of $\underset{\sim}{\delta}$

The calculation of $\underset{\sim}{\delta}$ is carried out in instructions 93-156 of the Fortran listing of the subroutine, but these instructions include some other calculation. We begin by predicting both the Newton correction, $\underset{\sim}{\nu}$ say, to $\underset{\sim}{x}$, and also the steepest descent direction $\underset{\sim}{g}$ of $F(\underset{\sim}{x})$: instruction 102 calculates

$$\nu_i = - \sum_{j=1}^{n} J^{-1}_{ij} f_j(\underset{\sim}{x}), \quad i=1,2,\ldots,n, \tag{24}$$

and instruction 101 calculates

$$g_i = - \sum_{j=1}^{n} J_{ji} f_j(\underset{\sim}{x}), \quad i=1,2,\ldots,n. \tag{25}$$

The purpose of Δ is to limit the size of the correction vector $\underset{\sim}{\delta}$, and we impose the condition $||\underset{\sim}{\delta}|| \leq \Delta$, the length $||.||$ being Euclidean, in accordance with equation (13). Because of the good convergence properties of the Newton iteration, we would like to set $\underset{\sim}{\delta}$ to $\underset{\sim}{\nu}$. Therefore if the inequality

$$||\underset{\sim}{\nu}|| \leq \Delta \tag{26}$$

holds, which is tested by instruction 120, we let $\underset{\sim}{\delta}=\underset{\sim}{\nu}$. Otherwise we include a multiple of $\underset{\sim}{g}$ in $\underset{\sim}{\delta}$, and the correction vector satisfies the condition $||\underset{\sim}{\delta}|| = \Delta$.

The correction is just a positive multiple of $\underset{\sim}{g}$, if this choice is not greater than the predicted displacement to the minimum of $F(\underset{\sim}{x})$ along the steepest descent direction. This minimum is at $\underset{\sim}{x}+\mu\underset{\sim}{g}$, where μ is defined by the equation

$$\mu = ||\underset{\sim}{g}||^2 \Big/ ||J\underset{\sim}{g}||^2. \tag{27}$$

Therefore instruction 139 tests the inequality

$$\mu||\underset{\sim}{g}|| \geq \Delta, \tag{28}$$

and, if it holds, we set

$$\underset{\sim}{\delta} = \Delta\underset{\sim}{g}/||\underset{\sim}{g}|| \; . \tag{29}$$

If neither inequality (26) nor inequality (28) is satisfied, we let $\underset{\sim}{\delta}$ be on the straight line joining the points $\mu\underset{\sim}{g}$ and $\underset{\sim}{\nu}$, so we need the value of the positive number θ that is defined by the equation

$$||(1-\theta)\; \mu\underset{\sim}{g} + \theta\underset{\sim}{\nu}|| = \Delta. \tag{30}$$

Straightforward algebra gives the solution

$$\theta = \frac{\Delta^2 - ||\mu\underset{\sim}{g}||^2}{(\underset{\sim}{\nu}-\mu\underset{\sim}{g},\mu\underset{\sim}{g}) + [\{(\underset{\sim}{\nu},\mu\underset{\sim}{g}) - \Delta^2\}^2 + \{||\underset{\sim}{\nu}||^2 - \Delta^2\}\{\Delta^2 - ||\mu\underset{\sim}{g}||^2\}]^{\frac{1}{2}}} \tag{31}$$

and instruction 153 calculates the components

$$\delta_i = (1-\theta)\; \mu g_i + \theta\nu_i, \quad i=1,2,\dots,n. \tag{32}$$

This specification of $\underset{\sim}{\delta}$ is preferred to the Levenberg/Marquardt choice, because it can be calculated in of order n^2 computer operations. However we cannot match the elegant theorem that supports Marquardt's method (1963). This is not a cause of anxiety, because we have retained the most important feature, which is that if the length of the correction must be small, its direction is biased towards the steepest descent direction of $F(\underset{\sim}{x})$.

5. The revision of Δ

Usually we try to adjust Δ so that it is as large as possible, subject to the condition that each Jacobian approximation provides a good prediction of the differences $\{f_k(\underset{\sim}{x}+\underset{\sim}{\delta}) - f_k(\underset{\sim}{x})\}$, $k=1,2,\ldots,n$, because we would like to decrease the sum of squares $F(\underset{\sim}{x})$ on every iteration, without taking extravagantly small steps. The initial choice of Δ is specified in Section 8, and now we just discuss the method for changing its value. This adjustment is made by instructions 241 to 256 of the Fortran listing.

Before beginning this part of the subroutine, we have calculated the function values $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$, $k=1,2,\ldots,n$, and the sum of squares $F(\underset{\sim}{x}+\underset{\sim}{\delta})$. Also we have predicted these quantities, using the Jacobian approximation, the predictions being

$$\phi_k = f_k(\underset{\sim}{x}) + \sum_{j=1}^{n} J_{kj}\,\delta_j \approx f_k(\underset{\sim}{x}+\underset{\sim}{\delta}) \tag{33}$$

and

$$\Phi = \sum_{k=1}^{n} \phi_k^2 \approx F(\underset{\sim}{x}+\underset{\sim}{\delta}) \ . \tag{34}$$

The method for revising Δ depends on the goodness of these predictions, and in particular we note that $\underset{\sim}{\delta}$ is calculated in such a way that $\Phi < F(\underset{\sim}{x})$, so we expect the new value of the sum of squares to be less than the old one.

If the actual change in the sum of squares is worse than the predicted change, it is due to a combination of the two factors which we mentioned in Section 3. One is that, even if the current Jacobian approximation is correct, there can be errors in the expression (33), due to the non-linearities of the functions $f_k(\underset{\sim}{x})$. The second is that there are errors in the Jacobian approximation itself, due to the fact that J is assembled gradually, so some of the information in J is liable to be rather inaccurate due to the fact that it was obtained at points that are remote from the current value of $\underset{\sim}{x}$. Both errors usually become smaller if Δ is decreased.

Therefore the subroutine reduces Δ if $F(\underset{\sim}{x}+\underset{\sim}{\delta}) \geq F(\underset{\sim}{x})$, but also Δ may be made smaller when $F(\underset{\sim}{x}+\underset{\sim}{\delta}) < F(\underset{\sim}{x})$. The reason for this is that we are not satisfied if a reduction in the sum of squares is much less than the predicted reduction, so instruction 242 tests the inequality

$$F(\underset{\sim}{x}+\underset{\sim}{\delta}) > F(\underset{\sim}{x}) - 0.1\,\{F(\underset{\sim}{x}) - \Phi\}\ , \qquad (35)$$

and if it holds we replace Δ by the value

$$\max\ (\tfrac{1}{2}\Delta,\ \text{DSTEP}). \qquad (36)$$

We do not let Δ become less than DSTEP, because, if the user of the subroutine follows the advice of Section 2, the lower bound on Δ will permit an adequate Jacobian approximation. A lower bound is needed, because there is a danger that the value of expression (21) will be dominated by computer round-off errors when $\underset{\sim}{\delta}$ is very small.

If the inequality (35) is not satisfied, either the value of Δ remains the same, or it is increased. The justification that we give to support the method for increasing Δ is tenuous, but numerical examples show that the results of the method are quite satisfactory.

The basis of the method is that we attribute the differences $\{f_k(\underset{\sim}{x}+\underset{\sim}{\delta}) - \phi_k\}$, $k=1,2,\ldots,n$, to terms that are of order Δ^2, so if we multiply Δ by the factor λ, we expect the differences to be multiplied by about λ^2. Guided by this assumption, we estimate the value of λ that would just cause the condition (35) to fail. We ignore the fact that a larger value of Δ will lead to a different value of $\underset{\sim}{\delta}$, and we calculate the value of λ that makes the expression

$$\sum_k \Big[\,|f_k(\underset{\sim}{x}+\underset{\sim}{\delta})| + (\lambda^2-1)\,|f_k(\underset{\sim}{x}+\underset{\sim}{\delta}) - \phi_k|\,\Big]^2 \qquad (37)$$

equal to the right-hand side of the inequality (35). This value is obtained by instruction 252, which sets

$$\lambda^2 = 1 + \frac{\text{DMULT}}{\text{SP} + (\text{SP.SP} + \text{DMULT.SS})^{\frac{1}{2}}}\ , \qquad (38)$$

where

$$\left.\begin{aligned} DMULT &= F(x) - 0.1\,\{F(x) - \Phi\} - F(x+\delta) \\ SP &= \sum_{k=1}^{n} |f_k(x+\delta)\,\{f_k(x+\delta) - \phi_k\}| \\ SS &= \sum_{k=1}^{n} \{f_k(x+\delta) - \phi_k\}^2 \end{aligned}\right] . \qquad (39)$$

The criterion for increasing Δ depends on the value of expression (38).

We calculate λ whenever condition (35) shows that Δ is not too large, but by trying different strategies on some test problems, we found that it is best not to scale Δ by λ whenever λ is calculated. One reason for this is that even when Δ is reduced to expression (36), the function values $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$, $k=1,2,\ldots,n$, can provide such a beneficial change to the Jacobian approximation, that, on the next iteration, multiplying the reduced step length by λ would restore Δ to about its original value. Usually this step length would have to be decreased again, so some of the strategies that we tried caused inefficient oscillatory behaviour.

We found that the oscillations are avoided (except in the case of an extreme example, reported in Section 9) by the simple expedient of increasing Δ only when <u>two</u> values of λ have been calculated, and they must both have been obtained since the last reduction in Δ. The factor by which Δ is multiplied is equal to the lesser value of λ, except that we are cautious, and ensure that the factor is never greater than two, and that Δ is bounded above by DMAX. To apply this strategy we introduce a parameter τ (the variable "TINC" of the subroutine is equal to τ^2), which is set to the value one both before the first iteration, and also whenever the step length is reduced. Immediately after calculating λ, we obey the instructions

$$\left.\begin{aligned} \mu &= \min\,(2,\lambda,\tau) \\ \tau &= \lambda/\mu \\ \Delta &= \min\,(\mu\Delta, DMAX) \end{aligned}\right] , \qquad (40)$$

and it should be clear that they increase Δ in the required way. Note that we permit consecutive iterations to increase the step length.

The value of Δ may also be revised in the block of programme that calculates $\underset{\sim}{\delta}$, according to the method of Section 4: instruction 121 changes Δ to the value $\max(||\underset{\sim}{\nu}||_2, \text{DSTEP})$ if $\underset{\sim}{\delta}$ is set to the full Newton-Raphson correction (24). We do this because consecutive successful Newton-Raphson corrections tend to decrease in length (due to the quadratic convergence properties), and we do not want Δ to be much larger than $||\underset{\sim}{\delta}||_2$.

6. The revision of J

The subroutine revises the matrices J and J^{-1} in instructions 273-309. We use the notation H in place of J^{-1}, and we let the revised matrices be J^* and H^*. Already we have stated, in equation (23), that the formula

$$J^* = J + (\underset{\sim}{\gamma} - J\underset{\sim}{\delta})\, \underset{\sim}{\delta}^T / ||\underset{\sim}{\delta}||^2 \tag{41}$$

is usually used, and the companion formula defining H^* is

$$H^* = H + (\underset{\sim}{\delta} - H\underset{\sim}{\gamma})\, \underset{\sim}{\delta}^T H / (\underset{\sim}{\delta}^T H \underset{\sim}{\gamma}) \ , \tag{42}$$

where the superscript "T" indicates a row vector. However it can happen that the scalar $(\underset{\sim}{\delta}^T H \underset{\sim}{\gamma})$ is zero, so the subroutine takes special steps to prevent a very large increase in the size of the elements of H^*.

In fact the formulae that are applied depend on a parameter α, and the actual revision is specified by the equations

$$\left.\begin{aligned} J^* &= J + \alpha(\underset{\sim}{\gamma} - J\underset{\sim}{\delta})\, \underset{\sim}{\delta}^T / ||\underset{\sim}{\delta}||^2 \\ H^* &= H + \alpha \frac{(\underset{\sim}{\delta} - H\underset{\sim}{\gamma})\, \underset{\sim}{\delta}^T H}{\alpha(\underset{\sim}{\delta}^T H \underset{\sim}{\gamma}) + (1-\alpha)\, ||\underset{\sim}{\delta}||^2} \end{aligned}\right] . \tag{43}$$

To avoid singularity, instruction 295 tests the inequality

$$|(\underset{\sim}{\delta}^T H \underset{\sim}{\gamma})| \geq 0.1\ ||\underset{\sim}{\delta}||^2 \ , \tag{44}$$

and if it holds the value $\alpha=1$ is used. Otherwise instruction 296 sets $\alpha=0.8$, so in all cases we ensure that the modulus of the denominator of the expression for H^* is at least $0.1\ ||\underset{\sim}{\delta}||^2$. The number 0.1 was chosen empirically.

Note that the application of the formulae requires only of order n^2 computer operations. Note also that if H is the exact inverse of J, and if there are no errors in the calculation, then H^* is the exact inverse of J^*. We show later that computer round-off errors do not spoil the calculation.

We want the formula (43) to have the property that J^* is better than J as an approximation to the true Jacobian matrix. A detailed discussion of this question is given by Powell (1969), and he shows that if the vectors $\underset{\sim}{x}$, obtained by the successive iterations of the algorithm, converge to a point $\bar{\underset{\sim}{x}}$, then, under mild differentiability conditions on $f_k(\underset{\sim}{x})$, k=1,2,...,n, the successive Jacobian approximations converge to the actual Jacobian at $\bar{\underset{\sim}{x}}$. To make this statement plausible, and to give the reader some confidence in the formulae (43), we repeat a remark (Broyden, 1965) that applies in the simple case when the functions $f_k(\underset{\sim}{x})$ are linear: say they are defined by the formula

$$f_k(\underset{\sim}{x}) = c_k + \sum_{j=1}^{n} \bar{J}_{kj}\, x_j, \quad k=1,2,\ldots,n, \tag{45}$$

so $\bar{J}$ is the true Jacobian matrix, which is independent of $\underset{\sim}{x}$. Broyden notes that, because the definition (21) provides the relation

$$\underset{\sim}{\gamma} = \bar{J}\,\underset{\sim}{\delta}\ , \tag{46}$$

the formula (43) leads to the identity

$$(J^* - \bar{J}) = (J - \bar{J})\left(I - \alpha\frac{\underset{\sim}{\delta}\,\underset{\sim}{\delta}^T}{||\underset{\sim}{\delta}||^2}\right). \tag{47}$$

This relation between the error $(J^*-\bar{J})$ and the error $(J-\bar{J})$ is very satisfactory, because it gives the inequality

$$\sum_{ij}\left[(J^* - \bar{J})_{ij}\right]^2 \leqslant \sum_{ij}\left[(J - \bar{J})_{ij}\right]^2, \tag{48}$$

and the inequality is strict unless

$$J\,\underset{\sim}{\delta} = \bar{J}\,\underset{\sim}{\delta}\ . \tag{49}$$

Therefore an iteration reduces the Frobenius norm of the error of the Jacobian approximation, unless the predictions (33) and (34) are exact, in which case we expect to obtain a substantial reduction in $F(\underset{\sim}{x})$.

Because of the above remarks, the Jacobian approximations J are adequate. But we need to consider the possibility that rounding errors of the computation may cause the matrices H to be useless. Because we modify the matrix H on every iteration, and every modification introduces some error, we are concerned that after many iterations the cumulative effect of small rounding errors may be disastrous. Fortunately this does not happen, because a property of the pair of formulae (41) and (42) is that the discrepancies between J^* and $(H^*)^{-1}$ tend to be less than those between J and H^{-1} in the following sense. Even if H is not the inverse of J, then the identity (Powell, 1968b)

$$\left(J^* - \{H^*\}^{-1}\right) = (J - H^{-1})\left(I - \frac{\underset{\sim}{\delta}\,\underset{\sim}{\delta}^T}{\underset{\sim}{\delta}^T\underset{\sim}{\delta}}\right) \tag{50}$$

is satisfied, provided that exact arithmetic is used. Therefore the discrepancy $(J-H^{-1})$ is multiplied by a projection matrix, which suppresses the accumulation of error. Moreover, because the method of the algorithm maintains linear independence in the successive directions $\underset{\sim}{\delta}$, the cumulative effect of the projection matrices is particularly favourable.

7. Maintaining linear independence

We remarked, in Section 3, that the method for revising the Jacobian approximation is such that we should avoid linear dependence in the directions $\underset{\sim}{\delta}$ that are generated by the successive iterations of the algorithm. The calculation of Section 4 often tends to provide dependent directions, so, in accordance with the flow diagram of Figure 1, the subroutine inspects the directions $\underset{\sim}{\delta}$, and occasionally extra directions are introduced to ensure that independence is maintained. This part of the calculation is carried out by instructions 150-228 of the Fortran listing in the Appendix, and now we describe the details of the method that is used.

For the purposes of the algorithm we depart from the usual strict definition of "linear dependence", because we want "independent directions" to be separated by a substantial amount. We say that the vector $\underset{\sim}{\delta}$ is independent of a set of directions, $(\underset{\sim}{d}_1,\underset{\sim}{d}_2,\ldots,\underset{\sim}{d}_j)$ say, only if the <u>least</u> angle between $\underset{\sim}{\delta}$ and some vector in the space spanned by the directions is not less than thirty degrees. In the sense of this definition, the subroutine ensures that, for most values of $k > 2n$, the directions $\underset{\sim}{\delta}^{(k-2n)}, \underset{\sim}{\delta}^{(k-2n+1)}, \ldots, \underset{\sim}{\delta}^{(k)}$ span the full space of the variables, where $\underset{\sim}{\delta}^{(t)}$ is the direction that is used in the revision of the Jacobian matrix on the t^{th} iteration.

The exceptional values of k occur because, if $\underset{\sim}{\delta}^{(k)}$ is equal to $\underline{v}$ (see equation (24)), then we accept $\underset{\sim}{\delta}^{(k)}$ even if it is dependent on the set of directions $(\underset{\sim}{\delta}^{(k-2n)}, \underset{\sim}{\delta}^{(k-2n+1)}, \ldots, \underset{\sim}{\delta}^{(k-1)})$. We make this decision because $\underline{v}$ is such that the sum of squares of residuals $F(\underset{\sim}{x}+\underset{\sim}{v})$ is predicted to be equal to zero. If this prediction is a good one, then our choice of $\underset{\sim}{\delta}^{(k)}$ is very successful, and if it is not realised, then equation (47) shows that the resultant revision of the Jacobian matrix is substantial. Therefore, in both cases, the iteration is useful in some way.

The $n\times n$ matrix Ω and the vector $\underset{\sim}{\omega}$, which has n integral components, are used to store the history of previous iterations that is needed to meet out requirements of linear independence. To be precise, we suppose that we are about to commence the k^{th} iteration, so we have just revised the Jacobian matrix, using the correction vector $\underset{\sim}{\delta}^{(k-1)}$. The purpose of $\underset{\sim}{\omega}$ is to provide the answer to the question: "for $j=1,2,\ldots,n$, what is the least integer $i(j)$ such that the $i(j)$ most recent correction vectors, $\underset{\sim}{\delta}^{(k-1)}, \underset{\sim}{\delta}^{(k-2)}, \ldots, \underset{\sim}{\delta}^{(k-i(j))}$, span j dimensions in the space of the variables", the answer to the question being the identity

$$i(j) = \omega_{n+1-j}, \quad j=1,2,\ldots,n. \tag{51}$$

Thus the case $j=1$ shows that we always have $\omega_n=1$, and further consideration of the question implies the ordering $\omega_1 > \omega_2 > \cdots > \omega_n$. Another illustration of the role of $\underset{\sim}{\omega}$ is that if $\omega_1 = 20$, say, we know that the full space of the variables is spanned by $\underset{\sim}{\delta}^{(k-20)}, \underset{\sim}{\delta}^{(k-19)}, \ldots, \underset{\sim}{\delta}^{(k-1)}$, while the directions $\underset{\sim}{\delta}^{(k-19)}, \underset{\sim}{\delta}^{(k-18)}, \ldots, \underset{\sim}{\delta}^{(k-1)}$ are "linearly dependent".

The columns of Ω are n orthonormal vectors $\underset{\sim}{d}_1, \underset{\sim}{d}_2, \ldots, \underset{\sim}{d}_n$, calculated so that, for $j=1,2,\ldots,n$, the vectors $\underset{\sim}{d}_{n-j+1}, \underset{\sim}{d}_{n-j+2}, \ldots, \underset{\sim}{d}_n$ are a basis of the j-dimensional space containing the $i(j)$ correction vectors $\underset{\sim}{\delta}^{(k-1)}, \underset{\sim}{\delta}^{(k-2)}, \ldots, \underset{\sim}{\delta}^{(k-i(j))}$. Therefore, for instance, $\underset{\sim}{d}_n$ is defined by the equation

$$\underset{\sim}{d}_n = \pm \underset{\sim}{\delta}^{(k-1)} / \|\underset{\sim}{\delta}^{(k-1)}\|_2 \, . \tag{52}$$

It should be clear that the information in $\underset{\sim}{\omega}$ and Ω is sufficient to discover whether the vector $\underset{\sim}{\delta}^{(k)}$, generated by the method of Section 4, is "sufficiently independent".

Because our purpose is to span the full space of the variables by sequences of (2n+1) directions, there is no need to modify $\underset{\sim}{\delta}^{(k)}$ if the previous (2n-1) vectors, $\underset{\sim}{\delta}^{(k-2n+1)}, \underset{\sim}{\delta}^{(k-2n+2)}, \dots, \underset{\sim}{\delta}^{(k-1)}$ already span the space. Therefore instruction 158 of the Fortran listing tests the inequality

$$\omega_1 \geq 2n, \tag{53}$$

and we consider changing the direction $\underset{\sim}{\delta}^{(k)}$, generated by the method of Section 4, only if the inequality holds.

Even if the inequality (53) holds, we do not change $\underset{\sim}{\delta}^{(k)}$ if it is already independent of the directions $(\underset{\sim}{d}_2, \underset{\sim}{d}_3, \dots, \underset{\sim}{d}_n)$. To test whether this is the case, in accordance with our definition of "linear independence", instruction 159 tries the condition

$$|(\underset{\sim}{\delta}^{(k)}, \underset{\sim}{d}_1)| < \tfrac{1}{2} \|\underset{\sim}{\delta}^{(k)}\|_2, \tag{54}$$

where the left hand side is the modulus of a Euclidean scalar product. If the inequality (54) fails, then $\underset{\sim}{\delta}^{(k)}$ includes a substantial component of $\underset{\sim}{d}_1$, so we leave $\underset{\sim}{\delta}^{(k)}$ unchanged. Otherwise, if both the conditions (53) and (54) are satisfied, and if $\underset{\sim}{\delta}^{(k)} \neq \underset{\sim}{y}$, then it is necessary to replace $\underset{\sim}{\delta}^{(k)}$ to maintain sufficient independence in the successive vectors that are used in the updating of the Jacobian matrix.

The replacement of $\underset{\sim}{\delta}^{(k)}$ is made by instructions 161-174 of the subroutine, which set

$$\underset{\sim}{\delta}^{(k)} = \text{DSTEP}\ \underset{\sim}{d}_1 \quad ; \tag{55}$$

we choose a multiple of $\underset{\sim}{d}_1$, because Ω is constructed in such a way that the directions used to update the Jacobian during the last $(\omega_1 - 1)$ iterations are all practically orthogonal to $\underset{\sim}{d}_1$. These instructions also prepare the elements of Ω and $\underset{\sim}{\omega}$ for the next iteration, changing the columns of Ω to $\underset{\sim}{d}_2, \underset{\sim}{d}_3, \dots, \underset{\sim}{d}_n, \underset{\sim}{d}_1$, and the elements of $\underset{\sim}{\omega}$ to

$$\left.\begin{aligned} \omega_i &= 1 + \omega_{i+1}, \quad i=1,2,\dots,n-1 \\ \omega_n &= 1 \end{aligned}\right\} . \tag{56}$$

In the usual case when $\underset{\sim}{\delta}^{(k)}$ is not the special step (55), and when the correction $\underset{\sim}{\delta}^{(k)}$ is not so small that

$$||\underset{\sim}{\delta}^{(k)}||_2 < \text{DSTEP} \tag{57}$$

(this case is treated in Section 8), the updating of Ω and of $\underset{\sim}{\omega}$ is less easy. We let the required new orthonormal directions be $\underset{\sim}{d}_1^*, \underset{\sim}{d}_2^*, \dots, \underset{\sim}{d}_n^*$, and the new positive integers be $\omega_1^*, \omega_2^*, \dots, \omega_n^*$, and they must be calculated to provide the correct information about the linear independence for the next iteration. Therefore, for example, we must set

$$\left.\begin{aligned} \underset{\sim}{d}_n^* &= \pm\, \underset{\sim}{\delta}^{(k)}/||\underset{\sim}{\delta}^{(k)}||_2 \\ \omega_n^* &= 1 \end{aligned}\right\} . \tag{58}$$

Further, from the definition of Ω and of $\underset{\sim}{\omega}$, we find that, if $\underset{\sim}{\delta}^{(k)}$ is "independent" of $\underset{\sim}{d}_n$, we must obtain the results

$$\left.\begin{aligned} \underset{\sim}{d}_{n-1}^* &= \beta_1\, \underset{\sim}{d}_n + \beta_2\, \underset{\sim}{\delta}^{(k)} \\ \omega_{n-1}^* &= \omega_n + 1 \end{aligned}\right\} , \tag{59}$$

where the parameters β_1 and β_2 are calculated so that $\underset{\sim}{d}_{n-1}^*$ is normalised and is orthogonal to $\underset{\sim}{d}_n^*$.

Continuing inductively we see that, if $\underset{\sim}{\delta}^{(k)}$ is independent of the vectors $\underset{\sim}{d}_{j+1}, \underset{\sim}{d}_{j+2}, \dots, \underset{\sim}{d}_n$, then $\underset{\sim}{d}_j^*$ must be the linear combination of $\underset{\sim}{d}_{j+1}, \underset{\sim}{d}_{j+2}, \dots, \underset{\sim}{d}_n$ and $\underset{\sim}{\delta}^{(k)}$ that is calculated to make the new directions orthonormal; also ω_j^* must be set to the value

$$\omega_j^* = \omega_{j+1} + 1 . \tag{60}$$

However if $\underset{\sim}{\delta}^{(k)}$ is dependent on the vectors $\underset{\sim}{d}_{j+1}, \underset{\sim}{d}_{j+2}, \dots, \underset{\sim}{d}_n$, then $\underset{\sim}{d}_j^*$ includes a component of $\underset{\sim}{d}_j$ with some linear combination of the vectors $\underset{\sim}{d}_{j+1}, \underset{\sim}{d}_{j+2}, \dots, \underset{\sim}{d}_n$ and $\underset{\sim}{\delta}^{(k)}$, and again $\underset{\sim}{d}_j^*$ is calculated to be normalised, and to be orthogonal to $\underset{\sim}{d}_{j+1}^*$, $\underset{\sim}{d}_{j+2}^*, \dots, \underset{\sim}{d}_n^*$. In this case it is necessary to set

$$\omega_j^* = \omega_j + 1. \tag{61}$$

Fortunately, because the vectors $\underset{\sim}{d}_1, \underset{\sim}{d}_2, \ldots, \underset{\sim}{d}_n$ are orthonormal, this updating of Ω and $\underset{\sim}{\omega}$ requires only of order n^2 computer operations, for we can use the idea described by Powell (1968a). We now give the details of the updating process.

It is carried out by instructions 176-228, and first we express $\underset{\sim}{\delta}^{(k)}$ in terms of the old directions

$$\underset{\sim}{\delta}^{(k)} = \sum_{i=1}^{n} \alpha_i \underset{\sim}{d}_i , \tag{62}$$

the multipliers α_i being calculated by instruction 183 from the scalar products

$$\alpha_i = (\underset{\sim}{\delta}^{(k)}, \underset{\sim}{d}_i), \quad i=1,2,\ldots,n . \tag{63}$$

According to our definition of linear independence, $\underset{\sim}{\delta}^{(k)}$ is independent of the directions $\underset{\sim}{d}_{j+1}, \underset{\sim}{d}_{j+2}, \ldots, \underset{\sim}{d}_n$ if and only if the inequality

$$\sum_{i=1}^{j} \alpha_i^2 \geq \tfrac{1}{4} \, ||\underset{\sim}{\delta}^{(k)}||_2^2 \tag{64}$$

is satisfied. Therefore instruction 190 calculates the least value of j, m say, such that the inequality (64) is obtained, and, in accordance with equations (60) and (61), instructions 186 and 194 define

$$\left.\begin{array}{ll} \omega_j^* = \omega_{j+1} + 1, & j = m, m+1, \ldots, n-1 \\ \omega_j^* = \omega_j + 1, & j = 1,2,\ldots,m-1 \end{array}\right\} . \tag{65}$$

Note that the definition of m ensures that $\underset{\sim}{\delta}^{(k)}$ contains a non-zero component of $\underset{\sim}{d}_m$. Therefore, for $j \leq m$, we can let $\underset{\sim}{d}_j^*$ be the linear combination of the (n-j+1) vectors $\underset{\sim}{d}_j, \underset{\sim}{d}_{j+1}, \ldots, \underset{\sim}{d}_{m-1}, \underset{\sim}{d}_{m+1}, \underset{\sim}{d}_{m+2}, \ldots, \underset{\sim}{d}_n$ and $\underset{\sim}{\delta}^{(k)}$, which is normalised and which is orthogonal to $\underset{\sim}{d}_{j+1}^*, \underset{\sim}{d}_{j+2}^*, \ldots, \underset{\sim}{d}_n^*$. This is how we take up the freedom (noted in the text between equations (60) and (61)), due to the fact that, for $j \leq m$, $\underset{\sim}{d}_j^*$ is some linear combination of (n-j+2) vectors, which has to satisfy only (n-j+1) conditions.

It is convenient at this stage to eliminate the dependence on m by changing the order of the columns of Ω to $\underset{\sim}{d}_m, \underset{\sim}{d}_1, \underset{\sim}{d}_2, \ldots, \underset{\sim}{d}_{m-1}, \underset{\sim}{d}_{m+1}, \ldots, \underset{\sim}{d}_n$. This operation is carried out by instructions 197-207, and the corresponding scalar products (63) are reordered to conform. We now call the reordered directions $\underset{\sim}{d}_1, \underset{\sim}{d}_2, \ldots, \underset{\sim}{d}_n$, and we note that, using the new nomenclature, we have to calculate $\underset{\sim}{d}_j^*$ (j=n, n-1,...,1) to be the linear combination of $\underset{\sim}{\delta}^{(k)}$, $\underset{\sim}{d}_n, \underset{\sim}{d}_{n-1}, \ldots, \underset{\sim}{d}_{j+1}$ that makes the new matrix Ω orthonormal.

The components of the required new directions are calculated by instructions 208-228. First instruction 209 sets a working space vector, $\underset{\sim}{\sigma}$ say, to zero, and then a number, s say, is set to the value α_1^2 by instruction 211. These quantities are used in a "do loop" to obtain the vectors $\underset{\sim}{d}_1^*, \underset{\sim}{d}_2^*, \ldots, \underset{\sim}{d}_{n-1}^*$. Specifically, for i=2,3,...,n, we apply the operations

$$\left.\begin{aligned} &\underset{\sim}{\sigma} = \underset{\sim}{\sigma} + \alpha_{i-1}\, \underset{\sim}{d}_{i-1} \\ &\underset{\sim}{d}_{i-1}^* = (s\, \underset{\sim}{d}_i - \alpha_i\, \underset{\sim}{\sigma}) \Big/ \sqrt{s(s+\alpha_i^2)} \\ &s = s + \alpha_i^2 \end{aligned}\right] . \tag{66}$$

Finally $\underset{\sim}{d}_n^*$ is obtained directly from equation (58). It is straightforward to show that this process generates the required elements of Ω (Powell, 1968a), and also that the process is stable against the effects of computer rounding errors. The description of the method for maintaining linear independence is now complete.

8. Other details of the algorithm

In order to start the iteration, outlined in Figure 1, we need values for the quantities listed in the second paragraph of Section 3, namely (i) $\underset{\sim}{x}$, an estimate of the solution of the equations, and the corresponding function values $f_k(\underset{\sim}{x})$ (k=1,2,...,n), (ii) the approximation J, of the Jacobian, (iii) the matrix J^{-1}, (iv) the elements of Ω and $\underset{\sim}{\omega}$, and (v) the step-length Δ. The initial value of $\underset{\sim}{x}$ is specified by the user of the subroutine, and the corresponding function values $f_k(\underset{\sim}{x})$ are obtained by the initial call of "CALFUN".

The initial elements of J are equal to finite differences, like expression (12). Specifically instructions 70-80 of Appendix A evaluate the numbers

$$J_{ij} = \frac{f_i(\underset{\sim}{x} + \mathrm{DSTEP}\ \underset{\sim}{e}_j) - f_i(\underset{\sim}{x})}{\mathrm{DSTEP}}, \quad i,j=1,2,\ldots,n, \tag{67}$$

where $\underset{\sim}{e}_j$ is the normalised j^{th} coordinate vector. Later in this section we will find that these finite differences are also calculated at another stage of the subroutine.

Instructions 81-92 calculate the initial elements of J^{-1}, by calling the library subroutine that inverts the matrix J.

For definiteness and simplicity, initially Ω is set to the unit matrix (by instructions 87 and 89), and instruction 90 specifies the values

$$\omega_i = n+1-i, \quad i=1,2,\ldots,n. \tag{68}$$

Consequently on the first iteration the method of Section 7 is applied, supposing that already n iterations have been carried out, and that the coordinate directions were used in the updating of the Jacobian matrix. This supposition governs the test which decides whether it is necessary to revise the values of $\underset{\sim}{\delta}$ generated by the early iterations, in order to obtain "sufficient independence". Indeed, because of the condition (53) and the choice (68), the special formula (55) is not applied during the first n iterations, and it is not needed if the value of DSTEP accords with the advice of Section 3, for then the choice (67) will be good. Numerical examples confirm that this initial assignment of numbers to the elements of Ω and $\underset{\sim}{\omega}$ is adequate.

The initial value of Δ is calculated during the first iteration, and it is set to the quantity $\mu||\underset{\sim}{g}||$, which is defined by expressions (25) and (27), except that we demand the inequality DSTEP $\leqslant \Delta \leqslant$ DMAX. In the subroutine it is more convenient to work with Δ^2, so instruction 141 sets the variable DD to the square of

$$\Delta = \max\,(\text{DSTEP}, \min[\text{DMAX}, \mu||\underset{\sim}{g}||]). \tag{69}$$

We decided on this value by considering its effect on the first iteration. It influences the calculated correction vector $\underset{\sim}{\delta}$, but, because of the method of Section 4, the range of all possible values of $\underset{\sim}{\delta}$ is very limited. Among these values, the basic ones are the full Newton-Raphson correction $\underset{\sim}{\nu}$, and the best predicted displacement along the steepest descent vector of $F(\underset{\sim}{x})$. However if $\underset{\sim}{x}$ happens to be such that J is nearly singular, then usually $||\underset{\sim}{\nu}||$ is unacceptably large, so it seems adequate to let the first iteration calculate $\underset{\sim}{\delta}=\mu\underset{\sim}{g}$. Therefore, remembering the inequality DSTEP $\leqslant \Delta \leqslant$ DMAX, the choice (69) is appropriate.

Sections 3,4,5,6 and 7 cover most of the points of the iterative process that require explanation. For instance, among the unexplained points, the printing of function values (instructions 59-63), and interchanging $\underset{\sim}{x}$ with $\underset{\sim}{x}+\underset{\sim}{\delta}$ if $F(\underset{\sim}{x}+\underset{\sim}{\delta})$ is less than $F(\underset{\sim}{x})$ (instructions 257-267) are straightforward. However we will discuss in the remainder of this section some of the Fortran instructions connecting the separate parts of the subroutine, the case when $||\underset{\sim}{\delta}|| <$ DSTEP (see expression (57)), and the conditions for finishing the execution of the subroutine.

Among the instructions connecting the different parts of the subroutine, the most important is the one numbered 64 (we are still referring to the numbers in the extreme left hand column of the Fortran listing). It is reached after every call of CALFUN, unless a condition for returning to the calling programme is recognised first. We see that it switches the flow of the subroutine to one of five separate points, depending on the value of the integer IS. We now distinguish the five possible values of IS.

IS is equal to five only for the first call of CALFUN. In this case instruction 64 switches to the block of orders that calculates the initial Jacobian approximation (67).

To apply formula (67), n separate calls of CALFUN are needed. During this operation IS is equal to three.

The other values of IS are appropriate to the calculations of $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$ that are specified in two of the boxes of Figure 1. The value IS=2 is reserved for the case when $\underset{\sim}{\delta}$ is set to the special value (55), when the results of CALFUN are used just to update J. Therefore, if IS=2, instruction 64 branches directly to the part of the programme that revises the Jacobian approximation.

Alternatively, when $\underset{\sim}{\delta}$ is defined by the method of Section 4, IS is set to either one or four, the value IS=4 being reserved for the case when $||\underset{\sim}{\delta}|| <$ DSTEP, which we discuss below. For IS=1 we branch to the instructions that revise Δ (see Section 5), and for IS=4, after branching to the part of the programme that interchanges $\underset{\sim}{x}$ with $\underset{\sim}{x}+\underset{\sim}{\delta}$ if $F(\underset{\sim}{x}+\underset{\sim}{\delta}) < F(\underset{\sim}{x})$, instructions 257 and 268 lead to the orders that change the value of $\underset{\sim}{\delta}$ to expression (55).

We take special action in the case $||\underset{\sim}{\delta}|| <$ DSTEP, which is recognised by instruction 124, because we have decided that, due to rounding errors, it is unwise to use such a small displacement to update J. However we do revise the Jacobian on every iteration, so, for this revision we have to assign a special value to the vector $\underset{\sim}{\delta}$. The information in Ω and in $\underset{\sim}{\omega}$ suggest that the choice (55) is particularly suitable, and this is the value of $\underset{\sim}{\delta}$ that is selected, by the process described in the previous paragraph.

Clearly the revision of Ω and $\underset{\sim}{\omega}$ also requires special treatment in the case $||\underset{\sim}{\delta}|| <$ DSTEP, because the elements of Ω and $\underset{\sim}{\omega}$ concern the directions that are used to update the Jacobian matrix. Therefore, after instruction 124 has found $||\underset{\sim}{\delta}|| <$ DSTEP, instruction 126 branches past the part of the programme that applies the method of Section 7. However, later in the iteration when $\underset{\sim}{\delta}$ is changed to the value (55), we alter the columns of Ω to $\underset{\sim}{d}_2, \underset{\sim}{d}_3, \ldots, \underset{\sim}{d}_n, \underset{\sim}{d}_1$, and the elements of $\underset{\sim}{\omega}$ to expression (56), in preparation for the next iteration.

The behaviour of the algorithm when $||\underset{\sim}{\delta}|| \leqslant$ DSTEP is such that it is very important to choose a sufficiently small value of DSTEP, for otherwise the subroutine may fail to calculate the required solution of the equations. The reason comes from the observation that our requirement $\Delta \geqslant$ DSTEP and the inequality (26) imply that, if the length of $\underset{\sim}{\nu}$ does not exceed DSTEP, then the method of Section 4 invariably sets $\underset{\sim}{\delta}=\underset{\sim}{\nu}$. In other words we always try and follow the unmodified Newton-Raphson iteration, defined by equation (3), unless $||\underset{\sim}{\delta}|| >$ DSTEP. But we stated in Section 1 that the classical iteration (3) is liable to diverge unless $\underset{\sim}{x}$ is sufficiently close to a solution of the equations, so the user of the subroutine must ensure that DSTEP is set to such a small value that, if the solution of the equation (3) satisfies $||\underset{\sim}{\delta}|| \leqslant$ DSTEP, then we will obtain $F(\underset{\sim}{x}+\underset{\sim}{\delta}) < F(\underset{\sim}{x})$. Otherwise our subroutine may never replace $\underset{\sim}{x}$ by $\underset{\sim}{x}+\underset{\sim}{\delta}$.

This remark is illustrated well by Rosenbrock's (1960) equations

$$\left.\begin{aligned} f_1 &\equiv 10(x_2-x_1^2) = 0 \\ f_2 &\equiv 1 - x_1 = 0 \end{aligned}\right\} . \qquad (70)$$

Suppose that we have reached a value of (x_1,x_2) satisfying $x_2=x_1^2$, and it happens that the Jacobian approximation is exact. Then we have $F(\underset{\sim}{x}) = (1-x_1)^2$, and it is straightforward to work out that, if $\underset{\sim}{\delta}$ is the solution of equation (3), then $F(\underset{\sim}{x}+\underset{\sim}{\delta}) = 100(1-x_1)^4$. In other words $F(\underset{\sim}{x}+\underset{\sim}{\delta}) < F(\underset{\sim}{x})$ only if $0.9 < x_1 < 1.1$. From this calculation we see that if $x_1 \leq 0.9$, and if $||\underset{\sim}{\nu}|| \leq$ DSTEP, there is a real danger that our subroutine persistently calculates values of $\underset{\sim}{\delta}$ satisfying $F(\underset{\sim}{x}+\underset{\sim}{\delta}) \geq F(\underset{\sim}{x})$, and so the estimate (x_1,x_2) is not improved. The difficulty is avoided if a sufficiently small value of DSTEP is chosen.

Two of the five conditions, specified in Section 2, that cause the subroutine to finish are straightforward. They are the test (made by instruction 29) to find out whether the accuracy (14) is obtained, and the test on the total number of calls of CALFUN (made by instruction 55). Moreover the test to discover whether a sequence of (n+4) iterations fails to decrease the sum of squares of residuals is complicated only by the fact that certain iterations may not be members of the sequence, namely those for which $\underset{\sim}{\delta}$ is the special step (55), and those for which $||\underset{\sim}{\delta}|| >$ DSTEP. It is applied by instructions 38-42.

We now describe the fourth condition for returning from the subroutine. Instruction 109 tests the inequality

$$F(\underset{\sim}{x}) > 2.\text{DMAX}.||\underset{\sim}{g}||_2 \ , \qquad (71)$$

where $\underset{\sim}{g}$, defined by equation (26), is equal to the predicted gradient of $F(\underset{\sim}{x})$ multiplied by $-\frac{1}{2}$. If the inequality holds, we may leave the subroutine, because of the danger of converging to a minimum of $F(\underset{\sim}{x})$ that is not a solution to the equations. We chose this test because it suggests that there is no solution to the equations within distance DMAX of $\underset{\sim}{x}$ (see Section 2 for the definition of DMAX), for, if there was such a solution, then the mean gradient of $F(\underset{\sim}{x})$ along the straight line joining $\underset{\sim}{x}$ to the solution would exceed $||\underset{\sim}{g}||_2$, which is unlikely because (i) the true gradient of $F(\underset{\sim}{x})$ is equal to zero at any solution of the equations, and (ii) the number $||\underset{\sim}{g}||_2$ is the greatest predicted gradient of $F(\underset{\sim}{x})$ along any line in the space of the variables. However the test will hold near any stationary point of the sum of squares of residuals, which is the reason for the wording of the diagnostic printing that is given.

We do not necessarily leave the subroutine if the inequality (71) holds, because the test may be satisfied only because the Jacobian approximation is wrong. Therefore usually when a local minimum is suspected, the elements of J are re-calculated using the finite difference formulae (67), after which a new iteration is begun. J is not recalculated only if the formulae (67) were applied during the next previous iteration, in which case the condition (71) causes the subroutine to finish.

The last condition for returning to the calling programme is when an iteration uses a completely new Jacobian approximation (67), when $||\underset{\sim}{\delta}|| \leq$ DSTEP, and nevertheless the iteration does not obtain the reduction $F(\underset{\sim}{x}+\underset{\sim}{\delta}) < F(\underset{\sim}{x})$. It is identified by the first branch of instruction 42, for NTEST is set to zero by instruction 114. We prefer to leave the subroutine in this case because, if DSTEP is (as it should be) so small that the functions $f_k(\underset{\sim}{x})$ $(k=1,2,\ldots,n)$ are practically linear over neighbourhoods of width DSTEP, then the failure to attain $F(\underset{\sim}{x}+\underset{\sim}{\delta}) < F(\underset{\sim}{x})$ is probably due to rounding errors (or programming mistakes) being significant.

9. Numerical examples

The examples of this section are intended to illustrate typical behaviour of the method, and for comparison with existing algorithms, They were all worked out by an I.B.M. 360/65 computer in single precision arithmetic.

We begin with the well known problem (Rosenbrock, 1960): calculate (x_1,x_2) to solve the equations (70), given the starting approximation $(-1.2,1.0)$. We chose the parameter values DSTEP = 0.01, DMAX = 10 and ACC = 0.000001, and found that 28 calls of CALFUN were required by the subroutine. The values of (x_1,x_2,f_1,f_2,F) for every call of CALFUN are given in Table 1.

The asterisks in column 1 of the table indicate the calls of CALFUN that were made just for the revision of the Jacobian approximation. The second and third calls provide the initial approximation, the eighth, twelfth, eighteenth and twenty second calls ensure "sufficient independence", and the twenty seventh call was made in accordance with the discussion of Section 8 on the case $||\underset{\sim}{\delta}|| <$ DSTEP.

Table 1 also provides a good illustration of the method for changing the step length Δ. Initially the value of Δ is 0.1727, and at the ninth call of CALFUN it has increased to 0.3568. However the tenth evaluation of $F(x_1,x_2)$ shows that Δ needs to be halved, and in fact Δ is halved again after the fourteenth call of CALFUN to the value 0.1078 (this amount is greater than one quarter of 0.3568, because Δ is increased after the thirteenth call of CALFUN). The value Δ = 0.1078 is

TABLE 1

Rosenbrock's example

	x_1	x_2	f_1	f_2	F
1	-1.2000	1.0000	-4.4000	2.2000	24.1999
2(*)	-1.1900	1.0000	-4.1610	2.1900	22.1099
3(*)	-1.2000	1.0100	-4.3000	2.2000	23.3299
4	-1.0402	1.0655	-0.1663	2.0402	4.1903
5	-0.9645	0.9103	-0.1985	1.9645	3.8985
6	-0.8390	0.6832	-0.2078	1.8390	3.4252
7	-0.7036	0.4618	-0.3322	1.7036	3.0126
8(*)	-0.6951	0.4671	-0.1607	1.6951	2.8991
9	-0.4893	0.1765	-0.6288	1.4893	2.6135
10	-0.2320	-0.0706	-1.2447	1.2320	3.0671
11	-0.3278	0.1007	-0.0671	1.3278	1.7676
12(*)	-0.3236	0.1098	0.0511	1.3236	1.7544
13	-0.1825	-0.0028	-0.3612	1.1825	1.5288
14	0.0216	-0.0724	-0.7289	0.9784	1.4884
15	0.1204	-0.0291	-0.4365	0.8796	0.9641
16	0.2123	0.0273	-0.1777	0.7877	0.6520
17	0.3433	0.0888	-0.2909	0.6567	0.5159
18(*)	0.3476	0.0797	-0.4108	0.6524	0.5945
19	0.4867	0.1942	-0.4270	0.5133	0.4459
20	0.5985	0.3326	-0.2569	0.4015	0.2272
21	0.7128	0.4822	-0.2587	0.2872	0.1494
22(*)	0.7208	0.4762	-0.4334	0.2792	0.2658
23	0.8372	0.6691	-0.3182	0.1628	0.1277
24	0.9431	0.8670	-0.2248	0.0569	0.0538
25	1.0000	0.9937	-0.0633	0.0000	0.0040
26	1.0000	1.0014	0.0144	0.0000	0.0002
27(*)	1.0091	0.9973	-0.2098	-0.0091	0.0441
28	1.0000	1.0000	0.0003	0.0000	0.0000

increased progressively by the subsequent iterations, until, for the twenty fifth call of CALFUN, $\underset{\sim}{\delta}$ is equal to the full Newton-Raphson step (24).

The fact that our programme requires 27 evaluations of (f_1,f_2) to solve the equations (70) compares favourably with other methods: Powell's (1965) method requires 70 evaluations, Broyden's (1965) methods require 59 and 39 evaluations, while Fletcher's (1968) method calculates (f_1,f_2) between 30 and 90 times, depending on the value of a parameter.

However comparisons with other methods are more interesting when there are more than two equations, so we have tried the subroutine on the system (Fletcher and Powell, 1963)

$$\sum_{j=1}^{n} A_{ij} \sin x_j + B_{ij} \cos x_j = E_i, \quad i=1,2,\ldots,n. \tag{72}$$

The elements A_{ij} and B_{ij} are uncorrelated random integers between -100 and +100, and the numbers E_i are calculated to accord with a particular solution $(x_1^*,x_2^*,\ldots,x_n^*)$, where each component x_i^* is selected randomly from $(-\pi,\pi)$. The initial estimate of $(x_1,x_2,\ldots,x_n)$ is $\underset{\sim}{x}^* + 0.1\,\underset{\sim}{\eta}$, where, for $i=1,2,\ldots,n$, η_i is another random number from $(-\pi,\pi)$. We chose the subroutine parameters DSTEP = 0.001, DMAX = 2 and ACC = 0.001, and applied our programme for n=5,10,20 and 30. For each value of n, two systems of equations (72) were solved, the different systems being generated by different random numbers. The number of calls of CALFUN that were required are given in the last column of Table 2.

Also in Table 2 we quote the number of calls of CALFUN that are required by some other methods on the same test problem. The method due to Powell (1965) is designed for non-linear least squares calculations, but it has been used very successfully on systems of equations, while Rosen's (1966) figures were obtained using a hybrid procedure, derived from the methods of Barnes (1965) and Broyden (1965). Because Rosen reports that his figures are superior to those he obtained using Barnes's and Broyden's methods separately, the new algorithm seems to compare very favourably with four other techniques.

We also compare our method with Fletcher's (1968) recent algorithm, but unfortunately he does not use the test problem (72). Instead he prefers the "Chebyquad" equations, defined in Fletcher (1965), which determine the abscissae of the Chebyshev quadrature formulae (see Hildebrand, 1956, for instance). To apply our

TABLE 2

Number of calls of CALFUN to solve the system (72)

n	Powell's method	Rosen's figures	This method
5	24	44	11
5	24	24	12
5		24	
5		25	
5		31	
10	38	45	19
10	34	48	23
10		68	
10		46	
10		39	
20	46	93	36
20	65	86	36
20		102	
20		106	
20		86	
30	75	135	47
30	61		43

programme to this problem we chose the parameters DSTEP = 0.0001, DMAX = 0.5 and ACC = 0.00000001. The consequent number of calls of CALFUN is given in Table 3, and this table also displays the number of function evaluations needed by Fletcher's algorithm.

TABLE 3

Number of calls of CALFUN to solve Chebyquad

n	Fletcher 4 decimals	Fletcher 6 decimals	This method
2	15	19	7
4	40	40	14
6	73	92	34
8	340	838	204
9	174	181	46

The two columns of figures quoted for Fletcher's method are the number of function evaluations needed to obtain the components of $\underset{\sim}{x}$ to four and to six decimals accuracy; our choice of ACC yields about four decimals accuracy. The case n=8 is special because there is no eight point Chebyshev quadrature formula. Therefore the corresponding system of non-linear equations has no solution, and so our procedure terminated with the error message "Error return from NS01A because a nearby stationary point of $F(\underset{\sim}{x})$ is predicted". Fletcher's algorithm also identifies the lack of a solution, but it is more sophisticated in this case, for it calculates the value of $\underset{\sim}{x}$ that minimizes $F(\underset{\sim}{x})$, which accounts for the large number of function evaluations needed to improve the accuracy of $\underset{\sim}{x}$ from four to six decimals.

The example given in the appendix illustrates the steps of the iterative process when a local minimum of $F(\underset{\sim}{x})$ is found. It is the system of equations (75), due to Freudenstein and Roth (1963). We note that only three iterations are needed to reduce $F(\underset{\sim}{x})$ from 1256.0 to 54.15 (the value at the local minimum is 48.98), the length of the step of the third iteration being $||\underset{\sim}{\delta}|| = 1.1771$. However the sixth to ninth calls of CALFUN each cause Δ to be halved, and so the step-length is reduced to 0.0736. Moreover, because it happens that $F(\underset{\sim}{x}+\underset{\sim}{\delta}) > F(\underset{\sim}{x})$ for the fourth, fifth, sixth and seventh iterations, J is revised substantially. Consequently a nearby stationary point is predicted after the tenth call of CALFUN, and so the eleventh and twelfth calls of CALFUN calculate function values to derive a new Jacobian approximation, in accordance with the difference formula (67). Using this new Jacobian, the test (71) is not satisfied, so the thirteenth call of CALFUN is made, and it leads to a successful reduction in $F(\underset{\sim}{x})$. However, for the new value of $\underset{\sim}{x}$, a nearby stationary point is again predicted, and it is confirmed by the matrix J resulting from the fourteenth and fifteenth calls of CALFUN. Therefore there is an error return from the subroutine.

Our final numerical example shows a badly scaled and difficult problem. We have stated that, because we measure step-lengths in a Euclidean metric, it is preferable to scale the separate variables $x_1,x_2,\dots,x_n$ so that their magnitudes are comparable, but it is interesting to discover what happens when this advice is not followed. Therefore we applied the subroutine to the system

$$\left.\begin{array}{l} 10000\, x_1\, x_2 = 1 \\ e^{-x_1} + e^{-x_2} = 1.0001 \end{array}\right] , \qquad (73)$$

starting at the estimate $(x_1,x_2) = (0,1)$. The other input variables were set to the values: DSTEP = 0.001, DMAX = 20, and ACC = 10^{-10}. After 223 calls of CALFUN, the accuracy criterion (14) was satisfied, and the final values of the variables were $x_1 = 1.106 \times 10^{-5}$, $x_2 = 9.038$, although the solution is $x_1 = 1.098 \times 10^{-5}$, $x_2 = 9.106$. Some of the 223 values of $F(x_1,x_2)$ that were calculated by the subroutine are listed in Table 4.

In fact Table 4 records the number of calls of CALFUN that were required to reduce $F(x_1,x_2)$ to less than 10^k for $k=1, 0, -1,\dots,-10$. In case the figures suggest to the reader that the slow convergence is due to the limited accuracy of single length arithmetic, or to a programming mistake, or to a poor choice of the parameter DSTEP, we must explain the difficulty of the example. It is that the quadratic convergence properties of the Newton-Raphson process do not dominate until the value of $F(x_1,x_2)$ becomes extremely small.

TABLE 4

Number of calls of CALFUN to solve the equations (73)

Calls of CALFUN	$x_1 \times 10^5$	x_2	$F(x_1,x_2)$
1	0.0000	1.0000	1.1353×10^{0}
4	10.0004	1.0010	1.3492×10^{-1}
15	7.9132	1.2560	8.1043×10^{-2}
21	4.2065	2.4662	8.5846×10^{-3}
27	2.5948	3.8917	5.0809×10^{-4}
33	2.1592	4.6422	9.6142×10^{-5}
51	1.7203	5.8132	8.2463×10^{-6}
77	1.4573	6.8624	8.7854×10^{-7}
113	1.2887	7.7601	9.9477×10^{-8}
159	1.1805	8.4710	9.7010×10^{-9}
199	1.1274	8.8698	9.0726×10^{-10}
223	1.1064	9.0380	7.3881×10^{-11}

To be specific we suppose that (x_1,x_2) satisfies the first of equations (73) exactly, and that the full Newton-Raphson correction, defined by the system (3), is applied. Then (x_1,x_2) is altered to $(x_1-\lambda\, x_1,\ x_2+\lambda\, x_2)$, where λ is the expression

$$\lambda = \frac{e^{-x_1} + e^{-x_2} - 1.0001}{x_2\, e^{-x_2} - x_1\, e^{-x_1}}$$

$$\approx 1000\left(e^{-x_1} + e^{-x_2} - 1.0001\right), \qquad (74)$$

the number 1000 being the nearest integer to the value of the factor $1/(x_2 e^{-x_2} - x_1 e^{-x_1})$ at the solution to the equations. Now if we let $(e^{-x_1}+e^{-x_2}-1.0001) = \rho$, we have $F(x_1,x_2) = \rho^2$, and, assuming the approximation (73) to be exact, we find that the sum of squares of residuals after the iteration is bounded by the expression

$$\begin{aligned} F(x_1-\lambda\, x_1,\ x_2+\lambda\, x_2) &\geq \left\{10000\ (x_1-\lambda\, x_1)(x_2+\lambda\, x_2) - 1\right\}^2 \\ &= \lambda^4 \\ &= 10^{12}\ \rho^4, \end{aligned} \qquad (75)$$

because of our hypothesis that $10000\, x_1 x_2 = 1$. Therefore the iteration decreases the sum of squares of residuals only if $F(x_1,x_2) < 10^{-12}$, so we cannot expect Table 4 to exhibit the quadratic convergence properties of our algorithm. Note that this remark about the unmodified Newton-Raphson iteration applies even if the variables x_1 and x_2 are scaled so that they are of the same magnitude, because a property of this iteration is that the change in $F(x_1,x_2)$ is independent of linear transformations of the variables.

We found that, in spite of the method that we use for adjusting the step-length (see Section 5), the example (74) leads to oscillatory behaviour in the value of Δ. Specifically, after 30 calls of CALFUN, the behaviour of the algorithm repeats itself every four iterations: one iteration uses the previous value of Δ, the next iteration uses a step-length that is close to or equal to the value 2Δ, the next iteration uses half this step-length, and the fourth call of CALFUN is needed to maintain sufficient linear independence in the directions that are used to revise the Jacobian approximation. These inefficiencies can be avoided by changing the scale of the variables.

10. Conclusion

Because this report contains so much detail, we conclude by isolating the points that seem to be particularly important. The most prominent is that the numerical examples indicate that the new algorithm is more efficient that its competitors. Much of this gain is obtained by two features: (i) the iterations do not include any searches along lines in the space of the variables, so most iterations require only one call of CALFUN, and (ii) the correction vectors $\underset{\sim}{\delta}$ interpolate between the Newton-Raphson and the steepest descent corrections in a way that gives fast ultimate convergence, and that is sensible when the estimate $\underset{\sim}{x}$ is a long way from the solution.

However a deficiency of the algorithm is that the user has to make some decisions carefully. In particular the parameter DSTEP must be so small that, for $||\underset{\sim}{\delta}|| \leq$ DSTEP, $f_k(\underset{\sim}{x}+\underset{\sim}{\delta})$, $k=1,2,\ldots,n$, is nearly a linear function of $\underset{\sim}{\delta}$ (the remarks of Section 8 on this question are more explicit), but it must not be so small that the differences (67) are dominated by computer rounding errors. Also the user's scaling of the unknowns $(x_1,x_2,\ldots,x_n)$ must be such that it is sensible to apply the usual Euclidean definitions of vector scalar products and orthogonality. We admit that the assignment of suitable parameters was easy in the numerical examples of Section 9, but the success of the subroutine on the trigonometric equations (72) is encouraging, because these equations become very ill-conditioned as n increases. It would be valuable to extend the algorithm so that DSTEP and the metric of the variables are assigned automatically.

Finally we wish to state the opinion that our main strategy could provide very good algorithms for many calculations that involve searching in many variable space. This strategy is to have a step-length parameter Δ, and, on each iteration, a correction $\underset{\sim}{\delta}$ is predicted subject to $||\underset{\sim}{\delta}|| \leq \Delta$. The step-length Δ is adjusted automatically, according to the success of $\underset{\sim}{\delta}$. Thus in many calculations one can manage with only one new value of the objective functions on each iteration.

References

Barnes, J.G.P. (1965) "An algorithm for solving non-linear equations based on the secant method", Computer Journal, Vol. 8, pp.66-72.

Broyden, C.G. (1965) "A class of methods for solving non-linear simultaneous equations", Math. Comp., Vol. 19, pp.577-593.

Broyden, C.G. (1967) "Quasi-Newton methods, and their application to function minimisation", Math. Comp., Vol. 21, pp.368-381.

Fletcher, R. (1965) "Function minimization without evaluating derivatives - a review", Computer Journal, Vol. 8, pp.33-41.

Fletcher, R. (1968) "Generalised inverse methods for the best least squares solution of systems of non-linear equations", Computer Journal, Vol. 10, pp.392-399.

Fletcher, R. and Powell, M.J.D. (1963) "A rapidly convergent descent method for minimisation", Computer Journal, Vol. 6, pp.163-168.

Freudenstein, F. and Roth, B. (1963) "Numerical solutions of systems of non-linear equations" J. Assoc. Comput. Mach., Vol. 10, pp.550-556.

Haselgrove, C.B. (1961) "The solution of non-linear equations and of differential equations with two-point boundary conditions", Computer Journal, Vol. 4, pp.255-259.

Hildebrand, F.B. (1956) "Introduction to numerical analysis", McGraw-Hill (New York).

Levenberg, K. (1944) "A method for the solution of certain non-linear problems in least squares", Quart. Appl. Math., Vol. 2, pp.164-168.

Marquardt, Donald W. (1963) "An algorithm for least squares estimation of non-linear parameters", J. Soc. Ind. Appl. Math., Vol. 11, pp.431-441.

Ostrowski, A.M. (1966) "Solution of equations and systems of equations", Academic Press (New York).

Powell, M.J.D. (1965) "A method for minimizing a sum of squares of non-linear functions without calculating derivatives", Computer Journal, Vol. 7, pp.303-307.

Powell, M.J.D. (1968a) "On the calculation of orthogonal vectors", Computer Journal, Vol. 11, pp.302-304.

Powell, M.J.D. (1968b) "A theorem on rank one modifications to a matrix and its inverse", Report No. T.P. 355.

Powell, M.J.D. (1969) "A hybrid method for non-linear equations", in preparation.

Rosen, Edward, M. (1966) "A review of Quasi-Newton methods in non-linear equation solving and unconstrained optimization", Proc. 21st A.C.M. National Conf., pp.37-41.

Rosenbrock, H.H. (1960) "An automatic method for finding the greatestor the least value of a function", Computer Journal, Vol. 3, pp.175-184.