A Class of Methods for Solving Nonlinear Simultaneous Equations
Author(s): C. G. Broyden
Source: *Mathematics of Computation*, Vol. 19, No. 92 (Oct., 1965), pp. 577-593
Published by: American Mathematical Society
Stable URL: http://www.jstor.org/stable/2003941
Accessed: 16/11/2009 12:31

Your use of the JSTOR archive indicates your acceptance of JSTOR's Terms and Conditions of Use, available at http://www.jstor.org/page/info/about/policies/terms.jsp. JSTOR's Terms and Conditions of Use provides, in part, that unless you have obtained prior permission, you may not download an entire issue of a journal or multiple copies of articles, and you may use content in the JSTOR archive only for your personal, non-commercial use.

Please contact the publisher regarding any further use of this work. Publisher contact information may be obtained at http://www.jstor.org/action/showPublisher?publisherCode=ams.

Each copy of any part of a JSTOR transmission must contain the same copyright notice that appears on the screen or printed page of such transmission.

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms of scholarship. For more information about JSTOR, please contact support@jstor.org.

American Mathematical Society is collaborating with JSTOR to digitize, preserve and extend access to *Mathematics of Computation.*

http://www.jstor.org

A Class of Methods for Solving Nonlinear Simultaneous Equations

By C. G. Broyden

1. Introduction. The solution of a set of nonlinear simultaneous equations is often the final step in the solution of practical problems arising in physics and engineering. These equations can be expressed as the simultaneous zeroing of a set of functions, where the number of functions to be zeroed is equal to the number of independent variables. If the equations form a sufficiently good description of a physical or engineering system, they will have a solution that corresponds to some state of this system. Although it may be impossible to prove by mathematics alone that a solution exists, this fact may be inferred from the physical analogy. Similarly, although the solution may not be unique, it is hoped that familiarity with the physical analogue will enable a sufficiently good initial estimate to be found so that any iterative process that may be used will in fact converge to the physically significant solution.

The functions that require zeroing are real functions of real variables and it will be assumed that they are continuous and differentiable with respect to these variables. In many practical examples they are extremely complicated and hence laborious to compute, and this fact has two important immediate consequences. The first is that it is impracticable to compute any derivative that may be required by the evaluation of the algebraic expression of this derivative. If derivatives are needed they must be obtained by differencing. The second is that during any iterative solution process the bulk of the computing time will be spent in evaluating the functions. Thus, the most efficient process will tend to be that which requires the smallest number of function evaluations.

This paper discusses certain modifications to Newton's method designed to reduce the number of function evaluations required. Results of various numerical experiments are given and conditions under which the modified versions are superior to the original are tentatively suggested.

2. Newton's Method. Consider the set of n nonlinear equations

$$f_j(x_1, x_2, \cdots, x_n) = 0, \qquad j = 1, 2, \cdots, n. \tag{2.1}$$

These may be written more concisely as

$$\mathbf{f}(\mathbf{x}) = 0 \tag{2.2}$$

where $\mathbf{x}$ is the column vector of independent variables and $\mathbf{f}$ the column vector of functions f_j. If $\mathbf{x}_i$ is the ith approximation to the solution of (2.2) and $\mathbf{f}_i$ is written for $\mathbf{f}(\mathbf{x}_i)$, then Newton's method (see e.g. [4]) is defined by

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{A}_i^{-1}\mathbf{f}_i \tag{2.3}$$

where $\mathbf{A}_i$ is the Jacobian matrix $[\partial f_j/\partial x_k]$ evaluated at $\mathbf{x}_i$.

Now Newton's method, as expressed in equation (2.3), suffers from two serious

Received April 20, 1965.

disadvantages from the point of view of practical calculation. The first of these is the difficulty of computing the Jacobian matrix. Even if the functions f_j are sufficiently simple for their partial derivatives to be obtained analytically, the amount of labour required to evaluate all n^2 of these may well be excessive. In the majority of practical problems, however, the f_j's are far too complicated for this, and an approximation to the Jacobian matrix must be obtained numerically. This is often carred out directly, i.e., $\partial f_j/\partial x_k$ is computed by evaluating f_j for two different values of x_k while holding the remaining $n - 1$ independent variables constant. Although this direct approach gives an immediate approximation to the partial derivatives, it is not without its disadvantages, the most serious of which is the amount of labour involved. For in order to compute the Jacobian matrix in this way the vector function $\mathbf{f}$ must be evaluated for at least $n + 1$ sets of independent variables.

The second disadvantage of Newton's method is the fact that without some modifications it frequently fails to converge. Conditions for convergence are in fact well known (see e.g. [4]), but they rely on the initial estimate of the solution being sufficiently good, a requirement that is often impossible to realise in practice. Hence Newton's method, as defined by (2.3), cannot be regarded as a good practical algorithm.

Despite these disadvantages, however, the method has much to commend it. The algorithm is simple, even though it does require the solution of n linear equations at every step, it has a sound theoretical basis and for many problems its convergence is rapid. Furthermore, it has few competitors. The methods based upon minimising some norm of the vector $\mathbf{f}$ by the steepest descent or some similar method tend to converge slowly if the problem is at all ill-conditioned, although the very new method described by Kizner [5] may ultimately prove to be more satisfactory.

In order to overcome some of the disadvantages of Newton's method it has been suggested that the Jacobian matrix be evaluated either once and for all or once every few iterations, instead of at every iteration as is strictly required. Both these variants, however, require the complete evaluation of the Jacobian matrix. In the class of methods described in this paper the partial derivatives $\partial f_j/\partial x_k$ are not estimated or evaluated directly, but corrections to the approximate inverse of the Jacobian matrix are computed from values of the vector function $\mathbf{f}$.

A correction of this type is carried out at each iteration and is much simpler to perform than the evaluation of the complete Jacobian. It is hoped, therefore, that this class of methods combines to some extent the simplicity of the constant matrix (once for all) method with the ultimately rapid convergence of the basic Newton method.

3. The Class of Methods. Let $\mathbf{x}_i$ be the ith approximation to the solution of (2.2) and defined $\mathbf{p}_i$ by

$$\mathbf{p}_i = -\mathbf{B}_i^{-1}\mathbf{f}_i \tag{3.1}$$

where $\mathbf{B}_i$ is some approximation to the Jacobian matrix evaluated at $\mathbf{x}_i$. Then a simple modification to Newton's algorithm gives $\mathbf{x}_{i+1}$ by

$$\mathbf{x}_{i+1} = \mathbf{x}_i + t_i\mathbf{p}_i, \tag{3.2}$$

where t_i, whose derivation will be discussed in section 5, is a scalar multiplier chosen

to prevent the process diverging. Let $\mathbf{x}$ now be defined as

$$\mathbf{x} = \mathbf{x}_i + t\mathbf{p}_i , \tag{3.3}$$

where t may take any arbitrary value. The functions f_j may now be regarded as functions of the single variable t and, since the partial derivatives $\partial f_j/\partial x_k$ are assumed to exist, possess a first derivative with respect to t. It will now be shown that an approximation to the derivatives df_j/dt, $j = 1, 2, \cdots, n$, may be used to improve the approximate Jacobian matrix $\mathbf{B}_i$.

Since $\mathbf{x}$ is now assumed to be a function of t alone,

$$\frac{df_j}{dt} = \sum_{k=1}^{n} \frac{\partial f_j}{\partial x_k} \frac{dx_k}{dt}, \qquad j = 1, 2, \cdots, n \tag{3.4}$$

and if by $d\mathbf{f}/dt$ is understood the vector $[df_j/dt]$, equation (3.4) becomes

$$\frac{d\mathbf{f}}{dt} = \mathbf{A}\mathbf{p}_i \tag{3.5}$$

where $\mathbf{A}$ is the Jacobian matrix. Hence if an accurate estimate of $d\mathbf{f}/dt$ were available, it could be used, via equation (3.5), to establish a condition that any approximation to the Jacobian matrix must satisfy. Now it has already been assumed that the functions f_j are sufficiently complicated for analytic expressions of their first partial derivatives not to be available; the only way of obtaining an estimate to $d\mathbf{f}/dt$ is by differencing. Since the aim is to obtain an approximation to the Jacobian matrix at the point $\mathbf{x}_{i+1}$, it is necessary to obtain an approximation to $d\mathbf{f}/dt$ at this point. Hence expand each f_j, still regarded as a function of t alone, as a Taylor series about t_i and assume that the second and higher-order terms are small, giving

$$\mathbf{f}(t_i - s) \cong \mathbf{f}_{i+1} - s\frac{d\mathbf{f}}{dt}. \tag{3.6}$$

Hence when $\mathbf{f}_{i+1}$ and $\mathbf{f}(t_i - s_i)$, where s_i is a particular value of s whose choice will be discussed in section 6, have been computed, equation (3.6) may be used without the need for further evaluations of the function $\mathbf{f}$ to obtain an approximation of $d\mathbf{f}/dt$ and hence to impose a condition upon any approximation to the Jacobian matrix. No special evaluation of $\mathbf{f}$ is ever necessary to do this since if s is put equal to t_i, $\mathbf{f}(t_i - s)$ becomes $\mathbf{f}_i$, a known quantity. Eliminating $d\mathbf{f}/dt$ between (3.5) and (3.6) then gives

$$\mathbf{f}_{i+1} - \mathbf{f}(t_i - s_i) \cong s_i\mathbf{A}\mathbf{p}_i . \tag{3.7}$$

Consider now equation (3.7). It relates four quantities that have already been computed to a fifth, the Jacobian matrix $\mathbf{A}$, to which only an approximation $\mathbf{B}_i$ is available and to which a better approximation $\mathbf{B}_{i+1}$ is sought. Now in the class of methods discussed in this paper $\mathbf{B}_{i+1}$ is chosen in such a way as to satisfy the equation

$$\mathbf{f}_{i+1} - \mathbf{f}(t_i - s_i) = s_i\mathbf{B}_{i+1}\mathbf{p}_i . \tag{3.8}$$

On comparing equations (3.7) and (3.8) it is seen that if the error in (3.7) is negligible, i.e., if the sum of the second and higher-order terms in the Taylor series expansion of each f_j is sufficiently small, then $\mathbf{B}_{i+1}$ will have at least one of the re-

quired properties of the Jacobian matrix. If, on the other hand, the omitted sums are large, the matrix $\mathbf{B}_{i+1}$ will not approximate to the Jacobian matrix. Hence it would appear reasonable to choose s_i to be as small as possible consistent with not introducing excessive rounding error into the estimate of $d\mathbf{f}/dt$.

If s_i is put equal to t_i, equation (3.8) becomes

$$\mathbf{f}_{i+1} - \mathbf{f}_i = t_i \mathbf{B}_{i+1} \mathbf{p}_i ; \tag{3.9}$$

and this shows that the jth element of the vector $\mathbf{B}_{i+1}\mathbf{p}_i$ assumes, from the first Mean Value Theorem, the value of df_j/dt evaluated at some point, which varies with j, between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. Although this is not as good as obtaining a set of derivatives evaluated at $\mathbf{x}_{i+1}$, it does ensure that $\mathbf{B}_{i+1}$ refers to points closer to the solution than $\mathbf{x}_i$, since $\mathbf{x}_{i+1}$ is, in some sense, a better approximation to the solution than $\mathbf{x}_i$.

Now if the calculation is carried out holding the matrix $\mathbf{B}_i$ in the store of the computer, it will be necessary to solve n linear equations to compute, using equation (3.1), the step direction vector $\mathbf{p}_i$; but if the inverse of this matrix is stored, this operation is reduced to a matrix-vector multiplication. If $\mathbf{H}_i$ and $\mathbf{y}_i$ are defined by

$$\mathbf{H}_i = -\mathbf{B}_i^{-1} \tag{3.10}$$

and

$$\mathbf{y}_i = \mathbf{f}_{i+1} - \mathbf{f}(t_i - s_i), \tag{3.11}$$

equations (3.1) and (3.8) become

$$\mathbf{p}_i = \mathbf{H}_i \mathbf{f}_i \tag{3.12}$$

and

$$\mathbf{H}_{i+1}\mathbf{y}_i = -s_i \mathbf{p}_i . \tag{3.13}$$

Equation (3.13) does not, like equation (3.8), define a unique matrix but a class of matrices. Hence equations (3.2) and (3.11)–(3.13) define a class of methods based upon that of Newton for solving a set of nonlinear simultaneous equations, and these methods will subsequently be referred to as quasi-Newton methods. By making further assumptions about the properties of $\mathbf{B}_{i+1}$ or $\mathbf{H}_{i+1}$ particular methods may be derived.

The most important feature of the quasi-Newton methods is that although the iteration matrix $\mathbf{H}_i$ changes from step to step, no evaluations of $\mathbf{f}(\mathbf{x})$ are required beyond those that would be necessary if $\mathbf{H}_i$ remained constant. This is an important consideration if the vector function $\mathbf{f}$ is laborious to compute. Another feature of these methods is that as $\mathbf{x}_i$ approaches the solution the assumptions made in deriving (3.6) become more valid. Hence if $\mathbf{B}_i$ tends to the Jacobian matrix, the rate of convergence may be expected to improve as the solution is approached, becoming ultimately quadratic.

4. Particular Methods.

Method 1. In the previous section a class of methods was derived in which the new approximation $\mathbf{B}_{i+1}$ to the Jacobian matrix is required to satisfy an equation

involving the two values of the vector function evaluated at $\mathbf{x}_{i+1}$ and $\mathbf{x}_i + (t_i - s_i)\mathbf{p}_i$, respectively. This equation from (3.8) and (3.11) is

$$\mathbf{y}_i = s_i \mathbf{B}_{i+1}\mathbf{p}_i \tag{4.1}$$

and relates the change $\mathbf{y}_i$ in the vector function to the change of $\mathbf{x}$ in the direction $\mathbf{p}_i$. No information, however, is available about the change in $\mathbf{f}$ when $\mathbf{x}$ is changed in a direction other than $\mathbf{p}_i$, hence it is not possible to obtain a fresh estimate of rates of change of $\mathbf{f}$ in these directions. So, in Method 1, $\mathbf{B}_{i+1}$ is chosen so that the change in $\mathbf{f}$ predicted by $\mathbf{B}_{i+1}$ in a direction $\mathbf{q}_i$ orthogonal to $\mathbf{p}_i$ is the same as would be predicted by $\mathbf{B}_i$. Symbolically, that is

$$\mathbf{B}_{i+1}\mathbf{q}_i = \mathbf{B}_i\mathbf{q}_i, \qquad \mathbf{q}_i{}^T\mathbf{p}_i = 0. \tag{4.2}$$

This, with (4.1), is sufficient to define $\mathbf{B}_{i+1}$ uniquely and it is readily verified that this unique value is given by

$$\mathbf{B}_{i+1} = \mathbf{B}_i + \frac{(\mathbf{y}_i - s_i\mathbf{B}_i\mathbf{p}_i)\mathbf{p}_i{}^T}{s_i\mathbf{p}_i{}^T\mathbf{p}_i}. \tag{4.3}$$

However, as it was pointed out in the previous section, it is preferable to store in the computer $\mathbf{H}_i$ as opposed to $\mathbf{B}_i$, and it is possible, using Householder's modification formula, to compute $\mathbf{H}_{i+1}$ with very little labour from $\mathbf{H}_i$. Householder's formula states that if $\mathbf{A}$ is a nonsingular matrix and $\mathbf{x}$ and $\mathbf{y}$ are vectors, all of order n, and if $(\mathbf{A} + \mathbf{x}\mathbf{y}^T)$ is nonsingular, then

$$(\mathbf{A} + \mathbf{x}\mathbf{y}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{x}\mathbf{y}^T\mathbf{A}^{-1}}{1 + \mathbf{y}^T\mathbf{A}^{-1}\mathbf{x}}. \tag{4.4}$$

Applying this to (4.3) and recalling (3.10) gives

$$\mathbf{H}_{i+1} = \mathbf{H}_i - \frac{(s_i\mathbf{p}_i + \mathbf{H}_i\mathbf{y}_i)\mathbf{p}_i{}^T\mathbf{H}_i}{\mathbf{p}_i{}^T\mathbf{H}_i\mathbf{y}_i}. \tag{4.5}$$

This equation defines the first of the methods to be discussed. Clearly $\mathbf{H}_{i+1}$ satisfies (3.13).

It will now be shown for this method that if the Jacobian matrix $\mathbf{A}$ is independent of $\mathbf{x}$, i.e., if the equations to be solved are linear, then $\mathbf{B}_{i+1}$ is not a worse approximation to $\mathbf{A}$ than $\mathbf{B}_i$ in the sense that the spectral norm of $(\mathbf{A} - \mathbf{B}_{i+1})$ is not greater than that of $(\mathbf{A} - \mathbf{B}_i)$. For if the linear equations are

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4.6}$$

it follows from (3.3) and (3.11) that

$$\mathbf{y}_i = s_i\mathbf{A}\mathbf{p}_i, \tag{4.7}$$

and substituting this in (4.3) gives

$$\mathbf{B}_{i+1} = \mathbf{B}_i - (\mathbf{B}_i - \mathbf{A})\frac{\mathbf{p}_i\mathbf{p}_i{}^T}{\mathbf{p}_i{}^T\mathbf{p}_i}. \tag{4.8}$$

If $\mathbf{C}_i$ is defined by

$$\mathbf{C}_i = \mathbf{B}_i - \mathbf{A}, \tag{4.9}$$

it follows from (4.8) that

$$\mathbf{C}_{i+1} = \mathbf{C}_i \left(\mathbf{I} - \frac{\mathbf{p}_i \mathbf{p}_i^T}{\mathbf{p}_i^T \mathbf{p}_i} \right). \tag{4.10}$$

But the matrix

$$\left(\mathbf{I} - \frac{\mathbf{p}_i \mathbf{p}_i^T}{\mathbf{p}_i^T \mathbf{p}_i} \right)$$

is symmetric and has $n - 1$ eigenvalues equal to unity and one equal to zero. Its spectral norm is thus unity and since the norm of the product of two matrices is less than or equal to the product of their norms (see e.g. [9]), then the norm of $\mathbf{C}_{i+1}$ cannot exceed that of $\mathbf{C}_i$ and the assertion is proved.

Method 2. The second quasi-Newton method is obtained by requiring that

$$\mathbf{H}_{i+1}\mathbf{v}_i = \mathbf{H}_i\mathbf{v}_i, \qquad \mathbf{v}_i^T\mathbf{y}_i = 0 \tag{4.11}$$

and is thus in some sense a complement of Method 1. Since $\mathbf{H}_{i+1}$ must also satisfy (3.13), it is readily seen that for this method $\mathbf{H}_{i+1}$ is uniquely given by

$$\mathbf{H}_{i+1} = \mathbf{H}_i - \frac{(s_i \mathbf{p}_i + \mathbf{H}_i \mathbf{y}_i)\mathbf{y}_i^T}{\mathbf{y}_i^T \mathbf{y}_i}. \tag{4.12}$$

By similar arguments used for Method 1 it can be shown that if the equations to be solved are linear, then $\mathbf{H}_{i+1}$ is not a worse approximation to $-\mathbf{A}^{-1}$ than $\mathbf{H}_i$. Since, however, this method appears in practice to be unsatisfactory, it will be discussed no further at this stage.

Method 3. The two methods described above are, in principle, suitable for solving a general set of nonlinear simultaneous equations. If now certain assumptions are made about the form of the equations, further methods may be derived. If, in particular, the functions f_j are the first partial derivatives of a convex function, then solving the equations is a way of minimising the function. Under these conditions it is known that at the solution the Jacobian matrix is both symmetric and positive definite, and $\mathbf{H}_{i+1}$ may be chosen to fulfil these requirements. In Method 3, which is, to the author's knowledge, the only one of this class of methods to have been previously published, $\mathbf{H}_{i+1}$ is defined by

$$\mathbf{H}_{i+1} = \mathbf{H}_i - \frac{\mathbf{H}_i \mathbf{y}_i \mathbf{y}_i^T \mathbf{H}_i}{\mathbf{y}_i^T \mathbf{H}_i \mathbf{y}_i} - \frac{s_i \mathbf{p}_i \mathbf{p}_i^T}{\mathbf{p}_i^T \mathbf{y}_i}. \tag{4.13}$$

Clearly, if $\mathbf{H}_i$ is symmetric, then so is $\mathbf{H}_{i+1}$ and equally (3.13) is satisfied. If s_i assumes the value t_i, the method becomes that of Davidon as described by Fletcher and Powell. Since, however, it has been fully discussed elsewhere [1, 2], it will be considered no further here beyond the comment that Powell [6] regards it not as a version of Newton's method but as a conjugate gradient method, and the discussion in [2] is along those lines.

5. The Prevention of Divergence. It was mentioned in section 2 that one of the practical problems of using Newton's method was its frequent failure to converge from a poor initial estimate of the solution. Although it is not possible to guarantee convergence, divergence may be prevented by ensuring that some norm of the vector

function $\mathbf{f}_i$ is a nonincreasing function of i. The simplest, and that used in the author's program, is the Euclidean norm although of course others may be employed.

In the practical application of one of the methods described in section 4, once a step direction $\mathbf{p}_i$ has been obtained from equation (3.12), $\mathbf{x}$ is constrained to satisfy equation (3.3) so that, until $\mathbf{x}_{i+1}$ has been determined, $\mathbf{f}(\mathbf{x})$ and its norm become functions of t alone. The value t_i of t that is finally used to determine $\mathbf{x}_{i+1}$ may then be chosen in various ways, and two of these are now considered.

In the first method t_i is chosen to minimise the norm of $\mathbf{f}_{i+1}$. This is, perhaps, the most obvious choice to make since it gives the greatest immediate reduction of the norm and hence, in some sense, the greatest improvement to the approximate solution. However, in order to do this the vector function $\mathbf{f}$ needs to be evaluated a number of times, and this means an increase in the amount of computation required compared with the alternative strategy of choosing a value of t_i which merely reduces the norm. Frequently the original Newtonian value of unity is sufficient to do this, but even if it is not, it will be less laborious to reduce the norm than to minimise it. Reducing the norm does not, of course, give as good an immediate improvement to the solution as norm minimisation, but it does mean that less work is involved in correcting the matrix $\mathbf{H}_i$.

The relative merits of these opposing strategies, that of obtaining the greatest improvement to the solution during one step or that of using the new information as soon as possible to correct $\mathbf{H}_i$ and compute a new $\mathbf{x}_{i+1}$, are difficult to assess from theory alone and recourse must be made to numerical experiment. The results of applying these two strategies to various problems are discussed in sections 9 and 10.

6. The Choice of s_i. Since $\mathbf{B}_i$ is required to approximate as closely as possible to the Jacobian matrix, it is reasonable to choose s_i so that $d\mathbf{f}/dt$ is approximated as closely as possible, and this implies that s_i should be as small as possible consistent with the difference between $\mathbf{f}(t_i - s_i)$ and $\mathbf{f}(t_i)$ being large enough for rounding error to be unimportant. However, it is also desirable for there to be no further evaluations of the vector function $\mathbf{f}$, use being made only of those values of the function computed in the course of minimising or reducing the norm. If the strategy of norm reduction is adopted and if the first value of t chosen results in the norm being reduced, then t_i is taken to be this value, and the only possible value that s_i can assume without extra function evaluation is t_i. If, however, more than one evaluation is needed to reduce the norm or if the norm minimisation method is used, then there is a certain amount of latitude about the choice of s_i within the requirement that no further evaluations are permitted. It may still be chosen to be t_i but from the point of view of minimising truncation error the smallest possible s should be chosen. This, however, is not easy to implement in practice. For assume that the norm $N(t)$ has been evaluated successively at $t = t^{(k)}$, $k = 1, 2, \cdots, m$, where

$$t^{(1)} = 1, \tag{6.1}$$

$$t^{(m)} = t_i, \tag{6.2}$$

and where t_i is either the value of t that minimises N or the first $t^{(k)}$ for which $N(t^{(k)}) < N(0)$. If $s^{(k)}$ is defined by

$$s^{(k)} = t^{(m)} - t^{(k)}, \qquad k = 1, 2, \cdots, m-1, \tag{6.3}$$

then s_i is best chosen to be that $s^{(k)}$ having the smallest modulus. It is, however, impossible to compute this until $t^{(m)}$ itself is known, and this results in the necessity of storing the m vectors $\mathbf{f}(t^{(k)})$, where m is indeterminate, in order to compute the appropriate value of $\mathbf{y}_i$.

In order to avoid this it is assumed that the value of $s^{(k)}$ most likely to have the smallest modulus is given by putting k equal to $m-1$ in equation (6.3), and so defining s_i by

$$s_i = t^{(m)} - t^{(m-1)}. \tag{6.4}$$

The corresponding value of $\mathbf{y}_i$ is then given by

$$\mathbf{y}_i = \mathbf{f}_{i+1} - \mathbf{f}(t^{(m-1)}) \tag{6.5}$$

so that only two values of $\mathbf{f}(t)$, namely $\mathbf{f}(t^{(k-1)})$ and $\mathbf{f}(t^{(k)})$, need be stored at any one time.

If s_i is put equal to t_i, the term "full step" will be used subsequently to describe the method, whereas if s_i is given by (6.4), the term "incremental" will be employed.

7. Some Details of the Test Program. In order to test the methods described above a program was prepared that would solve a given problem with given initial conditions using a variety of methods. The program was written in ALGOL and run on the English Electric KDF9 computer using the Whetstone ALGOL Translator [8]. In this implementation real numbers are stored as floating-point numbers having effectively a 39 bit mantissa and a 7 bit exponent with an extra two bits to indicate sign.

The initial estimate $\mathbf{H}_0$ to the inverse Jacobian matrix was obtained in every case and for every method by evaluating the approximation to the matrix elements $\partial f_j/\partial x_k$ using the equation

$$\frac{\partial f_j}{\partial x_k} \cong \frac{f_j(x_k + h_k) - f_j(x_k)}{h_k} \tag{7.1}$$

and inverting the resulting matrix by solving a matrix equation whose right hand sides were the n columns of the unit matrix using Gaussian elimination with row interchanges to select the pivot having largest modulus. The choice of h_k was arbitrary and was taken to be roughly one-thousandth part of x_k.

A first value $t^{(1)}$ of t was chosen to be unity since this is the value arising naturally in Newton's method. The second value, if required, was given by the equation

$$t^{(2)} = \frac{(1 + 6\theta)^{1/2} - 1}{3\theta} \tag{7.2}$$

where

$$\theta = \phi(1)/\phi(0) \tag{7.3}$$

and where $\phi(t)$ is the square of the Euclidean norm of $\mathbf{f}(t)$. Equation (7.2) is derived as follows. It is clear that if each f_j were a linear function of t and if the matrix $\mathbf{H}_i$ used to compute $\mathbf{p}_i$ was the true inverse negative Jacobian, then $\phi(t)$ would be a quadratic function of t having the value zero at $t = 1$. Since this func-

tion cannot be negative, it must therefore have zero slope at $t = 1$. Denoting this ideal quadratic approximation by $\phi_q(t)$, it follows that for $\phi_q(t)$ to satisfy the conditions at $t = 1$ it must be given by

$$\phi_q(t) \equiv (1 - 2t + t^2)\phi(0). \tag{7.4}$$

In practice, of course, the square of the Euclidean norm at $t = 1$ is very rarely equal to zero and the assumption is now made that this is due solely to the presence of an additional cubic term. Hence the function ϕ is assumed to be approximated by the cubic function ϕ_c where

$$\phi_c \equiv \phi_q + at^3. \tag{7.5}$$

By putting $t = 1$ in (7.5) it follows that $a = \phi(1)$ and hence

$$\phi_c \equiv (1 - 2t + t^2)\phi(0) + t^3\phi(1). \tag{7.6}$$

Now $t^{(2)}$ is chosen to be the value of t that minimises ϕ_c and differentiating (7.6), equating to zero and selecting the root lying between zero and unity gives equation (7.2).

If further improvement to ϕ is required, it is approximated by a quadratic function whose ordinates are specified at $t = 0$, 1 and $t^{(2)}$. If this function is convex, then $t^{(3)}$ is chosen to be the value of t that minimises it, and a new quadratic is formed whose ordinates are specified at $t^{(3)}$ and two of the previous triad of values of t. Since the precise method of reducing or minimising the norm is peripheral to the theory, it will be described no further here, the actual ALGOL procedure used being given in Appendix 2.

The program, in addition to being able to test Methods 1 and 2 for any combination of minimising/reducing norm with incremental/full step choice of s was also able to test both the "constant matrix" and "basic" methods, either minimising or reducing the norm at each step. In the constant matrix method $\mathbf{H}_0$ is set up and inverted as described above, and this matrix is then used unchanged to obtain $\mathbf{p}_i$ throughout the remainder of the calculation. In the basic method the matrix $\mathbf{H}_i$ is computed afresh at each step using the same algorithm that was employed to compute $\mathbf{H}_0$. In the test program the increments h_k were chosen initially to be about one-thousandth part of the variables x_k and were then held constant for the rest of the calculation.

8. Basis for Comparison. The criteria by which the merit of any algorithm may be assessed are the accuracy of the solution obtained, the time taken to reach this solution and the amount of computer storage space used in the computation. Frequently the second of these criteria is in conflict with the first and the third and some arbitrary assumptions must be made before it is possible to give an overall comparison of two different algorithms.

In the comparisons made in this paper storage space requirements are omitted completely, time and accuracy alone being considered. Since it is impossible to compare times accurately when two methods are tested on two different machines or using two different programming systems, the further assumptions are made that the total amount of computation, and hence the time taken, is directly proportional to the number of evaluations of the vector function $\mathbf{f}$ that are necessary to obtain

the solution and that the constant of proportionality is the same for all methods. This is approximately true when the functions are simple and becomes more nearly true as the complexity of the functions increases and the constant of proportionality approaches unity. This measure of computing time is clearly more satisfactory than the number of iterations since the amount of labour involved in an iteration may vary from the evaluation of $n + 1$ functions and the inversion of a matrix in the basic method to the evaluation of perhaps only one function and a few matrix-vector multiplications in Method 1 with norm reduction.

In order to compare the performance of the various methods under as nearly identical conditions as possible they were used to solve a set of test problems, starting each problem from the same initial conditions. Since, however, the course of computation varied with each method, it was impossible to terminate the calculations at the same point; the accuracy of the final solution as measured by the Euclidean norm of the vector $\mathbf{f}$ varied from method to method, although in each case the iteration was terminated as soon as this norm became less than a tolerance arbitrarily chosen to be 10^{-6}. Now one of the characteristics of Newton's method is that the rate of convergence increases as the solution is approached and it is quite possible for the final iteration to reduce the norm of $\mathbf{f}$ by as much as a factor of 10^{-2}. Hence although the iteration is terminated immediately the norm becomes less than 10^{-6}, the final norm may be as low as 10^{-8}, and it was thought necessary to include this variation when assessing the merit of a particular method.

The measure of efficiency of a method for solving a particular problem was thus chosen to be the mean convergence rate R, where R is given by

$$R = \frac{1}{m} \ln \frac{N_1}{N_m} \tag{8.1}$$

and where m is the total number of function evaluations and N_1, N_m the initial and final Euclidean norms of $\mathbf{f}$.

Although this definition of R is similar to Varga's definition of mean convergence rate [10], it must be emphasised that the R defined by (8.1) has none of the properties of that defined by Varga and in particular does not tend to an asymptotic value as $m \to \infty$. It is merely intended to be a concise index of the performance of a method applied to a particular problem in a particular situation and has no relevance outside the context of the problem and situation to which it refers.

9. Case Histories and Discussion. This section is devoted to a discussion of the behaviour of the various methods when applied to a number of test cases. The performance of the methods are summarised for each test problem in a table which gives the initial and final Euclidean norms N_1 and N_m of $\mathbf{f}$, the total number m of function evaluations required to reduce the norm to N_m and the mean rate of convergence R as defined in the previous section. The number of function evaluations m includes in every case those evaluations required initially to set up the approximation to the Jacobian matrix, and an asterisk against this figure indicates that the iteration had not converged after that number of evaluations, and the run had been terminated since the number of evaluations already exceeded the number needed for convergence using a superior method. The tables are numbered to correspond with the case numbers.

While the program was being developed, it became increasingly apparent that Method 2 was quite useless. In every case tried it rapidly reached a state where successive step vectors $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ were identical and further progress thus became impossible, and for this reason it is omitted from the following case histories.

Cases 1–3. The equations from these three cases arose from design studies on a pneumatically-damped hydraulic system and involve both logarithms and fractional powers. The three cases analysed were taken from a large number of cases, all of which were difficult to solve. The first case has three independent variables and the second and third four. The third case is merely the second case with an improved initial estimate.

Case 4. The equations here represent a counterflow gas/water steam-raising unit. They involve logarithms and various polynomial approximations of physical quantities, e.g. the saturation temperature of water as a function of pressure. There are only two independent variables and despite the magnitude of N_1 the equations were much easier to solve than those used to provide Cases 1–3.

Cases 5–8. These cases solved the specially-constructed equations:

$$f_1 = -(3 + \alpha x_1)x_1 + 2x_2 - \beta,$$

$$f_i = x_{i-1} - (3 + \alpha x_i)x_i + 2x_{i+1} - \beta, \qquad i = 2, 3, \cdots, n-1,$$

$$f_n = x_{n-1} - (3 + \alpha x_n)x_n - \beta.$$

Values of α, β and n are given in the tables, and the initial estimates in all cases were

$$x_i = -1.0, \qquad i = 1, 2, \cdots, n.$$

Case 9. These equations were taken from a paper by Powell [7]:

$$f_1 = 10(x_2 - x_1^2),$$

$$f_2 = 1 - x_1,$$

and the initial values of the independent variables were

$$x_1 = -1.2,$$

$$x_2 = 1.0.$$

Case 10. The equations for this case were due to Freudenstein and Roth [3]:

$$f_1 = -13 + x_1 + ((-x_2 + 5)x_2 - 2)x_2,$$

$$f_2 = -29 + x_1 + ((x_2 + 1)x_2 - 14)x_2.$$

The initial values were

$$x_1 = 15,$$

$$x_2 = -2.$$

An examination of Tables 1–4 shows that of the eight methods tested, three were consistently good for all the first four test cases. These were the full step norm-reducing variant of Method 1 and both versions of the basic method. All the other methods were either uniformly bad, or tended to be erratic in their behaviour. In

particular the norm-minimisation variants tended to be bad, or at least worse than the corresponding norm-reducing variants, and the incremental methods were either quite good or very bad. The reason for this latter behaviour is probably that the method chosen to select s is, in retrospect, not particularly good, for in several iterations the modulus of t was considerably less than the modulus of s. In view of this the incremental methods were not tested further, although it is possible that with a better method of choosing s they might become competitive.

The reason, on the other hand, for the poor performance of the norm-minimisation methods is that they appear to be inherently inferior to the norm-reduction ones. For to minimise the norm, regarded as a function of one variable, to the specified tolerance from four to six function evaluations in general were required whereas the comparable figure for norm reduction was from one to three. Hence unless the number of iterations was very considerably less for the minimisation methods they would require more evaluations than the reduction methods. In practice this was found not to occur, and in particular for the full step variant of Method 1 the number of iterations required when reducing the norm was in all four cases less than the number required when norm minimisation was used.

The constant matrix method did reasonably well if a good initial estimate was available, and the Jacobian matrix at that point was a good approximation to that at the solution. In no case, however, was it superior to the full step norm-reducing variant of Method 1, and if the initial estimate was poor, as in Case 1, it did very badly indeed.

TABLE 1
$N_1 = 94.33$

Method	N_m	m	R
Method 1, minimising, full step	$4.456_{10} - 3$	218*	0.046
Method 1, minimising, incremental	$8.843_{10} - 5$	233*	0.060
Method 1, reducing, full step	$1.289_{10} - 7$	75	0.272
Method 1, reducing, incremental	$2.198_{10} - 1$	193*	0.031
Constant matrix, minimising	$7.035_{10} - 1$	386*	0.013
Constant matrix, reducing	4.619	116*	0.026
Basic, minimising	$2.557_{10} - 8$	53	0.416
Basic, reducing	$2.990_{10} - 7$	33	0.593

TABLE 2
$N_1 = 3.050$

Method	N_m	m	R
Method 1, minimising, full step	$7.907_{10} - 7$	139	0.109
Method 1, minimising, incremental	$4.278_{10} - 6$	183*	0.074
Method 1, reducing, full step	$2.590_{10} - 8$	44	0.422
Method 1, reducing, incremental	$1.792_{10} - 8$	41	0.462
Constant matrix, minimising	$2.572_{10} - 1$	416*	0.006
Constant matrix, reducing	$1.508_{10} - 3$	45*	0.169
Basic, minimising	$8.476_{10} - 8$	69	0.252
Basic, reducing	$1.394_{10} - 8$	38	0.505

TABLE 3
$N_1 = 1.059_{10} - 1$

Method	N_m	m	R
Method 1, minimising, full step	$6.891_{10} - 8$	30	0.475
Method 1, minimising, incremental	$9.374_{10} - 7$	25	0.465
Method 1, reducing, full step	$1.593_{10} - 7$	10	1.341
Method 1, reducing, incremental	$1.593_{10} - 7$	10	1.341
Constant matrix, minimising	$5.160_{10} - 7$	30	0.408
Constant matrix, reducing	$4.266_{10} - 7$	11	1.129
Basic, minimising	$6.630_{10} - 8$	24	0.595
Basic, reducing	$4.649_{10} - 8$	16	0.915

TABLE 4
$N_1 = 1.418_{10} + 2$

Method	N_m	m	R
Method 1, minimising, full step	$1.008_{10} - 4$	106*	0.134
Method 1, minimising, incremental	$1.601_{10} - 5$	133*	0.120
Method 1, reducing, full step	$1.689_{10} - 7$	20	1.027
Method 1, reducing, incremental	1.164	62*	0.077
Constant matrix, minimising	$1.044_{10} + 2$	205*	0.001
Constant matrix, reducing	$1.047_{10} - 6$	40*	0.468
Basic, minimising	$1.245_{10} - 7$	52	0.401
Basic, reducing	$2.608_{10} - 8$	35	0.640

On comparing the two variants of the basic method, it is seen that the norm-reducing one was in all four cases superior to the norm-minimising one. Thus of the three methods that did consistently well in the first four cases one was always inferior to another, and for this reason was not tested further. This left only two methods which were worth more extensive testing, and these were tried with six more test cases.

On comparing the performances of the full step norm-reducing variant of Method 1 and the reducing variant of the basic method (subsequently referred to as method 1/fsr and basic/r), it appeared that method 1/fsr was superior to the basic/r method if the problem were only mildly nonlinear, and it was conjectured that this superiority would become more evident as the number of independent variables was increased. Cases 5–8 were chosen to test this conjecture and do in fact tend to support it.

Case 9 was taken from a paper by Powell where he describes a method for minimising the sum of squares of nonlinear functions. If this sum happens to be zero, then this procedure does in fact solve a set of nonlinear equations. The poor value of R (computed from data quoted in [7]) achieved by Powell's method is due not so much to the fact that this method required more function evaluations than either method 1/fsr or the basic/r method but to the phenomenal improvement of the norm that occurred during the final iteration of the latter two methods. In each case the norm was reduced from greater than 10^{-2} to its final value during the last

iteration. Because of this the number of function evaluations may be a better criterion of merit than the mean convergence rate, and using this criterion the basic/r method is again easily the best with method 1/fsr a poor second. It would thus appear that for certain problems the advantage shared by method 1/fsr and Powell's method of not requiring the computation of derivatives is more apparent than real.

TABLE 5

$N_1 = 1.910 \quad \alpha = -0.1 \quad \beta = 1.0 \quad n = 5$

Method	N_m	m	R
Method 1, reducing, full step	$1.991_{10} - 7$	11	1.462
Basic, reducing	$4.577_{10} - 8$	19	0.924

TABLE 6

$N_1 = 1.803 \quad \alpha = -0.5 \quad \beta = 1.0 \quad n = 5$

Method	N_m	m	R
Method 1, reducing, full step	$9.007_{10} - 8$	11	1.528
Basic, reducing	$8.918_{10} - 8$	19	0.885

TABLE 7

$N_1 = 2.121 \quad \alpha = -0.5 \quad \beta = 1.0 \quad n = 10$

Method	N_m	m	R
Method 1, reducing, full step	$1.597_{10} - 7$	18	0.911
Basic, reducing	$2.602_{10} - 7$	34	0.468

TABLE 8

$N_1 = 2.646 \quad \alpha = -0.5 \quad \beta = 1.0 \quad n = 20$

Method	N_m	m	R
Method 1, reducing, full step	$3.611_{10} - 7$	29	0.545
Basic, reducing	$4.185_{10} - 6$	64*	0.209

TABLE 9

$N_1 = 4.919$

Method	N_m	m	R
Method 1, reducing, full step	$4.729_{10} - 10$	59	0.391
Basic, reducing	$2.547_{10} - 10$	39	0.607
Powell	$1.000_{10} - 4$	70	0.154

Case 10, for which no table is given, is due to Freudenstein and Roth. Both method 1/fsr and the basic/r methods failed to reach a solution although both came to grief at a point that proved to be a local minimum of the norm of $\mathbf{f}$. Now the norm of $\mathbf{f}$ can be a minimum greater than zero only if the Jacobian matrix is singular at that point, and this contravenes one of the cardinal requirements for the success of Newton's method. For cases like this neither Newton's method nor any of its derivatives is likely to succeed and some radical innovation, like that due to Freudenstein and Roth, is necessary.

10. Conclusions. From the discussion in the previous section the following conclusions emerge.

(1) Norm reduction is a better strategy than norm minimisation.

(2) If a reasonably good initial estimate of the solution is available method 1/fsr is superior to the basic/r method.

(3) If a reasonably good initial estimate is not available then the basic/r method is superior to method 1/fsr.

Conclusions (2) and (3) are somewhat tentative, and the term "reasonably good" is itself rather vague. Furthermore the behaviour of the various methods depends upon the nature of the problem they are attempting to solve, and it is thus unlikely that the few test problems selected give a sufficiently accurate picture of the overall behaviour of the algorithms to enable any more precise statements on the relative merits of the methods to be made. However, since method 1/fsr has never failed to converge when the basic/r method has converged and since for many of the test cases it was superior to the basic/r method, it may prove to be a useful alternative, especially if a good initial estimate of $\mathbf{H}_i$ is available. This would occur in the method of Freudenstein and Roth or in solving a set of ordinary differential equations by a predictor-corrector technique, when the corrector equations could be solved by Method 1 using as an initial estimate for $\mathbf{H}_i$ the final $\mathbf{H}_i$ obtained during the previous step.

Acknowledgments. The author is glad to acknowledge his indebtedness to Mr. F. Ford for reading the manuscript and making several valuable suggestions and to the directors of the English Electric Co. Ltd. for permission to publish this work.

Applied Analysis Section,
Atomic Power Division
English Electric Co. Ltd.,
Whetstone, Leicester, England.

APPENDIX 1. *Summary of Method 1/fsr*

1. Obtain an initial estimate $\mathbf{x}_0$ of the solution.

2. Obtain an initial value $\mathbf{H}_0$ of the iteration matrix. This may be done either by computing and inverting the Jacobian matrix, or it is possible that a sufficiently good approximation may be available from some other source, e.g. the final value of $\mathbf{H}_i$ from some previous calculation.

3. Compute $\mathbf{f}_i = \mathbf{f}(\mathbf{x}_i)$.

4. Compute $\mathbf{p}_i = \mathbf{H}_i\mathbf{f}_i$.

5. Select a value t_i of t such that the norm of $\mathbf{f}(\mathbf{x}_i + t_i\mathbf{p}_i)$ is less than the norm of $\mathbf{f}(\mathbf{x}_i)$. This is done iteratively and a modified version of the ALGOL procedure given in Appendix 2 may be used. In the course of this calculation the following will have been computed:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + t_i\mathbf{p}_i,$$

$$\mathbf{f}_{i+1} = \mathbf{f}(\mathbf{x}_{i+1}).$$

6. Test $\mathbf{f}_{i+1}$ for convergence. If the iteration has not yet converged, then:
7. Compute $\mathbf{y}_i = \mathbf{f}_{i+1} - \mathbf{f}_i$.
8. Compute $\mathbf{H}_{i+1} = \mathbf{H}_i - (\mathbf{H}_i\mathbf{y}_i + t_i\mathbf{p}_i)\mathbf{p}_i^T\mathbf{H}_i/\mathbf{p}_i^T\mathbf{H}_i\mathbf{y}_i$.
9. Repeat from step 4.

APPENDIX 2. The minimisation of the Euclidean norm of $\mathbf{f}$ regarded as a function of the single variable t was carried out by a slightly modified version of the ALGOL procedure quadmin. The modifications enabled the procedure to perform some additional housekeeping operations and in no way affected the course of the iteration. On calling the procedure, the nonlocal variables n, last norm, x and p were set up as follows.

Nonlocal variable	*Value*
n	Number of independent variables
lastnorm	$\mathbf{f}_i^T\mathbf{f}_i$
x	$\mathbf{x}_i + \mathbf{p}_i$
p	$\mathbf{p}_i$

The procedure compute calculates the vector $\mathbf{f}(\mathbf{x}_i + t\mathbf{p}_i)$ for the current value of t, where $\mathbf{x}_i + t\mathbf{p}_i$ is stored in the array x, and stores it in the nonlocal array f. The real procedure evalcrit computes the square of the Euclidean norm of the vector stored in the array f. On entry to quadmin t is set to zero to prevent premature emergence, but emergence is enforced when the number of iterations exceeds 10.

```
procedure quadmin;
begin   comment This procedure minimises the Euclidean norm of the re-
                siduals, treated as a function of the single variable t.
                Quadratic interpolation is used;
        procedure set (i); value i; integer i;
        begin   integer j;
                ii := i; vt[i] := t;
                for j := 1 step 1 until n do x[j] := x[j] + (t - last t) × p[j];
                goto recompute;
        end set;
        real xx, xw, xy, last t, norm;
        integer i, ii, upper, mid, lower, iterone;
        array phi, vt[1:3];
        last t := 1.0; t := 0; iterone := 0;
recompute: iterone := iterone + 1;
        compute; comment f now contains an approximation to f_{i+1} ;
        norm := evalcrit;
        if (abs(t - last t) > 0.01 × abs(t) and iterone ≦ 10) or iterone = 2
        then begin      if iterone = 1 then
                        begin   vt[1] := 0; vt[3] := 1.0; phi[1] := lastnorm;
                                phi[3] := norm;
```

```
                    xx := norm/lastnorm; t := (sqrt(1.0 + 6.0 ×
                      xx) − 1.0)/(3.0 × xx);
                    lower := 1; mid := 2; upper := 3; set (2);
                end else
                begin   phi[ii] := norm;
                    xw := vt[2] − vt[3]; xx := vt[3] − vt[1]; xy :=
                      vt[1] − vt[2];
                    xw := −(phi[1] × xw + phi[2] × xx + phi[3] ×
                      xy)/(xw × xx × xy);
                    xx := (phi[1] − phi[2])/xy − xw × (vt[1] +
                      vt[2]);
                    last t := t;
                    t := (if xw > 0 then −xx/(2.0 × xw) else if
                      phi[upper] > phi[lower]
                    then 3.0 × vt[lower] − 2.0 × vt[mid] else 3.0 ×
                      vt[upper] − 2.0 × vt[mid]);
                    if t > vt[upper] then
                    begin   i := lower; lower := mid; mid :=
                      upper; upper := i; set(upper);
                    end else if t < vt[lower] then
                    begin   i := upper; upper := mid; mid :=
                      lower; lower := i; set(lower);
                    end else if t > vt[mid] then
                    begin   i := lower; lower := mid; mid := i;
                      set(mid);
                    end else
                    begin   i := upper; upper := mid; mid := i;
                      set(mid);
                    end;
                end;
            end modify step;
            lastnorm := norm;
    end quadmin;
```

1. W. C. DAVIDON, "Variable metric method for minimization," A.E.C. Research and Development Report, ANL-5990 (Rev. TID-4500, 14th ed.), 1959.
2. R. FLETCHER & M. J. D. POWELL, "A rapidly convergent descent method for minimization," *Comput. J.*, v. 6, 1963/64, pp. 163–168. MR **27** #2096.
3. F. FREUDENSTEIN & B. ROTH, "Numerical solutions of systems of nonlinear equations," *J. Assoc. Comput. Mach.*, v. 10, 1963, pp. 550–556.
4. A. S. HOUSEHOLDER, *Principles of Numerical Analysis*, McGraw-Hill, New York, 1953, pp. 135–138. MR **15**, 470.
5. W. KIZNER, "A numerical method for finding solutions of nonlinear equations," *J. Soc. Indust. Appl. Math.*, v. 12, 1964, pp. 424–428.
6. M. J. D. POWELL, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.*, v. 7, 1964, pp. 155–162.
7. M. J. D. POWELL, "A method for minimizing a sum of squares of non-linear functions without calculating derivatives," *Comput. J.*, v. 7, 1965, pp. 303–307.
8. B. RANDELL, "The Whetstone KDF9 ALGOL Translator," *Introduction to System Programming*, P. Wegner (Ed.), Academic Press, London, 1964, pp. 122–136.
9. R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, N. J. and International, London, 1962, p. 9. MR **28** #1725.
10. R. S. VARGA, op. cit., p. 62.