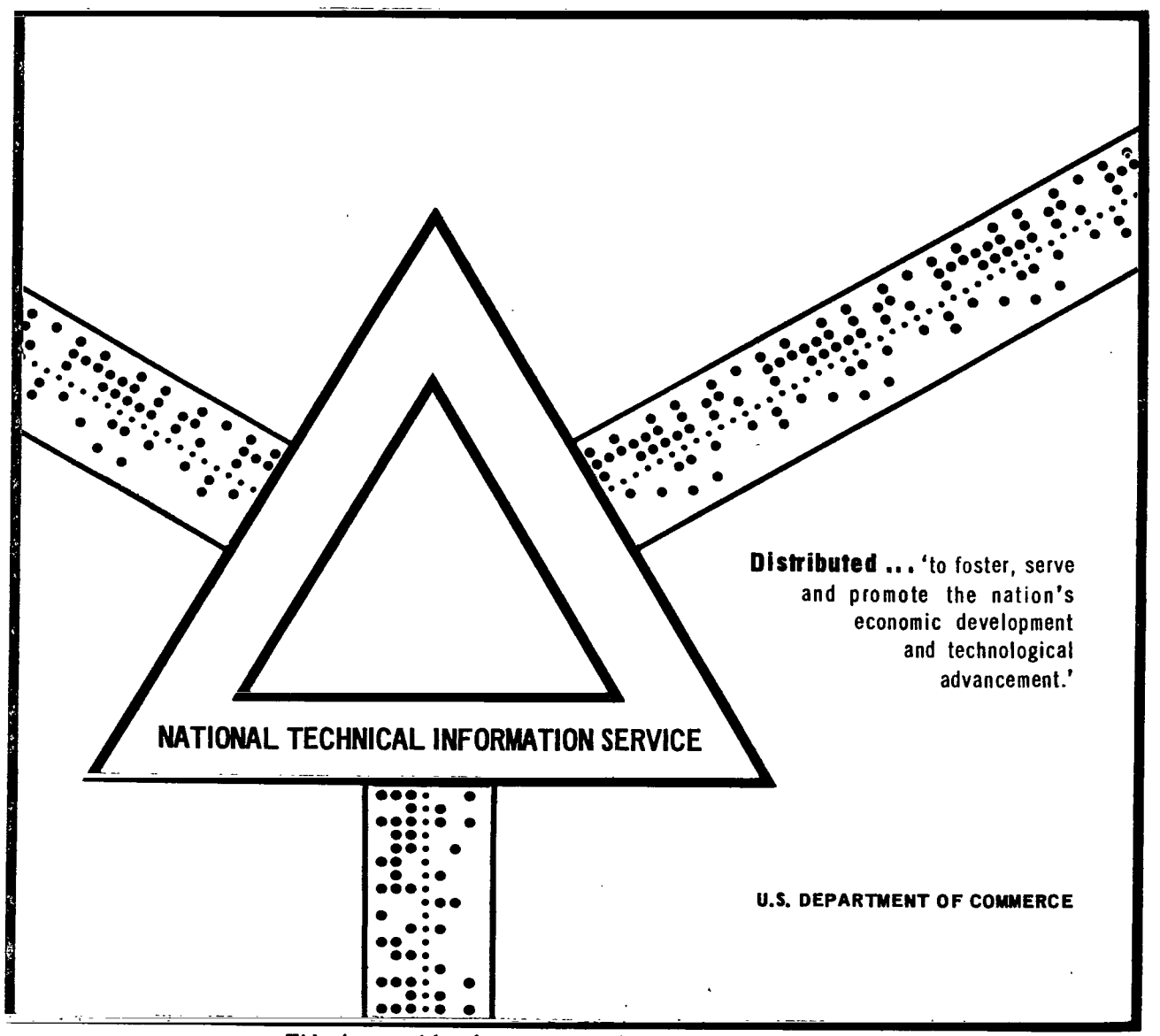
VARIANTS OF THE SECANT METHOD FOR SOLVING NONLINEAR SYSTEMS OF EQUATIONS

Clarence Cantor

Goddard Space Flight Center
Greenbelt, Maryland

January 1971



X-733-71-40
PREPRINT

NASA TM X-65453

VARIANTS OF THE SECANT METHOD FOR SOLVING NONLINEAR SYSTEMS OF EQUATIONS

CLARENCE CANTOR

JANUARY 1971

GSFC
GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-733-71-48

VARIANTS OF THE SECANT METHOD FOR SOLVING NONLINEAR SYSTEMS OF EQUATIONS

Clarence Cantor

January 1971

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland

VARIANTS OF THE SECANT METHOD FOR SOLVING NONLINEAR SYSTEMS OF EQUATIONS

Clarence Cantor

ABSTRACT

Some variants of the Secant Method are developed for solving $f(x) = 0$, n nonlinear equations in n unknowns. The new methods, consisting of Algorithms I and II, depart from existing versions of the Secant Method whenever certain conditions arise that would tend to cause poor convergence of the latter. These conditions are ascertained via simple test criteria associated with the new algorithms. When these criteria are satisfied initially at each step, the algorithms follow the same steps as existing versions of the Secant Method and, under certain assumptions, are shown to possess superlinear convergence. Whenever these test criteria are not satisfied initially, the algorithms follow logical alternate procedures that provide a basis for linear convergence. The results of numerical experiments with a series of randomly generated problems support the claim of improved convergence for the new methods. Of the two new algorithms, Algorithm II is judged to be the superior and warrants further numerical investigation.

CONTENTS

LIST OF TABLES

ACKNOWLEDGEMENTS

The author wishes to thank Dr. F. P. Emad of the University of Maryland for his guidance during this research. The research was supported by the Goddard Space Center of the National Aeronautics and Space Administration.

VARIANTS OF THE SECANT METHOD FOR SOLVING NONLINEAR SYSTEMS OF EQUATIONS

INTRODUCTION

The problem of solving $f(x) = 0$, n nonlinear equations in n unknowns, has many applications. For example, the equilibrium points of the n-th order dynamical system represented by the vector equation $\dot{x} = f(x)$ are simply the solutions $\bar{x}$ to $f(x) = 0$, In a discrete system given by the vector equation $x^{k+1} = g(x^k)$, the equilibrium points are obtained by solving $x = g(x)$ which is equivalent to solving $f(x) \triangleq g(x) - x = 0$. The problem can also arise as a result of the requirement to minimize (or maximize) some functional; equating the gradient to zero results in $f(x) = 0$, n equations in n unknowns. Except for very specialized sets of nonlinear equations, all methods for solving $f(x) = 0$ depend on iterative techniques, which are readily implemented by computers.

Some well known methods for solving $f(x) = 0$ are summarized in Section I with the emphasis on existing versions of the Secant Method and their limitations. The Secant Method has the attractive feature of requiring only one evaluation of the vector $f(x)$ per iteration (as opposed to $n + 1$ function evaluations in a discrete Newton's Method for example). However it suffers from poor convergence at times and this characteristic becomes more prevalent as the order of the system increases. The goal behind the development of the Secant Method variants described in Section II is to alleviate the conditions tending to cause poor convergence in present versions of the Secant Method. The results of theoretical considerations (Section III) and numerical experiments (Section IV) indicate that this goal has been largely realized.

SECTION I

BACKGROUND

Newton's Method in n dimensions (see e.g., [11], [16], [17], and [23]) is probably the most widely known method for solving the n-th order vector equation $f(x) = 0$. Analogous to the case of one dimension, Newton's Method in n dimensions is given by

$$x^{k+1} = x^k - J^{-1}(x^k)\, f(x^k) \qquad (1.1)$$

(assuming $J^{-1}(x^k)$ exists)

where

$f(x^k)$ is the value of $f(x)$ at $x = x^k$

$J(x^k)$ is the Jacobian matrix of $f(x)$ evaluated at x^k

x^k is the present (k-th) estimate of the n-dimensional solution

and

x^{k+1} is the next estimate of the solution.

Newton's Method is a direct result of linearizing the system $f(x) = 0$ about the point $x = x^k$. A Taylor's expansion of $f(x)$ about $x = x^k$, in which all terms above the first order are dropped, yields (1.1).

Assuming $f(x)$ is continuously differentiable in a neighborhood of a solution $\bar{x}$ and that the Jacobian matrix is nonsingular in this neighborhood, Newton's Method will converge to $\bar{x}$ if the starting estimate x^1 is "sufficiently close" to $\bar{x}$. In practice, Newton's Method suffers from three principal disadvantages:

1. The region of convergence in some problems may be very small. Thus the method may fail to converge for all but fairly accurate initial estimates x^1.

2. The Jacobian $J(x^k)$ requires the evaluation of n^2 partial derivatives at each iterate x^k. This can involve a considerable amount of computations even when the partial derivatives are approximated by finite differences, since $n + 1$ evaluations of $f(x)$ per iteration would then be required.

3. The inverse $J^{-1}(x^k)$ must be calculated at each step, again involving an undue amount of computations.

These difficulties are more or less overcome in the class of quasi-Newton Methods represented by

$$x^{k+1} = x^k - \alpha^k H^k f^k \tag{1.2}$$

where

H^k is some approximation to $J^{-1}(x^k)$

$f^k = f(x^k)$

and

α^k is some real number greater than zero.

If $H^k = J^{-1}(x^k)$ and α^k is chosen to minimize the norm φ^{k+1}, or to reduce φ^{k+1} below φ^k, where

$$\varphi^k \triangleq \sum_{i=1}^{n} (f_i^k)^2 \tag{1.3}$$

(f_i^k = i-th component of vector f^k)

then one has a reduced-step Newton's Method, namely

$$x^{k+1} = x^k - \alpha^k J^{-1}(x^k) f^k . \tag{1.4}$$

At each step in (1.4) a value of α^k greater than zero can always be found which will reduce φ^{k+1} below φ^k, provided that $J(x^k)$ is non-singular and provided

that $f(x^k) \neq 0$. This is easily proved by showing that the direction of Δx^k given by (1.4), namely $-J^{-1}(x^k)\, f^k$, has a component along the negative gradient of φ^k, i.e., by showing that

$$\left(\text{grad}\, \varphi^k,\ J^{-1}(x^k)\, f^k\right) > 0. \tag{1.5}$$

One has

$$\frac{\partial \varphi^k}{\partial x_j^k} = 2 \sum_{i=1}^{n} \frac{\partial f_i^k}{\partial x_j^k} f_i^k = 2 \left(\frac{\partial f^k}{\partial x_j^k}\right)^T f^k$$

or

$$\text{grad}\, \varphi^k = 2\, J^T(x^k)\, f^k. \tag{1.6}$$

Then

$$\left(\text{grad}\, \varphi^k, J^{-1}(x^k)\, f^k\right) = 2(f^k)^T J(x^k)\, J^{-1}(x^k)\, f^k$$

$$= 2(f^k)^T f^k > 0,\ f^k \neq 0. \tag{1.7}$$

This reduced-step Newton's Method, with α^k selected to reduce φ^{k+1} below φ^k, will converge to a solution of $f(x) = 0$ for any starting value in a region bounded by the contour φ = constant, if a solution exists in this region and if $J^{-1}(x)$ exists everywhere in this region. This region of convergence is as large or larger than that in the standard (full-step) Newton's Method so that the disadvantage of a small region of convergence often encountered in the full-step Newton's Method is somewhat alleviated by the reduced-step variant. However, the reduced-step version still has the disadvantages of requiring the calculation of $J(x^k)$ and its inverse at each step.

The Method of Steepest Descent, or Gradient Method (see e.g., [11], [16], [17], [19]), for solving $f(x) = 0$ can be regarded as a quasi-Newton method by letting $H^k = J^T(x^k)$, with α^k selected to minimize or reduce the norm φ^{k+1}. Thus one has

$$x^{k+1} = x^k - \alpha^k J^T(x^k) f^k . \tag{1.8}$$

Note that grad $\varphi^k = 2 J^T(x^k) f^k$ so that the direction of Δx^k given by (1.8) is along the path of steepest descent. Thus a value of $\alpha^k > 0$ can always be found to reduce φ^{k+1} below φ^k as long as $J^T(x^k) f^k \neq 0$. Assuming $J(x^k)$ is non-singular, $f^k \neq 0$ implies $J^T(x^k) f^k \neq 0$. In addition, we can sometimes satisfy $J^T(x^k) f^k \neq 0$ even though $J(x^k)$ is singular. Thus the region of convergence in the Gradient Method is as large or larger than that in the reduced-step or regular (full-step) Newton's Method. In addition, this method eliminates the need to calculate the inverse of $J(x^k)$. However, the Gradient Method lacks the quadratic convergence of Newton's Method. Its behavior near a solution of $f(x) = 0$ is rather oscillatory in general.

A method which combines the feature of the larger region of convergence inherent in the Gradient Method together with the quadratic convergence of Newton's Method near a solution, is Marquardt's Algorithm [14],

$$x^{k+1} = x^k - \left[J^T(x^k) J(x^k) + \lambda^k I\right]^{-1} J^T(x^k) f^k . \tag{1.9}$$

As $\lambda^k \to 0$, (1.9) becomes Newton's Method, while $\lambda^k \to \infty$ yields the Gradient Method. In general, a small value of λ^k is sought in order for the method to approach the quadratic convergence of Newton's Method, and λ^k is increased only as necessary to satisfy

$$\varphi^{k+1} < \varphi^k , \tag{1-10}$$

which is the basic criterion that must be satisfied at each step. While the convergent properties of Marquardt's algorithm are improvements over those in Newton's Method alone or the Gradient Method alone, the algorithm requires considerably more computations at each step than either of the latter two methods.

If $H^k = J^{-1}(x^1)$ and $\alpha^k = 1$, one obtains the simplified Newton's Method,

$$x^{k+1} = x^k - J^{-1}(x^1)\ f^k . \tag{1.11}$$

This simplified Newton's Method eliminates the need for frequent calculation of $J(x^k)$ and its inverse. In general, this method does not converge as rapidly as Newton's Method. In particular, it lacks the quadratic convergence of Newton's Method, unless by some odd chance, $J(\bar{x}) = J(x^1)$ where $\bar{x}$ is the solution. In addition it suffers, in common with Newton's Method, the fault of a small region of convergence in many problems. This fault can be alleviated somewhat, as was done for Newton's Method, by a reduced-step version, namely,

$$x^{k+1} = x^k - \alpha^k\ J^{-1}(x^1)\ f^k \tag{1.12}$$

where α^k is chosen to minimize φ^{k+1} or to satisfy the condition

$$\varphi^{k+1} < \varphi^k . \tag{1.13}$$

The Secant Method [2], [16] can be expressed as a quasi-Newton method in which $\alpha^k = 1$ and $H^k \triangleq (J^k)^{-1}$ is the inverse Jacobian of the equivalent linear system corresponding to the last $n + 1$ points. It is a generalization of the

Secant Method in one dimension, in which a straight line is passed through the two points (x^k, f^k) and (x^{k-1}, f^{k-1}); the intersection of this line with the x axis determines x^{k+1}. Thus in n dimensions, one has

$$x^{k+1} = x^k - (J^k)^{-1} f^k . \tag{1.14}$$

The matrix J^k (or $(J^k)^{-1}$) is determined from the requirement that

$$f^i = J^k \left[x^i - x^*\right], \qquad i = k, k-1, \ldots, k-n \tag{1.15}$$

where $x^* \triangleq x^{k+1}$ is the next estimate of the solution. Letting

$$\Delta x^i \triangleq x^{i+1} - x^i \tag{1.16}$$

and

$$\Delta f^i \triangleq f^{i+1} - f^i$$

equation (1.15) yields

$$\Delta f^i = J^k \Delta x^i, \qquad i = k-1, k-2, \ldots, k-n. \tag{1.17}$$

With the $n \times n$ matrices ΔX^k and ΔF^k defined as

$$\Delta X^k \triangleq \left[\Delta x^{k-n} \ldots \Delta x^{k-2} \Delta x^{k-1}\right] \tag{1.18}$$

and

$$\Delta F^k \triangleq \left[\Delta f^{k-n} \ldots \Delta f^{k-2} \Delta f^{k-1}\right]$$

equation (1.17) yields

$$\Delta F^k = J^k \Delta X^k \tag{1.19}$$

or

$$J^k = \Delta F^k \, (\Delta X^k)^{-1}$$

(assuming $(\Delta X^k)^{-1}$ exists)

and

$$H^k \triangleq (J^k)^{-1} = \Delta X^k \, (\Delta F^k)^{-1} \tag{1.20}$$

(assuming $(\Delta F^k)^{-1}$ exists).

Thus, solving equation (1.20) for $(J^k)^{-1}$ and substituting in (1.14) yields x^{k+1}, the next estimate of the solution to $f(x) = 0$, after which the process is repeated. The Secant Method is defined even if ΔX^k is singular, since the existence of H^k requires only that $(\Delta F^k)^{-1}$ exists. However, if ΔX^k is singular (implying H^k is singular), then it can be shown that the iteration cannot converge to a solution except under specialized conditions. Note that J^k in the Secant Method is not the same as $J(x^k)$ in Newton's Method. The latter represents the Jacobian at the point x^k (formed from n^2 partial derivatives $\frac{\partial f_i}{\partial x_j}$ evaluated at $x = x^k$) while J^k is the Jacobian of the linear system (1.15) that interpolates the $n + 1$ points $(x^k, f^k), (x^{k-1}, f^{k-1}), \ldots, (x^{k-n}, f^{k-n})$.

An equivalent representation of the Secant Method is obtained by defining

$$\delta x^i \triangleq x^k - x^i, \quad \delta f^i \triangleq f^k - f^i \tag{1.21}$$

$$i = k-1, k-2, \ldots, k-n$$

and $n \times n$ matrices δX^k and δF^k as

$$\delta X^k \triangleq \left[\delta x^{k-n} \ \ldots \ \delta x^{k-2} \ \delta x^{k-1}\right] \tag{1.22}$$

and

$$\delta F^k \triangleq \left[\delta f^{k-n} \ldots \delta f^{k-2}\ \delta f^{k-1}\right].$$

Then equation (1.15) yields

$$\delta f^i = J^k \delta x^i, \qquad i = k-1, \ldots, k-n \tag{1.23}$$

$$\delta F^k = J^k \delta X^k \tag{1.24}$$

or

$$J^k = \delta F^k (\delta X^k)^{-1}$$

(assuming $(\delta X^k)^{-1}$ exists)

and

$$(J^k)^{-1} = \delta X^k (\delta F^k)^{-1} \tag{1.25}$$

(assuming $(\delta F^k)^{-1}$ exists).

The matrix J^k and its inverse, defined by (1.24) and (1.25), are the same as the corresponding ones obtained from (1.19) and (1.20), since both representations are derived from the same linear system (1.15); assuming the existence of the required inverses, $n+1$ points (x^i, f^i) uniquely define the parameters of an n-th order linear system. It is also easy to show that $(\delta X^k)^{-1}$ exists if and only if $(\Delta X^k)^{-1}$ exists and that $(\delta F^k)^{-1}$ exists if and only if $(\Delta F^k)^{-1}$ exists. Thus equations (1.25) and (1.14) form an equivalent representation of the Secant Method.

The Secant Method avoids the necessity of calculating $J(x^k)$ (and its n^2 partial derivatives) at each step but still requires (in the quasi-Newton form of the Secant Method) a matrix inversion and multiplication at each step. The Secant Method has local superlinear convergence provided the last n vectors Δx^i

forming ΔX^k remain sufficiently independent (see Section III). But there is no way of insuring that this condition is met at every step, so that convergence, let alone superlinear convergence, is not assured in general. Even when local convergence occurs, the region of convergence may be rather limited in many problems as is the case for the full-step Newton's Method. Again, this region can be enlarged in the reduced-step version, namely

$$x^{k+1} = x^k - \alpha^k (J^k)^{-1} f^k \tag{1.26}$$

with $(J^k)^{-1}$ determined as before from (1.20) or (1.25) and α^k selected to satisfy (1.13).

Probably the greatest drawback of the Secant Method is the numerical instability and lack of convergence resulting from near singular matrices ΔX^k and ΔF^k. This occurs when Δx^{k-1} (Δf^{k-1}) is "nearly" a linear combination of the previous $n - 1$ vectors. The matrix J^k determined from such near singular matrices fails to approach $J(x^k)$ at the solution, thus ruining the theoretical convergence of the Secant Method.

It is not necessary to calculate J^k or its inverse explicitly in the Secant Method in order to solve for $x^* = x^{k+1}$, the next estimate of the solution. Following Wolfe's derivation [25], one can denote the last set of $n + 1$ points by (x^1, f^1), (x^2, f^2), ..., (x^{n+1}, f^{n+1}). Then equation (1.15) would read

$$f^i = J(x^i - x^*), \quad i = 1, 2 \ldots, n + 1 \tag{1.27}$$

where J is the matrix of the linear system corresponding to these $n + 1$ points (x^i, f^i). Since the $n + 1$ vectors f^i must be linearly dependent, there exist constants π_i not all zero such that

$$\sum_{i=1}^{n+1} \pi_i \, f^i = 0. \tag{1.28}$$

One can normalize the constants π_i such that

$$\sum_{i=1}^{n+1} \pi_i = 1. \tag{1.29}$$

Utilizing equation (1.27) in (1.28), and multiplying by J^{-1} which is assumed to exist, one obtains

$$\sum_{i=1}^{n+1} \pi_i \, (x^i - x^*) = 0. \tag{1.30}$$

Solving for x^* (noting that $\Sigma \pi_i = 1$) yields

$$x^* = \sum_{i=1}^{n+1} \pi_i \, x^i. \tag{1.31}$$

The $n + 1$ constants π_i can be obtained from equations (1.28) and (1.29). With the $(n + 1) \times (n + 1)$ matrix A defined as

$$A \triangleq \begin{bmatrix} f^1 & f^2 & \cdots & f^{n+1} \\ \hline 1 & 1 & \cdots & 1 \end{bmatrix} \tag{1.32}$$

and the $(n + 1)$-vector π as

$$\pi^T \triangleq \left[\pi_1 \; \pi_2 \cdots \pi_{n+1}\right]^T \tag{1.33}$$

equations (1.28) and (1.29) can be written as

$$A\pi = \left[0 \; 0 \cdots 0 \; 1\right]^T \tag{1.34}$$

or

$$\pi = A^{-1} \left[0\ 0\ \ldots\ 0\ 1\right]^T \tag{1.35}$$

Equations (1.35) and (1.31) yield x^*, the next estimate of the solution. Wolfe suggests using the pair $(x^*, f(x^*))$ to replace the pair (x^j, f^j) for which the norm $\varphi^i \triangleq (f^i)^T f^i$ is a maximum. Denoting the new matrix of (1.32) as A^* (in which $f(x^*)$ has replaced f^j), he uses the Sherman-Morrison modification method [12], [20] to obtain a simplified formula for calculating $(A^*)^{-1}$ based on knowledge of the previous inverse, A^{-1}.

This version of the Secant Method has several drawbacks. For one, the matrix J is not explicitly calculated, and in some problems it is necessary or desirable to obtain an approximation to the Jacobian of the system at the solution. Secondly, it is inherently a full-step Secant Method; it does not lend itself to the introduction of a constant α^k different from unity. Thus the norm reducing feature of the reduced-step method, which can enlarge the region of convergence, is not available in this version. Finally, the matrix A often becomes ill-conditioned as we approach a solution, resulting in the same degraded convergence or lack of convergence near the solution that afflicts other versions of the Secant Method. This problem can be alleviated in the proposed variants of the Secant Method, where $(J^k)^{-1}$ is available at each step. This will be discussed later.

Barnes' version of the Secant Method [1] starts with an initial estimate J^1 of the Jacobian, and develops corrections to the estimated Jacobian J at each step as

$$J^{k+1} = J^k + D^k \tag{1.36}$$

where

$$D^k = \frac{f^{k+1} (z^k)^T}{(z^k)^T \Delta x^k}$$

and Δx^k is obtained in the usual way (e.g., see (1.14)) from

$$J^k \Delta x^k = - f^k . \tag{1.37}$$

If $k \leq n$, the vector z^k at each step is selected to be orthogonal to the previous $(k-1)$ vectors $\Delta x^1, \Delta x^2, \ldots, \Delta x^{k-1}$. If $k > n$, z^k is selected to be orthogonal to the previous $n-1$ vectors $\Delta x^{k-n+1}, \ldots, \Delta x^{k-1}$. The result of this choice, using (1.36), is that

$$D^k \Delta x^i = 0, \quad 0 < k - i < n \tag{1.38}$$

and

$$J^{k+1} \Delta x^i = J^k \Delta x^i, \quad 0 < k - i < n . \tag{1.39}$$

Since (1.39) is true for any k such that $0 < k - i < n$, this implies that

$$J^{k+1} \Delta x^i = J^k \Delta x^i = J^{k-1} \Delta x^i = \ldots = J^{i+1} \Delta x^i . \tag{1.40}$$

Utilizing (1.36) once more, yields

$$J^{i+1} \Delta x^i = J^i \Delta x^i + f^{i+1} \tag{1.41}$$

or

$$J^{i+1} \Delta x^i = \Delta f^i$$

using equation (1.37). Hence, from (1.40) and (1.41),

$$J^k \Delta x^i = \Delta f^i, \quad 0 < k - i \leq n . \tag{1.42}$$

If $k > n$, one has

$$J^k \Delta x^i = \Delta f^i, \quad i = k-1, k-2, \ldots, k-n \tag{1.43}$$

This is exactly the same as equation (1.17) which forms the basis for the Secant Method. Thus, in n iterations, starting from initial estimates J^1 and x^1, Barnes' algorithm generates $n+1$ points $(x^1, f^1), (x^2, f^2), \ldots, (x^{n+1}, f^{n+1})$ and a matrix J^{n+1} which is the Jacobian of the linear system interpolating these $n+1$ points. Every subsequent iteration drops the earliest point and adds the most recent one, modifying the J matrix to correspond to this most recent set of $n+1$ points. In other words, after n iterations Barnes' algorithm is exactly the same as the Secant Method as previously derived.

Barnes' algorithm has the advantage of not requiring an initial set of $n+1$ points in order to begin the Secant Method. It can begin with only initial estimates J^1 and x^1 and form as a matter of course the $n+1$ points leading to the regular Secant Method. Of course, the better the initial estimate J^1, the faster will be the convergence in general. For a linear system the algorithm will theoretically converge to a solution of $f(x) = 0$ within $n+1$ iterations, regardless of the initial estimate J^1, since the matrix J^{n+1} corresponding to the $n+1$ points $(x^1, f^1), \ldots \ldots, (x^{n+1}, f^{n+1})$ completely defines the linear system; the next $(n+1)$ application of (1.37) (with $k = n+1$) yields the solution of the linear system $f(x) = 0$. Barnes' algorithm has the disadvantage of requiring the solution of n linear equations at each step (in solving (1.37) for Δx^k), which is almost equivalent to requiring the inversion of J^k at each step. It also, in its original form, represents a full-step Secant Method ($\alpha^k = 1$) so that its region of convergence is restricted compared to the reduced-step version.

Rosen's modification [18] of Barnes' algorithm eliminates these last two disadvantages. In addition to restructuring the algorithm as a reduced-step method, Rosen formulates the algorithm to update $(J^k)^{-1}$ directly rather than J^k. This is accomplished as follows. Let

$$\Delta x^k = -\alpha^k (J^k)^{-1} f^k \tag{1.44}$$

corresponding to a reduced-step method, and let the correction to J^k be represented as

$$J^{k+1} = J^k + D^k \tag{1.45}$$

where

$$D^k \triangleq \frac{(\Delta f^k - J^k \Delta x^k)(z^k)^T}{(z^k)^T \Delta x^k}$$

With z^k selected as in Barnes' algorithm, this yields the reduced-step Secant Method after n iterations, in the same way that Barnes' original algorithm produced the full-step version. However, instead of updating J^k as in (1.45), Rosen uses the Sherman-Morrison modification formula [12], [20] to convert (1.45) into an updating of $H^k \triangleq (J^k)^{-1}$. Thus, one obtains

$$H^{k+1} = H^k + \frac{(x^k - H^k f^k)(z^k)^T H^k}{(z^k)^T H^k \Delta f^k} \tag{1.46}$$

where

$$H^k \triangleq (J^k)^{-1}.$$

Thus H^{k+1} can be calculated fairly simply from a knowledge of H^k. Then Δx^k can be calculated directly from (1.44) without the need for solving a set of n linear equations or of completely recalculating the inverse of J^k at each step.

The Rosen-Barnes algorithms do not solve the problem of numerical instability inherent in the Secant Method whenever the matrix ΔX^{k+1} (or ΔF^{k+1}) becomes ill-conditioned as a result of Δx^k (or Δf^k) being "nearly" a linear combination of the previous $n - 1$ vectors. In the Barnes algorithm, the first condition manifests itself whenever

$$\frac{|(z^k)^T \Delta x^k|}{\|z^k\|_2 \ \|\Delta x^k\|_2} < \epsilon \tag{1.47}$$

where ϵ is some small positive number. This condition would raise doubts as to the validity of the results obtained in calculating J^{k+1} from equation (1.47). In Rosen's modification, the second condition would be evident whenever

$$\frac{|(z^k)^T H^k \Delta f^k|}{\|(H^k)^T z^k\|_2 \ \|\Delta f^k\|_2} < \epsilon' \tag{1.48}$$

where ϵ' is some small positive number. Again, this condition would detract from the reliable computation of H^{k+1} from (1.46). Barnes considers the problem and suggests rejecting the vector Δx^k whenever the condition of equation (1.47) occurs. He claims that a satisfactory alternative is to select a Δx^k for this step so as to result in z^k being parallel to Δx^k. Of course this cures the numerical problem associated with the reliable computation of J^{k+1}. However, it does little to insure fast convergence to a solution of $f(x) = 0$ since the direction of the selected Δx^k would be quite different from the presumably optimum direction suggested by the previous value of J^k.

Broyden's algorithms [4] form another quasi-Newton method, which can be related to the Rosen-Barnes version of the Secant Method. Broyden's Method 1 algorithm (which he found to be effective) is

$$x^{k+1} = x^k - \alpha^k H^k f^k \tag{1.49}$$

and

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)(\Delta x^k)^T H^k}{(\Delta x^k)^T H^k \Delta f^k} \tag{1.50}$$

where H^k is an approximation to $J^{-1}(x^k)$.

Note that this can be obtained from the Rosen-Barnes algorithm of equation (1.46) by letting $z^k = \Delta x^k$. Recall that in the latter method, z^k is selected to be orthogonal to the previous i vectors $\Delta x^{k-i}, \ldots, \Delta x^{k-2}, \Delta x^{k-1}$, where $i \leq n - 1$. Gram-Schmidt orthogonalization is used to accomplish this. Thus, starting with $k = 1$, one would form:

$$\begin{aligned} z^1 &= \Delta x^1 \\ z^2 &= \Delta x^2 - \frac{(z^1)^T \Delta x^2}{(z^1)^T z^1} z^1 \\ &\vdots \\ z^k &= \Delta x^k - \sum_{i=1}^{k-1} \frac{(z^i)^T \Delta x^k}{(z^i)^T z^i} z^i, \quad k \leq n. \end{aligned} \tag{1.51}$$

(For $k > n$, the above expressions are modified to form z^k orthogonal to the previous $(n - 1)$ vectors Δx^i.)

Hence, when only one vector (Δx^1) is available, the selection of $z^1 = \Delta x^1$ in the Rosen-Barnes algorithm is the same as would be made in Broyden's Method 1 algorithm. However, as additional vectors Δx^2, Δx^3, etc., become available, z^k

in the Rosen-Barnes algorithm are formed from (1.51), whereas Broyden's algorithm continues to use $z^k = \Delta x^k$. The choice of z^k as per equation (1.51) results in the Rosen-Barnes algorithm evolving into the Secant Method after n iterations, so that for a linear system the matrix H^{n+1} equals J^{-1}, the inverse Jacobian of the system, and the next $(n + 1)$ application of equation (1.44) results in the solution of the linear system $f(x) = 0$. There is no such assurance in Broyden's algorithm that the method will converge in $n + 1$ iterations for a linear system, or even converge at all, although subsequent work by Broyden [6] establishes conditions under which his method converges for linear systems. Numerical experience seems to indicate that Broyden's Method 1 algorithm is effective in a wide variety of problems, especially if the initial estimate H^1 of the inverse Jacobian is a good one, and/or the initial estimate x^1 is close to the solution. If this is not the case, the Rosen-Barnes algorithm is considered to be more effective.

Rosen indicates that on a particular test problem, better results were obtained using a z^k that was the average of that obtained from his algorithm with the z^k obtained from Broyden's algorithm, than were obtained using either procedure alone. This author is inclined to think that one of the reasons for this is that Broyden's algorithm inherently avoids the numerical problems evidenced by the occurrence of condition (1.47). In fact, for Broyden's algorithm,

$$\frac{|(z^k)^T \Delta x^k|}{\|z^k\|_2 \, \|\Delta x^k\|_2} = 1 \qquad (1.52)$$

since $z^k = \Delta x^k$. The average of the z^k's from the two methods will consistently avoid the condition of (1.47). At the same time, this average may still produce

a fair representation of the linear system corresponding to the previous points.

Broyden's Method 2 consists of equation (1.49) together with

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)\ (\Delta f^k)^T}{(\Delta f^k)^T \Delta f^k} \tag{1.53}$$

where H^k is again an approximation to the inverse Jacobian. Equation (1.53) can be obtained from (1.46) by letting

$$(z^k)^T H^k = (\Delta f^k)^T. \tag{1.54}$$

Equation (1.53) can also be derived from Zeleznik's form of a generalized quasi-Newton Method [26], namely equation (1.49) combined with

$$H^{k+1} = H^k + \frac{\Delta x^k\ (u^k)^T}{(u^k)^T \Delta f^k} - \frac{H^k \Delta f^k\ (v^k)^T}{(v^k)^T \Delta f^k}. \tag{1.55}$$

Letting $u^k = v^k = \Delta f^k$ results in (1.53). Zeleznik shows that for a linear system, in order for (1.55) to converge to the inverse Jacobian within n iterations, the following conditions must be satisfied.

$$(u^k)^T \Delta f^i = 0, \quad (v^k)^T \Delta f^i = 0. \tag{1.56}$$

$$0 < k - i < n$$

But, as Zeleznik points out, the choice of $u^k = v^k = \Delta f^k$ gives no assurance of meeting (1.56); hence there is no assurance that Broyden's Method 2 will converge even for a linear system. However under certain conditions, namely when $\|I - AH^{k_0}\|_2 < 1$ at some instant k_0, it is easy to show that for $\alpha^k \in (0, 1]$

Broyden's Method 2 converges to the solution $\bar{x}$ of the linear system $f(x) \triangleq A(x - \bar{x}) = 0$. Still, numerical experience has indicated that Broyden's Method 1 is effective while Method 2 is not. This author is of the opinion that the ineffectiveness of Method 2 is due largely to its inability to insure against singularity of H^k, a condition which would prevent convergence of the iteration except under specialized conditions. Broyden's Method 1 does prevent singularity of H^k as shown by the following.

Substituting Δx^k from (1.49) in (1.50) yields

$$H^{k+1} = H^k - \frac{H^k (\alpha^k f^k + \Delta f^k) (\Delta x^k)^T H^k}{(\Delta x^k)^T H^k \Delta f^k}$$

or

$$H^{k+1} = H^k (I - \sigma u v^T) \tag{1.57}$$

where

$$\sigma \triangleq \frac{1}{(\Delta x^k)^T H^k \Delta f^k}$$

$$u \triangleq \alpha^k f^k + \Delta f^k, \quad v \triangleq (H^k)^T \Delta x^k .$$

Assume that $\det H^k \neq 0$. Then $\det H^{k+1} \neq 0$ if and only if $\det (I - \sigma u v^T) \neq 0$. But

$$\det (I - \sigma u v^T) = 1 - \sigma v^T u \qquad \text{(see [12])}$$

$$= 1 - \frac{(\Delta x^k)^T H^k (\alpha^k f^k + \Delta f^k)}{(\Delta x^k)^T H^k \Delta f^k}$$

$$= 1 + \frac{(\Delta x^k)^T (\Delta x^k - H^k \Delta f^k)}{(\Delta x^k)^T H^k \Delta f^k} \qquad \text{(using (1.49))}$$

$$= \frac{(\Delta x^k)^T \Delta x^k}{(\Delta x^k)^T H^k \Delta f^k} \tag{1.58}$$

(assuming $(\Delta x^k)^T H^k \Delta f^k \neq 0$).

Thus Broyden's Method 1 insures that H^{k+1} is non-singular if H^k is non-singular.

It is also of interest to note that the Rosen-Barnes (Secant) Method will insure the non-singularity of H^{k+1} if condition (1.47) is avoided. A development analogous to the preceding yields

$$H^{k+1} = H^k (I - \sigma u v^T) \tag{1.59}$$

where

$$\sigma \triangleq \frac{1}{(z^k)^T H^k \Delta f^k}$$

$$u \triangleq \alpha^k f^k + \Delta f^k$$

$$v \triangleq (H^k)^T z^k$$

and

$$\det (I - \sigma u v^T) = \frac{(z^k)^T \Delta x^k}{(z^k)^T H^k \Delta f^k} \tag{1.60}$$

(assuming $(z^k)^T H^k \Delta f^k \neq 0$).

Thus the non-singularity of H^k implies the non-singularity of H^{k+1} if $(z^k)^T \Delta x^k \neq 0$.

Even though Broyden's Method 1 assures that H^{k+1} is non-singular, as does the Rosen-Barnes (Secant) Method with the additional assumption of $(z^k)^T \Delta x^k \neq 0$, neither method insures that H^k exists, since the denominator in each algorithm could approach zero. In Chapter II where some new variants of the Secant Method

are presented, it will be shown that the algorithm denoted as Algorithm II insures that H^k both exists and is non-singular.

There are many other methods for solving $f(x) = 0$ including two-point secant methods (as opposed to the $(n + 1)$-point secant method which is simply called the Secant Method in this report) and Steffensen's methods [16], [24]. Both of these classes of methods exhibit superlinear convergence (1.61 and 2 respectively). But, like the discrete Newton's Method, they generally require $n + 1$ function evaluations and a matrix inversion per iteration. The system $f(x) = 0$ can also be solved by any of a number of minimization techniques [16], since solution of $f(x) = 0$ is equivalent to minimizing $\varphi(x) \triangleq f^T(x)\, f(x)$ (to zero). A relatively simple minimization technique is the Fletcher-Powell-Davidon Method [7]. Finally, there are more complex methods, such as the Freudenstein-Roth Method [8], for solving $f(x) = 0$ in the difficult case where the Jacobian is singular at points in any region connecting the starting estimate with the solution. The other methods mentioned are ineffective in this case.

The remainder of this report will concern itself with some new variants of the Secant Method. These variants were developed to overcome the previously discussed limitations of existing versions of the Secant Method.

SECTIONS II

ALGORITHMS I AND II

2.1 INTRODUCTION

Algorithms I and II are variants of the Secant Method for solving $f(x) = 0$, n nonlinear equations in n unknowns. These algorithms are designed to avoid some numerical problems that can cause poor convergence in the Secant Method. The complex series of steps leading to their development will be omitted; rather the final form of the algorithms will be presented and analyzed.

Both algorithms follow the quasi-Newton iteration

$$x^{k+1} - x^k \triangleq \Delta x^k = -\alpha^k H^k f^k \tag{2.1}$$

$$f^k \triangleq f(x^k)$$

where α^k is either selected as unity (full step version) or else so as to insure that

$$\varphi^{k+1} < \varphi^k \tag{2.2}$$

where

$$\varphi^k \triangleq (f^k)^T f^k .$$

The matrix H^k, representing an estimate of the inverse Jacobian at x^k, is formed to satisfy

$$\Delta x_i^k = H^k \Delta f_i^k, \qquad i = 1, 2, \ldots, m_k \tag{2.3}$$

$$m_k \leq n .$$

The vectors Δx_i^k are certain previous difference vectors Δx^j, $j < k$, formed

via (2.1). The vectors Δf_i^k are the corresponding difference vectors Δf^j formed via the system function $f(x)$.

The set of m_k difference vectors Δx^k are not necessarily those corresponding to the last $m_k + 1$ iterates x^j, as would be the case in existing versions of the Secant Method. Rather they are formed from those most recent iterates x^j such that certain conditions are met. In Algorithm I, the governing condition is that the m_k vectors Δf_i^k are linearly independent. This insures the existence of H^k. In Algorithm II, the governing conditions are that the m_k vectors Δx_i^k are linearly independent as well as the corresponding set of m_k vectors Δf_i^k. This insures that H^k both exists and is non-singular.

The formation of H^k in each algorithm is described in the following sections.

2.2 ALGORITHM I

Algorithm I consists of the combination of Algorithm IB (when $m_k < n$) followed by Algorithm IA (when $m_k = n$). The matrix H^k in Algorithm I is updated as per

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)\,(b_j^k)^T}{(b_j^k)^T \Delta f^k} \tag{2.4}$$

provided that the vector b_j^k satisfies

$$\beta_j^k \triangleq \frac{|(b_j^k)^T \Delta f^k|}{\|b_j^k\|_2 \, \|\Delta f^k\|_2} > \rho_1 \tag{2.5}$$

where ρ_1 is a preselected constant in the interval (0, 1). Otherwise H^k is retained unchanged (i.e., $H^{k+1} \equiv H^k$). The formation of b_j^k in Algorithm I is accomplished as follows.

In Algorithm IA (used when $m_k = n$), b_j^k is formed from the j-th row of $(\Delta F^k)^{-1}$ where ΔF^k is the $n \times n$ matrix formed from n previous vectors Δf^k i.e.,

$$(b_j^k)^T = \text{j-th row of } (\Delta F^k)^{-1} \tag{2.6}$$

where

$$\Delta F^k \triangleq \left[\Delta f_1^k \; \Delta f_2^k \; \ldots \; \Delta f_n^k\right]_{n \times n}. \tag{2.7}$$

The integer j is selected as the first integer in the sequence j = 1, 2, etc., such that criterion (2.5) is satisfied. The new matrix ΔF^{k+1} is formed from ΔF^k by replacing Δf_j^k in the latter matrix with the current vector Δf^k and permuting the resultant matrix so as to place Δf^k in the last column, i.e.,

$$\Delta F^{k+1} = \left[\Delta F^k + (\Delta f^k - \Delta f_j^k)\,(e_j)^T\right] (P_j)^T \tag{2.8}$$

where

$$(P_j)^T \triangleq \left[e_1 \; \ldots \; e_{j-1} \; e_{j+1} \; \ldots \; e_n \; e_j\right]_{n \times n} \tag{2.9}$$

and

$$(e_i)^T \triangleq \left[0 \; \ldots \; 0 \;\; 1 \;\; 0 \; \ldots \; 0\right]_{1 \times n}.$$

(1 in i-th column)

The matrix ΔF^{k+1} is not explicitly needed in Algorithm IA but rather the matrix $(\Delta F^{k+1})^{-1}$. This matrix is readily calculated from

$$(\Delta F^{k+1})^{-1} = P_j \left[(\Delta F^k)^{-1} + \frac{\left(e_j - (\Delta F^k)^{-1} \Delta f^k\right) (b_j^k)^T}{(b_j^k)^T \Delta f^k}\right] \tag{2.10}$$

where P_j, e_j are as defined in (2.9).

$M(P_j)^T$ – col interchanges

$P_j M$ – row interchanges

Equations (2.4), (2.6), and (2.10), in conjunction with (2.1), constitute Algorithm IA in the case where a suitable vector b_j^k can be determined, i.e., one satisfying criterion (2.5). In this case, the vector pair $(\Delta x^k, \Delta f^k)$ becomes one of the n pairs satisfying (2.3) at $k + 1$. Satisfaction of criterion (2.5) has insured that Δf^k is "sufficiently" independent of the remaining $(n - 1)$ vectors retained in ΔF^{k+1}.

If a suitable vector b_j^k cannot be determined, then H^k and $(\Delta F^k)^{-1}$ are retained unchanged (i.e., $H^{k+1} \equiv H^k$ and $(\Delta F^{k+1})^{-1} \equiv (\Delta F^k)^{-1}$) and the iteration is continued with (2.1). In this case, the vector pair $(\Delta x^k, \Delta f^k)$ does not become one of the pairs satisfying (2.3) at $k + 1$.

In order to be initiated at some step k_0, Algorithm IA requires n previous vector pairs $(\Delta x_i^{k_0}, \Delta f_i^{k_0})$, $i = 1$ to n. Assuming the vectors $\Delta f_i^{k_0}$ are independent, the corresponding matrix H^{k_0} can be determined from (2.3). The use of Algorithm IA then insures that at every subsequent step $k > k_0$, the vectors Δf_i^k of equation (2.3) are independent. This insures that H^k is well defined (although it may become singular). If criterion (2.5) is ignored (i.e., by letting $j = 1$ in (2.6) at every step), the resultant algorithm (denoted as IA-S) is essentially the same as the quasi-Newton version of the Secant Method described e.g., in [16]. In this case, there is no assurance that H^k exists or is nonsingular.

When $m_k < n$, Algorithm IB is utilized to form b_j^k. This algorithm permits the iteration to start with only initial estimates x^1 of the solution and H^1 of the inverse Jacobian. After n or more iterations, Algorithm IB will have formed n vector pairs $(\Delta x_i^{k_0}, \Delta f_i^{k_0})$ in which the independence of the vectors $\Delta f_i^{k_0}$ is

assured. Thus Algorithm IA can take over at $k = k_0$ as previously discussed. Algorithm IB can be described as follows.

At any step k, where $m_k < n$, let matrices B_1^k and D_1^k be represented as

$$B_1^k \triangleq \begin{bmatrix} (b_1^k)^T \\ \vdots \\ (b_{m_k}^k)^T \end{bmatrix}_{m_k \times n}, \quad D_1^k \triangleq \left[\Delta f_1^k \ \ldots \ \Delta f_{m_k}^k\right]_{n \times m_k} \tag{2.11}$$

where

$$B_1^k \, D_1^k = I_{m_k}.$$

The vectors Δf_i^k, $i = 1$ to m_k, represent m_k independent vectors Δf^m, $m < k$, that were previously formed and retained in D_1^k. Form the vector b_j^k by some orthogonalization method (e.g., Gram-Schmidt orthogonalization) so as to be orthogonal to the m_k vectors Δf_i^k in D_1^k. If the vector b_j^k thus formed satisfies criterion (2.5), then form H^{k+1} as per (2.4) and form B_1^{k+1}, D_1^{k+1} as

$$B_1^{k+1} = \begin{bmatrix} B_1^k, 0 \\ \hline (b_j^k)^T \\ \hline (b_j^k)^T \Delta f^k \end{bmatrix}_{(m_k+1) \times n} \tag{2.12}$$

where

$$B_{1,0}^{k} \triangleq B_1^k - \frac{B_1^k \Delta f^k (b_j^k)^T}{(b_j^k)^T \Delta f^k}$$

and

$$D_1^{k+1} = \left[D_1^k \mid \Delta f^k \right]_{n \times (m_k + 1)} . \tag{2.13}$$

It is easy to show that the matrices B_1^{k+1} and D_1^{k+1} thus formed satisfy

$$B_1^{k+1} D_1^{k+1} = I_{m_{k+1}} \tag{2.14}$$

where

$$m_{k+1} = m_k + 1 .$$

Moreover, satisfaction of criterion (2.5) has insured that Δf^k is "sufficiently" independent of the remaining m_k vectors retained in D_1^{k+1}.

If the vector b_j^k as formed by orthogonalization fails to satisfy criterion (2.5), then $(b_j^k)^T$ is selected as the j-th row of B_1^k where j is the smallest integer beginning with 1 such that criterion (2.5) is satisfied. If such an integer exists ($1 \leq j \leq m_k$), then the associated vector b_j^k is used in equation (2.4) to form H^{k+1}, while B_1^{k+1} and D_1^{k+1} are calculated from

$$B_1^{k+1} = P_j \left[B_1^k + \frac{(e_j - B_1^k \Delta f^k) (b_j^k)^T}{(b_j^k)^T \Delta f^k} \right] \tag{2.15}$$

and

$$D_1^{k+1} = \left[D_1^k + (\Delta f^k - \Delta f_j^k) (e_j)^T \right] (P_j)^T \tag{2.16}$$

where

$$(P_j)^T \triangleq \left[e_1 \cdots e_{j-1}\ e_{j+1} \cdots e_{m_k}\ e_j\right]_{m_k \times m_k} \tag{2.17}$$

and

$$(e_i)^T \triangleq \left[0 \ldots 0 \quad 1 \quad 0 \ldots 0\right]_{1 \times m_k}.$$

(1 in i-th column)

It is easy to show that the matrices B_1^{k+1} and D_1^{k+1} thus formed satisfy

$$B_1^{k+1} D_1^{k+1} = I_{m_{k+1}} \tag{2.18}$$

where

$$m_{k+1} = m_k .$$

This alternate procedure for forming b_j^k implies that Δf^k is used to replace some earlier vector Δf_j^k in forming D_1^{k+1}. At the same time, the independence of the m_k vectors comprising D_1^{k+1} is assured.

If an acceptable vector b_j^k cannot be determined by this alternate procedure, then the procedure is as follows. Let

$$\begin{aligned} H^{k+1} &= H^k \\ B_1^{k+1} &= B_1^k \\ D_1^{k+1} &= D_1^k \end{aligned} \tag{2.19}$$

and continue the iteration with (2.1).

If and when a point $k = p$ is reached such that $m_{k+1} = n$, then the matrix B_1^{p+1} will satisfy

$$B_I^{p+1} = (D_I^{p+1})^{-1} \equiv (\Delta F^{p+1})^{-1} \tag{2.20}$$

where ΔF^{p+1} contains n independent vectors Δf^m. At $k = p + 1 \equiv k_0$, the iteration can be continued with Algorithm IA (which is essentially the same as the alternate procedure just described for forming b_j^k, with $m_k = n$).

To start Algorithm IB (at $k = 1$), simply treat B_I^1 as an empty row vector and D_I^1 as an empty column vector, i.e., $m_1 = 0$. Then let $b_j^1 = \Delta f^1$ (as would be the case for example using Gram-Schmidt orthogonalization), and this obviously satisfies criterion (2.5).

It still remains to be shown that the matrix H^k, as formed by Algorithm I (i.e., Algorithm IB followed by IA), satisfies equation (2.3). This will be shown next.

Consider the formation of b_j^k in Algorithm IB at each step k at which an acceptable b_j^k can be formed. If b_j^k is formed by the orthogonalization procedure, then it is orthogonal to the previous vectors Δf_i^k comprising D_I^k. Expressed another way, b_j^k is orthogonal to all vectors retained in D_I^{k+1} except Δf^k. If b_j^k is formed by the alternate procedure (as the j-th row of B_I^k), then again it is orthogonal to all vectors retained in D_I^{k+1} except Δf^k (by virtue of $B_I^k D_I^k = I_{m_k}$). In either case then, one has

$$(b_j^k)^T \Delta f_i^{k+1} = 0, \quad i = 1, 2, \ldots, (m_{k+1} - 1) \tag{2.21}$$

since Algorithm IB places Δf^k in the m_{k+1} column of D_I^{k+1}.

The vectors Δf_i^{k+1} of equation (2.21) were formed at some previous instants k_i, that is

$$\Delta f_i^{k+1} \equiv \Delta f^{k_i}, \quad i = 1, 2, \ldots, (m_{k+1} - 1) \tag{2.22}$$

where

$$1 \leq k_1 < k_2 < \ldots < k_{(m_{k+1}-1)} < k.$$

Then equation (2.21) can be expressed as

$$b_j^k \Delta f^{k_i} = 0, \qquad i = 1, 2, \ldots, (m_{k+1} - 1). \tag{2.23}$$

From equation (2.4) of Algorithm IB, one obtains

$$H^{k+1} \Delta f^k = \Delta x^k \tag{2.24}$$

for every value of k from 1 to p at which an acceptable vector b_j^k is formed. (The step k = p in Algorithm IB is defined by $m_{k+1} = n$.) Also from (2.4),

$$H^{k+1} \Delta f^{k_i} = H^k \Delta f^{k_i}, \qquad i = 1, 2, \ldots, (m_{k+1} - 1) \tag{2.25}$$

using (2.23). Equation (2.25) implies that

$$H^{k+1} \Delta f^{k_i} = H^k \Delta f^{k_i} = H^{k-1} \Delta f^{k_i} = \ldots = H^{k_i+1} \Delta f^{k_i} \tag{2.26}$$

or

$$H^{k+1} \Delta f^{k_i} = \Delta x^{k_i}, \qquad i = 1, 2, \ldots, m_{k+1} - 1 \tag{2.27}$$

using (2.24).

Using the identity of (2.22), equation (2.27) becomes

$$H^{k+1} \Delta f^{k+1} = \Delta x_i^{k+1}, \qquad i = 1, 2, \ldots, m_{k+1} - 1. \tag{2.28}$$

Since it has been assumed that an acceptable vector b_j^k has been formed at step k, Algorithm IB has placed Δf^k in the m_{k+1} column of B_1^{k+1}, i.e., $\Delta f^k \equiv f^{k+1}_{m_{k+1}}$. Hence, from (2.24),

$$H^{k+1} \Delta f^{k+1}_{m_{k+1}} = \Delta x^{k+1}_{m_{k+1}}. \tag{2.29}$$

Combining (2.28) and (2.29) yields

$$H^{k+1} \Delta f_i^{k+1} = \Delta x_i^{k+1}, \quad i = 1, 2, \ldots, m_{k+1} \tag{2.30}$$

for every step k at which an acceptable vector b_j^k has been formed.

At any step k at which an acceptable vector b_j^k cannot be determined, Algorithm IB utilizes equation (2.19) which implies that (2.30) is trivially true at this step as well. Hence, equation (2.30) is true at every step k, which verifies that equation (2.3) holds for Algorithm IB. The proof for Algorithm IA follows directly, since Algorithm IA can be derived from the alternate procedure of Algorithm IB by letting $m_k = n$.

Note that Algorithm IB has the same weakness as IA in that there is no assurance that the vectors Δx^i, corresponding to the retained independent vectors Δf^i, are themselves independent. Thus even though H^k is always defined, it may become singular. Once this happens, say at $k = q$, Algorithm I will not be able to converge to a solution $\bar{x}$ of $f(x) = 0$ unless the vector $x^q - \bar{x}$ happens to be in the subspace spanned by the columns of H^q. This can be shown by the following, which is similar to the development in [5].

In both Algorithm IA and IB, H^{k+1} can be written as

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)(b_j^k)^T}{(b_j^k)^T \Delta f^k}$$

$$= H^k \left[I - \frac{(\alpha^k f^k + \Delta f^k)(b_j^k)^T}{(b_j^k)^T \Delta f^k} \right] \quad \text{(using (2.1))}$$

or

$$H^{k+1} = H^k L^k \tag{2.31}$$

where

$$L^k \triangleq I - \frac{(\alpha^k f^k + \Delta f^k)(b_j^k)^T}{(b_j^k)^T \Delta f^k} .$$

Then

$$H^{k+1} = H^q L^q L^{q+1} \ldots L^k \tag{2.32}$$

where H^q is assumed to be singular. Hence from (2.1),

$$\Delta x^{k+1} = - \alpha^{k+1} H^{k+1} f^{k+1}$$

or

$$\Delta x^{k+1} = H^q v \tag{2.33}$$

where

$$v \triangleq - \alpha^{k+1} L^q L^{q+1} \ldots L^k f^{k+1} .$$

Thus Δx^{k+1}, which is some linear combination of the columns of H^q, is confined to the subspace spanned by these columns, and this is true for every $k \geq q$. It is also true obviously for Δx^q so that the iteration cannot possibly converge to a solution $\bar{x}$ unless $(x_q - \bar{x})$ happens to lie in this same subspace.

If criterion (2.5) is ignored, the resultant algorithm (denoted as IB-S) is similar to Zeleznik's construction [26]. The independence (or lack of independence) of the vectors Δf^i is ignored in this construction. Thus H^k may not be defined (in addition to having the risk of becoming singular as in Algorithm IB), causing obvious numerical problems. The combination of Algorithm IB-S followed by IA-S forms a logical Secant Method algorithm, denoted as I-S, in which the independence of the vectors Δf^i is consistently ignored.

Algorithm II, to be discussed next, assures that the retained vectors Δx^i are independent, as well as the corresponding vectors Δf^i. Thus the matrix H^k remains well-defined and non-singular.

2.3 ALGORITHM II

Algorithm II consists of the combination of Algorithm IIB (when $m_k < n$) followed by Algorithm IIA (when $m_k = n$). The matrix H^k in Algorithm II is updated as per

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)\ (a_j^k)^T H^k}{(a_j^k)^T H^k \Delta f^k} \tag{2.34}$$

provided that the vector a_j^k satisfies

$$\gamma_j^k \triangleq \frac{|(a_j^k)^T \Delta x^k|}{\|a_j^k\|_2\ \|\Delta x^k\|_2} > \rho_1 \tag{2.35}$$

and

$$\beta_j^k \triangleq \frac{|(a_j^k)^T H^k \Delta f^k|}{\|(H^k)^T a_j^k\|_2\ \|\Delta f^k\|_2} > \rho_2 \tag{2.36}$$

where ρ_1, ρ_2 are preselected constants in the interval (0, 1). Otherwise H^k is retained unchanged (i.e., $H^{k+1} \equiv H^k$). The formation of a_j^k in Algorithm II is accomplished as follows.

In Algorithm IIA (used when $m_k = n$), a_j^k is formed from the j-th row of $(\Delta X^k)^{-1}$ where ΔX^k is the $n \times n$ matrix formed from the vectors Δx_i^k, i.e.,

$$(a_j^k)^T = \text{j-th row of } (\Delta X^k)^{-1} \tag{2.37}$$

where

$$\Delta X^k \triangleq \left[\Delta x_1^k \; \Delta x_2^k \; \ldots \; \Delta x_n^k\right]_{n \times n}. \tag{2.38}$$

The integer j is selected as the first integer in the sequence j = 1, 2, etc., such that criteria (2.35) and (2.36) are both satisfied. The new matrix ΔX^{k+1} is formed from ΔX^k by replacing Δx_j^k in the latter matrix with the current vector Δx^k and permitting the resultant matrix so as to place Δx^k in the last column, i.e.,

$$\Delta X^{k+1} = \left[\Delta X^k + (\Delta x^k - \Delta x_j^k)\,(e_j)^T\right](P_j)^T \tag{2.39}$$

where P_j and e_j are as defined in (2.9).

The matrix ΔX^{k+1} is not explicitly needed in Algorithm IIA but rather the matrix $(\Delta X^{k+1})^{-1}$. This matrix is readily calculated from

$$(\Delta X^{k+1})^{-1} = P_j\left[(\Delta X^{k+1})^{-1} + \frac{\left(e_j - (\Delta X^k)^{-1}\,\Delta x^k\right)(a_j^k)^T}{(a_j^k)^T\,\Delta x^k}\right] \tag{2.40}$$

where P_j and e_j are as defined in (2.9).

Equations (2.34), (2.37), and (2.40), in conjunction with (2.1), constitute Algorithm IIA in the case where a suitable vector a_j^k can be determined, i.e., one satisfying criteria (2.35) and (2.36). In this case, the vector pair $(\Delta x^k, \Delta f^k)$ becomes one of the pairs satisfying (2.3) at k + 1. If a suitable vector a_j^k cannot be determined, then H^k and $(\Delta X^k)^{-1}$ are retained unchanged (i.e., $H^{k+1} \equiv H^k$ and $(\Delta X^{k+1})^{-1} \equiv (\Delta X^k)^{-1}$) and the iteration is continued with (2.1). In the latter case, the vector pair $(\Delta x^k, \Delta f^k)$ does not satisfy (2.3) at k + 1.

A simpler algorithm naturally results if the criteria for independence, (2.35) and (2.36), are ignored. This algorithm, denoted as IIA-S, is obtained by letting $j = 1$ in equation (2.37). It is another Secant Method algorithm similar to Algorithm IA-S. Again, the existence and non-singularity of H^k are not assured in this algorithm.

As was the case with Algorithm IA, Algorithm IIA requires an initial set of n vector pairs $(\Delta x_i^{k_0}, \Delta f_i^{k_0})$, $i = 1$ to n. Again, the vectors $\Delta f_i^{k_0}$ must be independent in order to define H^{k_0} from (2.3). In addition, the vectors $\Delta x_i^{k_0}$ must be independent in order that $(\Delta X^{k_0})^{-1}$ exist. The use of Algorithm IIA then insures that at every subsequent step $k > k_0$, the vectors Δf_i^k are independent as well as the vectors Δx_i^k. This insures that H^k is both defined and non-singular.

When $m_k < n$, Algorithm IIB is utilized to form a_j^k. This algorithm permits the iteration to start with only initial estimates x^1 of the solution and H^1 of the inverse Jacobian. After n or more iterations, Algorithm IIB will have formed n vector pairs $(\Delta x_i^{k_0}, \Delta f_i^{k_0})$ in which the independence of the vectors $\Delta x_i^{k_0}$ is assured as well as that of the vectors $\Delta f_i^{k_0}$. Thus Algorithm IIA can take over at $k = k_0$. Algorithm IIB can be described as follows.

At any step k, where $m_k < n$, let matrices A_1^k and C_1^k be represented as

$$A_1^k \triangleq \begin{bmatrix} (a_1^k)^T \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ (a_{m_k}^k)^T \end{bmatrix}_{m_k \times n} \qquad C_1^k \triangleq \left[\Delta x_1^k \ \ldots\ldots \ \Delta x_{m_k}^k\right]_{n \times m_k} \tag{2.41}$$

where

$$A_1^k \, C_1^k = I_{m_k}.$$

The vectors Δx_i^k, $i = 1$ to m_k, represent $m_k < n$ independent vectors Δx^m, $m < k$, that were previously formed (by this algorithm) and retained in C_1^k. Form the current vector Δx^k in the usual way, that is from equation (2.1). Then form the vector a_j^k by some orthogonalization method (e.g., Gram-Schmidt orthogonalization) so as to be orthogonal to the m_k vectors Δx_i^k in C_1^k.

If the vector a_j^k thus formed satisfies criteria (2.35) and (2.36), then form H^{k+1} as per (2.34) and form A_1^{k+1}, D_1^{k+1} as

$$A_1^{k+1} = \left[\begin{array}{c} A_{1,0}^k \\ \hline \dfrac{(a_j^k)^T}{(a_j^k)^T \, \Delta x^k} \end{array}\right]_{(m_k+1)\times n} \tag{2.42}$$

where

$$A_{1,0}^k \triangleq A_1^k - \frac{A_1^k \, \Delta x^k \, (a_j^k)^T}{(a_j^k)^T \, \Delta x^k}$$

and

$$C_1^{k+1} = \left[C_1^k \mid \Delta x^k\right]_{n\times(m_k+1)} \tag{2.43}$$

It is easy to show that the matrices A_1^{k+1} and C_1^{k+1} thus formed satisfy

$$A_1^{k+1} \, C_1^{k+1} = I_{m_{k+1}} \tag{2.44}$$

where

$$m_{k+1} = m_k + 1.$$

If the vector a_j^k as formed by orthogonalization fails to satisfy criteria (2.35) and (2.36), then select a_j^k as

$$(a_j^k)^T \triangleq (e_j)^T A_1^k = \text{j-th row of } A_1^k \tag{2.45}$$

where j is the smallest integer ($1 \leq j \leq m_k$) such that criteria (2.35) and (2.36) are both satisfied. Then, using this j and associated vector a_j^k, form H^{k+1} as per (2.34) and form A_1^{k+1}, C_1^{k+1} as

$$A_1^{k+1} = P_j \left[A_1^k + \frac{(e_j - A_1^k \Delta x^k)(a_j^k)^T}{(a_j^k)^T \Delta x^k} \right] \tag{2.46}$$

and

$$C_1^{k+1} = \left[C_1^k + (\Delta x^k - \Delta x_j^k)(e_j)^T \right] (P_j)^T \tag{2.47}$$

where P_j and e_j are as defined in (2.17). It is easy to show that the matrices A_1^{k+1} and C_1^{k+1} thus formed satisfy

$$A_1^{k+1} C_1^{k+1} = I_{m_{k+1}} \tag{2.48}$$

where

$$m_{k+1} = m_k .$$

If an acceptable a_j^k (satisfying criteria (2.35) and (2.36)) cannot be found by this alternate procedure, then let

$$H^{k+1} = H^k$$

$$A_I^{k+1} = A_I^k \qquad (2.49)$$

$$C_I^{k+1} = C_I^k$$

and continue the iteration with equation (2.1).

If and when a point $k = p$ is reached such that $m_{k+1} = n$, then one has

$$A_I^{p+1} = (C_I^{p+1})^{-1} \equiv (\Delta X^{p+1})^{-1} \qquad (2.50)$$

where ΔX^{p+1} contains n independent vectors Δx^i. At $k = p + 1 \equiv k_0$, the iteration can be continued with Algorithm IIA (which is essentially the same as the alternate procedure just described for forming a_j^k, with $m_k = n$).

To start Algorithm IIB (at $k = 1$), one treats A_I^1 and D_I^1 as empty, i.e., $m_1 = 0$. Then try $a_j^1 = \Delta x^1$ (as would be the case using Gram-Schmidt orthogonalization), and this obviously satisfies criterion (2.34). Usually, criterion (2.35) would also be satisfied by this choice. If not, H^2 would be set equal to H^1 (as per (2.49)) and the iteration continued (A_I^2 and C_I^2 would remain empty in this case).

Note that satisfaction of criterion (2.35) in Algorithm IIB implies that Δx^k does not lie almost entirely in the subspace of the previous $(m_{k+1} - 1)$ vectors Δx^i that are to be retained in C_I^{k+1}. This follows from the fact that a_j^k, whether formed by orthogonalization or by the alternate procedure of equation (2.45), is orthogonal to these $(m_{k+1} - 1)$ vectors. Hence if a_j^k is not orthogonal to Δx^k, then Δx^k cannot be a linear combination of these $(m_{k+1} - 1)$ vectors. Similarly satisfaction of criterion (2.36) insures that Δf^k does not lie almost entirely in the subspace formed by the previous $(m_{k+1} - 1)$ vectors Δf^i corresponding to

the above Δx^i. This follows from the fact that the vector $(H^k)^T a_j^k$ is orthogonal to these $(m_{k+1} - 1)$ vectors Δf^i as shown by the following.

$$(a_j^k)^T H^k \Delta f^i = (a_j^k)^T \Delta x^i \quad \text{(using (2.3))} \qquad (2.51)$$

$$= 0.$$

A simpler algorithm than IIB naturally results if the criteria for independence (2.35) and (2.36) are ignored. This algorithm, denoted as Algorithm IIB-S, consists of equations (2.1), (2.34), and (2.42). Like (Secant) Algorithm IIA-S, it would encounter numerical problems whenever the vector Δx^k (Δf^k) failed to be sufficiently independent of the previous vectors Δx^i (Δf^i), $0 < k - i < n$. The combination of Algorithm IIB-S followed by IIA-A is denoted as Algorithm II-S. It is essentially the same as the Rosen-Barnes (Secant) Method [1], [18], although for $k > n$ it involves fewer computations per iteration than the latter method.

The proof that H^k, as formed by Algorithm II, satisfies equation (2.3) is directly analogous to that given for Algorithm I and so will not be repeated.

SECTION III

LOCAL CONVERGENCE OF ALGORITHMS I AND II

First the conditions insuring local convergence of the regular Secant Method will be reviewed. Assume that

a. $f(x)$ has continuous second derivatives in some neighborhood of a solution $\bar{x}$ to $f(x) = 0$.

b. The Jacobian matrix $J(x) \triangleq \left(\frac{\partial f_i}{\partial x_j}\right)$ is non-singular at the solution.

Let

$$d^k \triangleq \det\left(\frac{\delta x^{k-n}}{\|\delta x^{k-n}\|_2} \cdots \frac{\delta x^{k-1}}{\|\delta x^{k-1}\|_2}\right) \tag{3.1}$$

where

$$\delta x^i \triangleq x^k - x^i, \quad i = k-1, \ldots, k-n.$$

Then Bittner [2] proved that local convergence of the Secant Method is assured when the following additional condition is met.

Given ω such that $0 < \omega < 1$,

$$|d^k| > \omega, \text{ for every } k > n. \tag{3.2}$$

where d^k is defined by (3.1).

Tornheim [21] proved that under these same conditions, local convergence is superlinear of order at least that of the positive root of

$$\lambda^{n+1} = \lambda^n + 1. \tag{3.3}$$

Some representative values of this root are 1.62 for $n = 1$, 1.18 for $n = 10$, and 1.03 for $n = 100$.

Ortega and Rheinboldt [16] have generalized these results and relaxed requirement (a) to $f(x)$ having Lipschitz continuous first derivatives in some neighborhood of the solution. The matrix δX^k (as defined by (1.22)) is said to be a member of a class of uniformly non-singular matrices if condition (3.2) is satisfied. Assuming this to be the case, it is shown that J^k (as given by (1.24) or (1.19)) is a strongly consistent approximation to $J(x^k)$ in that there exists constants c and $r > 0$ such that

$$\| J^k - J(x^k) \| \leq c \| \delta X^k \| \tag{3.4}$$

when

$$\| \delta X^k \| < r .$$

Under some additional assumptions that the iterates x^k given by

$$x^{k+1} = x^k - (J^k)^{-1} f^k \tag{3.5}$$

are well defined and that $\| \delta X^k \|$ remains suitably small, it is shown that the iteration of (3.5) converges superlinearly to a solution $\bar{x}$ of $f(x) = 0$, the order of convergence being at least equal to the positive root of (3.3). As a corollary, $J^k \rightarrow J(\bar{x})$ as $k \rightarrow \infty$.

The determination of whether (3.2) is satisfied at every step $k > n$ is generally impractical since it involves a considerable amount of computations. In Algorithms I and II, comparatively simple criteria are utilized to determine the relative independence of the vectors comprising ΔX^k and ΔF^k. If these criteria

are initially satisfied at every step, these algorithms follow the same steps as the Secant Method. Under these conditions and with some additional assumptions, it will be shown that Algorithms I and II satisfy condition (3.2) and hence possess superlinear convergence. When these criteria are not initially satisfied at every step, then as previously discussed these algorithms depart from the regular Secant Method. In so doing, the algorithms at least permit linear convergence as will be discussed later.

First it will be shown that the condition of (3.2), under one additional assumption, is satisfied by the condition

$$|\Delta^k| > \omega_1, \quad \text{for all } k > n \tag{3.6}$$

where

$$\omega_1 \in (0, 1)$$

is given,

$$\Delta^k \triangleq \det \left(\frac{\Delta x^{k-n}}{\|\Delta x^{k-n}\|_2} \quad \cdots \quad \frac{\Delta x^{k-1}}{\|\Delta x^{k-1}\|_2} \right)$$

and

$$\Delta x^i \triangleq x^{i+1} - x^i .$$

Note that

$$\det (\delta x^{k-n} \ \ldots \ \delta x^{k-1}) = \det (\Delta x^{k-n} \ \ldots \ \Delta x^{k-1}) \tag{3.7}$$

since simple operations that leave the value of a determinant unchanged will convert one form into the other.

Thus

$$d^k = \frac{\det\left(\delta x^{k-n} \ \ldots\ \delta x^{k-1}\right)}{\|\delta x^{k-n}\|_2 \ \ldots\ \|\delta x^{k-1}\|_2} \qquad \text{(from (3.1))}$$

$$= \frac{\det\left(\Delta x^{k-n} \ \ldots\ \Delta x^{k-1}\right)}{\|\delta x^{k-n}\|_2 \ \ldots\ \|\delta x^{k-1}\|_2} \qquad \text{(using (3.7))}$$

$$= \Delta^k \frac{\|\Delta x^{k-n}\|_2 \ \ldots\ \|\Delta x^{k-1}\|_2}{\|\delta x^{k-n}\|_2 \ \ldots\ \|\delta x^{k-1}\|_2} \qquad (3.8)$$

using the definition of Δ^k in (3.6).

The left hand determinant in (3.7) represents the volume of the n-dimensional parallelopiped with a vertex at the origin and n sides δx^i, while the right hand determinant represents the equal volume parallelopiped having a vertex at the origin and n sides Δx^i, $i = k - n, k - n + 1, \ldots, k - 1$. Assume that for every $k > n$, the ratio of the smallest side in the second representation to the largest side in the first representation is bounded below by a constant c_1, i.e.,

$$\frac{\|\Delta x^i\|_2}{\|\delta x^j\|_2} \geq c_1, \qquad i, j, = k - n, \quad k - n + 1, \quad \ldots, \quad k - 1$$

$$k > n \qquad (3.9)$$

where

$$c_1 \in (0, 1]$$

The bound c_1 cannot exceed unity since $\Delta x^{k-1} \equiv \delta x^{k-1}$. Then if (3.6) is satisfied, (3.8) and (3.9) imply that

$$|d^k| > \omega_1 (c_1)^{n-1} \triangleq \omega, \quad \omega \in (0, 1) \qquad (3.10)$$

showing that (3.2) is satisfied.

It will now be shown that satisfaction of criteria (2.35) and (2.36) in Algorithm IIB via the orthogonalization procedure at every step $k \leq n$, and satisfaction of these criteria in Algorithm IIA with $j = 1$ at every step $k > n$, insure that condition (3.6) is satisfied. This then would insure the superlinear convergence of Algorithm II as well as the convergence of H^k to $J(\bar{x})$ where $\bar{x}$ is the solution to $f(x) = 0$. Note that satisfaction of the aforementioned criteria in the manner described means that Algorithm II would be following the same steps as Algorithm II-S which is equivalent to the Secant Method.

At any instant k, $k \leq n$, Algorithm IIB-S has formed the vectors $\Delta x^1, \Delta x^2, \ldots, \Delta x^{k-1}, \Delta x^k$, as well as the vectors a_j^i, $i = 1$ to k, via an orthogonalization procedure. The Gramian of the k vectors Δx^i, $i = 1$ to k, is given by

$$G(\Delta x^1, \ldots, \Delta x^k) \triangleq \begin{vmatrix} (\Delta x^1, \Delta x^1) & \ldots & (\Delta x^1, \Delta x^k) \\ \vdots & & \\ (\Delta x^k, \Delta x^1) & \ldots & (\Delta x^k, \Delta x^k) \end{vmatrix} \tag{3.11}$$

As shown by Gantmacher [10], this Gramian can be expressed as

$$G(\Delta x^1, \ldots, \Delta x^k) = G(\Delta x^1, \ldots, \Delta x^{k-1}) (h_k)^2 \tag{3.12}$$

where h_k is the Euclidean length of the component of Δx^k that is orthogonal to the subspace spanned by the vectors $\Delta x^1, \ldots, \Delta x^{k-1}$.

Now the formation of a_j^k via Algorithm IIB-S has insured that a_j^k is orthogonal to this same subspace. Further, it is assumed that criterion (2.35) is satisfied at every k, $k \leq n$, which implies that

$$\frac{|(a_j^k, \Delta x^k)|}{\|a_j^k\|_2} > \rho_1 \|\Delta x^k\|_2. \tag{3.13}$$

Suppose one constructs an orthonormal basis for the orthogonal complement to this subspace in which $a_j^k / \|a_j^k\|_2$ forms one basis vector. Then an interpretation of (3.13) is that the vector Δx^k has a component greater than $\rho_1 \|\Delta x^k\|_2$ along this unit vector, the latter being orthogonal to the aforementioned subspace; hence the component of Δx^k orthogonal to this subspace must be greater than $\rho_1 \|\Delta x^k\|_2$, i.e.,

$$h_k > \rho_1 \|\Delta x^k\|_2. \tag{3.14}$$

Then from (3.12),

$$G(\Delta x^1, \ldots, \Delta x^k) > (\rho_1)^2 (\|\Delta x^k\|_2)^2 G(\Delta x^1, \ldots, \Delta x^{k-1}). \tag{3.15}$$

Since (3.15) is true for every k, $k \leq n$, one has

$$G(\Delta x^1, \Delta x^2) > (\rho_1)^2 (\|\Delta x^2\|_2)^2 G(\Delta x^1)$$

$$= (\rho_1)^2 (\|\Delta x^1\|_2)^2 (\|\Delta x^2\|_2)^2$$

$$G(\Delta x^1, \Delta x^2, \Delta x^3) > (\rho_1)^2 (\|\Delta x^3\|_2)^2 G(\Delta x^1, \Delta x^2)$$

$$> (\rho_1)^4 (\|\Delta x^1\|_2)^2 (\|\Delta x^2\|_2)^2 (\|\Delta x^3\|_2)^2$$

and finally

$$G(\Delta x^1, \ldots, \Delta x^n) > (\rho_1)^{2(n-1)} (\|\Delta x^1\|_2)^2 \ldots (\|\Delta x^n\|_2)^2. \tag{3.16}$$

Since

$$\Delta X^{n+1} \triangleq [\Delta x^1 \; \Delta x^2 \; \ldots \; \Delta x^n] \tag{3.17}$$

equation (3.11) implies that

$$G(\Delta x^1, \ldots, \Delta x^n) = \det \left[(\Delta X^{n+1})^T \Delta X^{n+1} \right]$$

$$= \left[\det (\Delta X^{n+1})\right]^2 . \tag{3.18}$$

Hence from (3.16) and (3.18),

$$|\det (\Delta X^{n+1})| > (\rho_1)^{n-1} \|\Delta x^1\|_2 \ldots \|\Delta x^n\|_2 \tag{3.19}$$

$$\left| \det \left(\frac{\Delta x^1}{\|\Delta x^1\|_2} \ldots . \frac{\Delta x^n}{\|\Delta x^n\|_2} \right) \right| > (\rho_1)^{n-1}$$

or

$$|\Delta^{n+1}| > (\rho_1)^{n-1} \triangleq \omega_1 . \tag{3.20}$$

Thus criterion (3.7) is satisfied for $k = n + 1$. This result will now be extended to all k, $k > n$.

Similar to (3.12), one can express $G(\Delta x^{k-n+1}, \ldots, \Delta x^k)$ as

$$G(\Delta x^{k-n+1}, \ldots, \Delta x^k) = G(\Delta x^{k-n+1}, \ldots, \Delta x^{k-1}) \; (h_k)^2 \tag{3.21}$$

where h_k is the length of the component of Δx^k that is orthogonal to the subspace formed by the $n - 1$ vectors $\Delta x^{k-n+1}, \ldots, \Delta x^{k-1}$. Algorithm IIA-S forms a_j^k so as to be orthogonal to this subspace; hence satisfaction of criterion (2.35) produces the same inequality for h_k as before, namely

$$h_k > \rho_1 \|\Delta x^k\|_2 . \tag{3.22}$$

Hence,

$$G(\Delta x^{k-n+1}, \ldots, \Delta x^k) > (\rho_1)^2 \left(\|\Delta x^k\|_2\right)^2 G(\Delta x^{k-n+1}, \ldots, \Delta x^{k-1}). \tag{3.23}$$

Now a_j^{k-1}, whether formed by Algorithm IIA-S or IIB-S, is orthogonal to the subspace formed by Δx^{k-n+1} to Δx^{k-2} (it may be orthogonal to a larger subspace), so that one obtains

$$G(\Delta x^{k-n+1}, \ldots, \Delta x^{k-1}) > (\rho_1)^2 \left(\|\Delta x^{k-1}\|_2\right)^2 G(\Delta x^{k-n+1}, \ldots, \Delta x^{k-2}). \tag{3.24}$$

Continuing in this fashion,

$$\begin{aligned} G(\Delta x^{k-n+1}, \ldots, \Delta x^{k-2}) &> (\rho_1)^2 \left(\|\Delta x^{k-2}\|_2\right)^2 G(\Delta x^{k-n+1}, \ldots, \Delta x^{k-3}) \\ &\vdots \\ G(\Delta x^{k-n+1}, \Delta x^{k-n+2}) &> (\rho_1)^2 \left(\|\Delta x^{k-n+2}\|_2\right)^2 G(\Delta x^{k-n+1}) \\ &= (\rho_1)^2 \left(\|\Delta x^{k-n+2}\|_2\right)^2 \left(\|\Delta x^{k-n+1}\|_2\right)^2 . \end{aligned} \tag{3.25}$$

Combining the above inequalities yields

$$G(\Delta x^{k-n+1}, \ldots, \Delta x^k) > (\rho_1)^{2(n-1)} \left(\|\Delta x^{k-n+1}\|_2\right)^2 \ldots \left(\|\Delta x^k\|_2\right)^2 . \tag{3.26}$$

Since

$$G(\Delta x^{k-n+1}, \ldots, \Delta x^k) = \left[\det (\Delta X^{k+1})\right]^2 \tag{3.27}$$

one obtains

$$\left| \det \left(\frac{\Delta x^{k-n+1}}{\|\Delta x^{k-n+1}\|_2} \cdots \frac{\Delta x^{k}}{\|\Delta x^{k}\|_2} \right) \right| > (\rho_1)^{n-1}, \quad k > n \tag{3.28}$$

$$\left| \det \left(\frac{\Delta x^{k-n}}{\|\Delta x^{k-n}\|_2} \cdots \frac{\Delta x^{k-1}}{\|\Delta x^{k-1}\|_2} \right) \right| > (\rho_1)^{n-1}, \quad k > n+1$$

or

$$|\Delta^k| > (\rho_1)^{n-1} \triangleq \omega_1, \quad k > n+1. \tag{3.29}$$

Combining (3.29) with (3.20) shows that condition (3.7) is satisfied for all k, $k > n$, when the criteria of Algorithms IIB and IIA are initially satisfied as described. Then under these conditions and the previous assumptions, Algorithm II possesses local superlinear convergence of order at least that of the positive root of (3.3).

The extension of these results to Algorithm I is fairly simple, although some additional assumptions are required as will be shown. First assume that criterion (2.5) in Algorithm IB is satisfied at each step $k \leq n$ via the orthogonalization procedure and that this criterion in Algorithm IA is satisfied at each step $k > n$ by $j = 1$. Then in an analogous manner to the foregoing one obtains

$$\left| \det \left(\frac{\Delta f^{k-n}}{\|\Delta f^{k-n}\|_2} \cdots \frac{\Delta f^{k-1}}{\|\Delta f^{k-1}\|_2} \right) \right| > (\rho_1)^{n-1}, \quad k > n \tag{3.30}$$

where ρ_1 is a preselected number in the interval (0, 1). The matrices ΔX^k and ΔF^k satisfy

$$\Delta X^k = H^k \Delta F^k \tag{3.31}$$

so that

$$\det(\Delta X^k) = \det(H^k) \det(\Delta F^k). \tag{3.32}$$

Then from (3.32),

$$\left| \det \left(\frac{\Delta x^{k-n}}{\|\Delta x^{k-n}\|_2} \cdots \frac{\Delta x^{k-1}}{\|\Delta x^{k-1}\|_2} \right) \right|$$

$$= |\det(H^k)| \frac{\|\Delta f^{k-n}\|_2 \cdots \|\Delta f^{k-1}\|_2}{\|\Delta x^{k-n}\|_2 \cdots \|\Delta x^{k-1}\|_2} \left| \det \left(\frac{\Delta f^{k-n}}{\|\Delta f^{k-n}\|_2} \cdots \frac{\Delta f^{k-1}}{\|\Delta f^{k-1}\|_2} \right) \right|$$

$$> \frac{|\det(H^k)|}{\left(\|H^k\|_2\right)^n} (\rho_1)^{n-1} \qquad \text{(using (3.30) and (3.31))}. \tag{3.33}$$

Assume that H^k is non-singular. This is not assured in Algorithm I even though the satisfaction of criterian (2.5) assures the non-singularity of ΔF^k. However, as previously discussed, if the vectors Δf^i comprising ΔF^k are formed from iterates close enough to a solution $\bar{x}$, then the independence of the vectors Δf^i assures the independence of the corresponding vectors Δx^i (assuming $J(\bar{x})$ is non-singular) and hence assures the non-singularity of H^k. Assume further that

$$\frac{|\det(H^k)|}{\left(\|H^k\|_2\right)^n} \geq c_2, \quad k > n \tag{3.34}$$

where $c_2 \in (0,1]$.

Note that c_2 cannot exceed unity since for any matrix A,

$$|\det A| \leq \left(\rho(A)\right)^n \tag{3.35}$$

and

$$\|A\| \geq \rho(A) \text{ for any matrix norm}$$

where $\rho(A) \triangleq$ spectral radius of A.

Then (3.35) can be written as

$$\left| \det \left(\frac{\Delta x^{k-n}}{\|\Delta x^{k-n}\|_2} \cdots \frac{\Delta x^{k-1}}{\|\Delta x^{k-1}\|_2} \right) \right| > c_2 \, (\rho_1)^{n-1} \triangleq \omega_1 \tag{3.36}$$

or

$$|\Delta^k| > \omega_1, \ \omega_1 \in (0, 1), \ k > n$$

so that condition (3.7) is satisfied. Thus under the foregoing assumptions, Algorithm I has the same superlinear convergence if the associated test criteria are initially satisfied at every step.

If the test criteria in Algorithm I or II are not initially satisfied in the manner described, then superlinear convergence is not assured. However, linear convergence is still possible if the matrix H^k satisfies

$$\|I - H^k J(\bar{x})\| < 1 \tag{3.37}$$

where $J(\bar{x})$ is the Jacobian matrix at the solution $\bar{x}$. This is shown by the following. With $\alpha^k = 1$, the next iterate using (2.5) is

$$x^{k+1} = x^k - H^k f^k$$

or

$$x^{k+1} - \bar{x} = x^k - \bar{x} - H^k \left[J(\bar{x}) \, (x^k - \bar{x}) + v_\epsilon \right] \tag{3.38}$$

where v_ϵ accounts for the higher order terms in the Taylor expansion of $f^k \triangleq f(x^k)$ about the solution $\bar{x}$. Then

$$x^{k+1} - \bar{x} = \left[I - H^k J(\bar{x}) \right] (x^k - \bar{x}) - H^k v_\epsilon$$

and

$$\| x^{k+1} - \bar{x} \| \leq \| I - H^k J(\bar{x}) \| \| x^k - \bar{x} \| + \| H^k \| \| v_\epsilon \| \tag{3.39}$$

where

$$\| v_\epsilon \| = 0(\| x^k - \bar{x} \|^2) .$$

Thus local linear convergence is assured if (3.37) is satisfied.

A prerequisite to satisfying (3.37) is that H^k be non-singular (it has already been assumed that $J(\bar{x})$ is non-singular). For if H^k is singular, then so is $H^k J(\bar{x})$, and $I - H^k J(\bar{x})$ would have a characteristic root equal to unity since

$$\det \left(\lambda I - \left(I - H^k J(\bar{x}) \right) \right) = 0 \tag{3.40}$$

when

$$\lambda = 1 .$$

Then the spectral radius of $I - H^k J(\bar{x})$ would be at least equal to unity implying that

$$\| I - H^k J(\bar{x}) \| \geq 1 \tag{3.41}$$

for any matrix norm. Algorithm II inherently insures that H^k is non-singular. However Algorithm I does not, although as previously discussed, H^k will be non-singular in Algorithm I as well if the vectors Δf^i comprising ΔF^k are formed from iterates close enough to a solution $\bar{x}$.

Even though H^k is non-singular in Algorithm II, and assumed to be so in Algorithm I, there still is no assurance that (3.37) will be satisfied. Heuristically though, one would expect this to be the case if for example the initial estimate H^1 is a good approximation to $J(\bar{x})$ and if the problem is not severely nonlinear. In any event, the conditions under which Algorithms I and II depart from the Secant Method are those tending to cause poor convergence of the latter. The alternatives followed by Algorithms I and II in these circumstances at least retain the potential for linear convergence.

SECTION IV

NUMERICAL EXPERIMENTS

In Section II, variants of the Secant Method are presented consisting of Algorithms I and II. It is claimed that these algorithms are an improvement, in terms of convergence, over corresponding existing versions of the Secant Method. In order to test this hypothesis, a series of randomly generated test problems have been run, each via several different methods. The problems were run on an APL/360 computing system having a precision of approximately 16 decimal digits. Programming and other details are given in Appendix 2.

The type of problem investigated is the same as that considered by Fletcher and Powell [7] and by Rosen [18], namely

$$f_i(x) = E_i - \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j) = 0 \qquad (4.1)$$

$$i = 1, 2, \ldots n .$$

This can be expressed in vector form as

$$f(x) = E - (A \sin x + B \cos x) = 0 \qquad (4.2)$$

where the notation sin x (or cos x) refers to the n-vector whose i-th component is sin x_i (or cos x_i), where x_i is the i-th component of x. The components of the n × n matrices A and B are random integers in the range -100 to +100. The n-vector E is obtained from

$$E = A \sin \bar{x} + B \cos \bar{x} \qquad (4.3)$$

where $\bar{x}$ is an n-vector each of whose components is a random number between

42

$-\pi$ and $+\pi$. Thus $\bar{x}$ is automatically one solution to (4.2). There are generally multiple solutions to (4.2) as noted in [18]. For each problem, the starting estimate x^1 is generated as

$$x^1 = \bar{x} + 0.1\delta \tag{4.4}$$

where δ is also an n-vector whose components are each random numbers between $-\pi$ and $+\pi$. In a sense then, the starting estimate x^1 is close to a solution $\bar{x}$ although there may be other solutions in the vicinity.

Five independent problems were generated for each value of n (see Appendix 2 for details). The values of n considered were 2, 5, 10, and 15 for a total of 20 problems. Each problem was run with eight different algorithms. Besides Algorithms I and II, these included a discrete Newton's Method, Secant Algorithms I-S and II-S, Broyden's Methods 1 and 2, and a combination of Broyden's Method 1 and Algorithm II-S. Newton's Method was selected to serve as a standard against which the efficacy of the other algorithms could be judged. Secant Algorithm II-S is equivalent to the Rosen-Barnes version of the Secant Method, although as noted previously in this report, it involves fewer computations per iteration when $k > n$. Secant Algorithm I-S is equivalent to a form suggested by Zeleznik [26], or to the quasi-Newton form of the Secant Method described in [16]. Thus Algorithms I-S and II-S provided a basis for judging the claim that Algorithms I and II are improved variants over corresponding versions of the Secant Method. Broyden's Methods 1 and 2 were included because of their simplicity and because of the initial similarity of Method 1 to Algorithm II (or II-S) and of Method 2 to Algorithm I (or I-S). Finally a combination of Broyden's Method 1 and Algorithm II-S was included because of Rosen's determination

that such a combination generally yielded better results in his series of problems than either method alone.

When Algorithm I or II was utilized, it was necessary to specify a value of ρ_1. Each time these Algorithms were employed, two or more runs were made with different values of ρ_1, in order to obtain some measure of the effect of ρ_1 on convergence. Concurrently, an attempt was made to develop a rule of thumb for selecting a nominal value of ρ_1 that would yield good results. Section III provided some insight on the effect that ρ_1 has in establishing a lower bound for the ill-conditioning of Δx^k. This together with initial experimentation resulted in the selection of some "nominal" values of ρ_1 as $n \times 10^{-3}$ and $2n \times 10^{-3}$. The results using these values of ρ_1 are listed in subsequent tables.

When Algorithm II was used, it was also necessary to specify a value of ρ_2. In order to avoid a further increase in the number of runs, which was already considerable, only one value of ρ_2 was specified for each value of ρ_1. The specification was arbitrarily set at $\rho_2 = 0.1\,\rho_1$. The parameter ρ_2 is used in criterion (2.36) to insure the independence of the vectors Δf^i that are retained for subsequent use. However when $m_k < n$, criterion (2.36) is a sufficient but not necessary condition for assuring this independence. In other words, criterion (2.36) could fail even though the vectors Δf^i were sufficiently independent. In order to reduce the probability of this unnecessary rejection of a_j^k as formed by orthogonalization, it was deemed desirable to set ρ_2 lower than ρ_1. At the same time a value of $\rho_2 > 0$ was still considered necessary to avoid $\|H^k\| \to \infty$.

In the combination of Broyden's Method 1 and Algorithm II-S, the vector z^k in

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)\ (z^k)^T H^k}{(z^k)^T H^k \Delta f^k} \qquad (4.5)$$

is formed as an average of the vectors in the individual methods. When each of the above methods is used by itself, only the direction of z^k is important since a constant multiplier has no effect. But when an average of two different vectors is performed, the relative scaling assigned to each is important since it affects the direction of the resultant average. Since this scaling was not explicitly described in Rosen's paper [18], the author decided on the magnitude scaling inherent in each method as it is described elsewhere in this report. Thus in Broyden's Method 1, z^k is set equal to Δx^k, while in Algorithm I-S $\bar{z}^k$ is the component of Δx^k that is orthogonal to the subspace formed by the previous vectors Δx^i, $0 < k - i < n$. In this scheme, the ratio of the magnitude of z^k as produced by Algorithm II-S to that produced by Broyden's Method 1 is a maximum of unity when Δx^k is orthogonal to the aforementioned subspace and becomes zero when Δx^k lies in this subspace.

All of the algorithms that have been listed could be utilized either in a full step version ($\alpha^k = 1$) or in a reduced-step version (α^k selected to minimize or reduce φ^{k+1}). Only the full step version was used in these experiments for the following reasons. Methods that select α^k on the basis of minimizing φ^{k+1} are somewhat inefficient in terms of function evaluations. Secondly, there are a great many norm-reducing schemes. To utilize even just a few of these on top of the run permutations already considered would result in a morass of data in which the basic purpose of the investigations would be masked. Certainly the relative efficiency of norm-reducing methods is peripheral to this investigation. The basic purpose of this experimentation is to test the convergence characteristics of Algorithms I

and II near a solution and to compare them with those of Secant Algorithms I-S and II-S among others. This comparison is facilitated by the use of a single generally accepted method of selecting the step size and certainly $\alpha^k = 1$ qualifies.

Each algorithm considered requires an initial estimate H^1 of the inverse Jacobian as well as an initial estimate x^1 of the solution. Algorithms I and II, as well as I-S and II-S, could function with any initial estimate H^1 provided a norm-reducing method for selecting the step size were included. These algorithms, for a linear system, generate the system inverse Jacobian after at most n iterations. For a non-linear system, these algorithms would generate an approximation to the inverse Jacobian at x^1 within n iterations, provided the step size were not so great as to drive the iterate to a locale with a markedly different Jacobian. A norm-reducing constraint would generally prevent such deviations. However, norm-reducing schemes were not included in this numerical experimentation for reasons previously discussed. Secondly, starting with an arbitrary H^1 (e.g., $H^1 = I$) provides little if any saving in computations since the n iterations required to develop a good approximation to the inverse Jacobian at x^1 are comparable to the computational effort in directly calculating an approximate $J^{-1}(x)$. For these reasons, the initial estimate H^1 in this investigation was uniformly generated by computing a discretized approximation to $J(x^1)$ and obtaining its inverse. This selection also has the effect of not penalizing Broyden's Method unnecessarily; the latter is at its best when $H^1 \approx J^{-1}(x^1)$.

The use of the full-step version for all of the algorithms, as opposed to a reduced-step norm-reducing version, raises the possibility that in some cases the iteration will fail to converge to a solution for any of the methods. To accommodate this possibility the following strategy was adopted. If all of the quasi-Newton

algorithms (i.e., all except the discrete Newton's Method) failed to converge to a solution, the starting estimate x^1 given by (4.4) would be changed to

$$x^1 = \bar{x} + \left(\frac{1}{2}\right)^q \times 0.1\delta \tag{4.6}$$

where q assumed the values 1, 2, etc., until at least one of the quasi-Newton methods converged to a solution.

The effect of (4.6) is to bring the starting estimate x^1 closer and closer to a solution $\bar{x}$ so as to increase the probability of convergence. The reason for adopting this strategy is that the main purpose of the investigation is to compare the local convergence of Algorithms I and II with that of the other quasi-Newton methods, especially that of Secant Algorithms I-S and II-S. The latter methods will at times encounter difficulties after reaching points very close to a solution; it is primarily this type of problem that Algorithms I and II were designed to overcome. An evaluation of the region of convergence of the various methods is beyond the scope of this investigation. In the actual numerical work, equation (4.6) had to be utilized only once, in problem 55, when all of the quasi-Newton methods failed to converge. The use of $q = 1$ in (4.6) resulted in all of the methods converging to a solution in this problem.

In judging the effectiveness of the various algorithms in this particular series of problems, the primary criterion used is the number of vector function evaluations required by the algorithm to converge to a solution. This criterion, which has been used by many authors, is simple to apply and provides a fair representation of total computational effort especially if the function is at all complicated. In the discrete Newton's Method, $n + 1$ function evaluations are required at each step except the last, where only one evaluation is required. The other

(quasi-Newton) methods require but one function evaluation per step except for the first step ($k = 1$) which requires $n + 1$ evaluations to establish the initial estimate H^1. Thus denoting the final iterate index as k_f (at which φ^{k_f} is less than the convergence criterion), the total number of function evaluations is given by

$$\text{No. of f evaluations} = (k_f - 1)(n + 1) + 1 \tag{4.7}$$

$$= k_f (n + 1) - n$$

for the discrete Newton's Method, and by

$$\text{No. of f evaluations} = (k_f - 1) + (n + 1) \tag{4.8}$$

$$= k_f + n$$

for the quasi-Newton methods. The convergence criterion in all cases was arbitrarily set at

$$\| f^k \|_2 < 10^{-6}$$

or

$$\varphi^k \triangleq (f^k)^T f^k < 10^{-12}. \tag{4.9}$$

A secondary criterion used to judge the effectiveness of the various algorithms is the accuracy of the final matrix H^{k_f} generated by each algorithm, since the accuracy of the estimated inverse Jacobian at the solution is of importance in some applications. Also the accuracy of H^{k_f} is a distinguishing characteristic, especially in those cases where convergence is poor. The standard used for comparison is the inverse of the estimated Jacobian at the solution as determined by the same discrete process used in the discrete Newton's Method. Denoting the

latter matrix as $J^{-1}(\bar{x})$, the error criterion used is the maximum absolute component value of the error matrix $(H^{kf} - J^{-1}(\bar{x}))$.

For purposes of comparison, it is convenient to divide the quasi-Newton algorithms into two groups. Group 1 consists of those algorithms in which the updating of H^k has the form

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)(z^k)^T H^k}{(z^k)^T H^k \Delta f^k} \tag{4.10}$$

This group then includes the Broyden 1 algorithm, Algorithms II and II-S, and the combination of Broyden 1 and Algorithm II-S. Similarly, Group 2 algorithms are classified by the form

$$H^{k+1} = H^k + \frac{(\Delta x^k - H^k \Delta f^k)(z^k)^T}{(z^k)^T \Delta f^k} \tag{4.11}$$

Group 2 then consists of the Broyden 2 algorithm and Algorithms I and I-S.

The results of the computer runs in this series of problems are summarized in Tables 4.1 through 4.4. When convergence occurred, as determined by $\|f\|_2 < 10^{-6}$, the distance of the final iterate x^{kf} from a true solution was always less than 10^{-6} (as measured in the infinity norm). Usually, the closeness of x^{kf} to a true solution was in the order of 10^{-8}. An asterisk next to the number of function evaluations for convergence indicates convergence to a solution other than the nominal solution $\bar{x}$ originally set up. In all cases, these other solutions were within 2π of $\bar{x}$ and usually much closer. The term "Diverge" is used to indicate when an algorithm was divergent, the criterion being divergence to a distance (again in the infinity norm) of 100 or more from $\bar{x}$. The term "Osc" is used to denote

Table 4.1

Convergence of n = 2 Problems

1st entry: No. of f evaluations to reduce $\|f\|_2$ below 10^{-6}.

2nd entry: Max. absolute component error in final estimate of inverse Jacobian at solution.

Algorithm	Prob. #21	Prob. #22	Prob. #23	Prob. #24	Prob. #25
Discrete Newton	13 $< 10^{-6}$	10 $< 10^{-6}$	46* $< 10^{-6}$	10 $< 10^{-6}$	25 3.6×10^{-6}
Broyden 1	9 .00103	7 .00240	13 .0130	8 .00039	13 .0575
Alg II-S	9 6.9×10^{-6}	8 2.5×10^{-5}	12 .00758	8 2.2×10^{-6}	13 12.6
Br 1 - Alg II-S	9 7.3×10^{-6}	8 4.4×10^{-5}	13 .00625	8 1.4×10^{-6}	13 12.5
Alg II ($\rho_1 = n \times 10^{-3}$)	9 6.9×10^{-6}	8 2.5×10^{-5}	12 .0363	8 2.2×10^{-6}	13 .815
Alg II ($\rho_1 = 2n \times 10^{-3}$)	9 6.9×10^{-6}	8 .0345	12 .0363	8 2.2×10^{-6}	13 .815
Broyden 2	11 .0168	7 .00067	Diverge	8 .00020	14 .0735
Alg I-S	10 7.8×10^{-6}	8 2.6×10^{-5}	Diverge	8 2.4×10^{-6}	13 287
Alg I ($\rho_1 = n \times 10^{-3}$)	10 7.8×10^{-6}	8 2.6×10^{-5}	Diverge	8 2.4×10^{-6}	13 7.48
Alg I ($\rho_1 = 2n \times 10^{-3}$)	10 7.8×10^{-6}	8 2.6×10^{-5}	Diverge	8 2.4×10^{-6}	13 7.48

*Converged to solution other than nominal one.

Table 4.2

Convergence of n = 5 Problems

1st entry: No. of f evaluations to reduce $\|f\|_2$ below 10^{-6}.

2nd entry: Max. absolute component error in final estimate of inverse Jacobian at solution.

Algorithm	Prob. #51	Prob. #52	Prob. #53	Prob. #54	Prob. #55
Discrete Newton	61* $< 10^{-6}$	25 $< 10^{-6}$	61 $< 10^{-6}$	31 $< 10^{-6}$	25* $< 10^{-6}$
Broyden 1	23 .138	13 .00350	17 .0165	14* .0160	17* .0503
Alg II-S	26* 65200	14 .0437	18 .00081	14* .0215	16* .0316
Br 1 - Alg II-S	23 1990	14 .0328	16 .0198	15* .00529	17* .200
Alg II ($\rho_1 = n \times 10^{-3}$)	22* 1.40	13 .00034	16 .00369	14* .00935	16* .00093
Alg II ($\rho_1 = 2n \times 10^{-3}$)	21* .0882	13 .00034	16 .00969	14* .0208	17* 1.30
Broyden 2	Diverge	13 .00975	28 .0930	16* .0576	19* .106
Alg I-S	Osc 56+ 4430	13 .00114	37 .0252	15* .0651	Osc 56+ 86.1
Alg I ($\rho_1 = n \times 10^{-3}$)	30* 1.66	13 .00114	27 1.82	15* .0651	43* 1.39
Alg I ($\rho_1 = 2n \times 10^{-3}$)	30* 1.77	13 .00114	25 .00120	15* .0651	40* 5.15

*Converged to solution other than nominal one.

Table 4.3

Convergence of n = 10 Problems

1st entry: No. of f evaluations to reduce $\|f\|_2$ below 10^{-6}.

2nd entry: Max. absolute component error in final estimate of inverse Jacobian at solution.

Algorithm	Prob.#101	Prob.#102	Prob.#103	Prob.#104	Prob.#105
Discrete Newton	45 $< 10^{-6}$	Diverge	45 $< 10^{-6}$	89* 1.2×10^{-6}	56 $< 10^{-6}$
Broyden 1	21 .00963	36 .169	22 .00235	27* 1.37	24 .00682
Alg II-S	21 .00489	94 30.9	21 .00348	31* 259	23 .0192
Br 1 - Alg II-S	21 .00928	62 10.2	22 .00867	29* 294	24 2.09
Alg II ($\rho_1 = n \times 10^{-3}$)	20 .00163	37 2.07	21 .00348	26* .147	22 .0215
Alg II ($\rho_1 = 2n \times 10^{-3}$)	20 .00163	40 .284	21 .00360	26* .152	23 .0371
Broyden 2	23 .0105	Diverge	23 .00470	31* .231	26 .0436
Alg I-S	21 .0791	Diverge	21 .00355	29* 83.2	28 .410
Alg I ($\rho_1 = n \times 10^{-3}$)	21 .0791	Diverge	21 .00355	28* 3.77	27 .437
Alg I ($\rho_1 = 2n \times 10^{-3}$)	21 .00547	Diverge	21 .00213	28* 2.51	27 .414

*Converged to solution other than nominal one.

Table 4.4

Convergence of n = 15 Problems

1st entry: No. of f evaluations to reduce $\|f\|_2$ below 10^{-6}.

2nd entry: Max. absolute component error in final estimate of inverse Jacobian at solution.

Algorithm	Prob.#151	Prob.#152	Prob.#153	Prob.#154	Prob.#155
Discrete Newton	81 $< 10^{-6}$	129* $< 10^{-6}$	Diverge	161 $< 10^{-6}$	97 $< 10^{-6}$
Broyden 1	28 .00449	Diverge	35 .0153	36* .366	31 .0127
Alg II-S	28 .0193	Osc 91+ 1.55	33 .157	34 2.64	30 .0719
Br 1 - Alg II-S	28 .00323	Diverge	36 .116	35 406	30 .00996
Alg II ($\rho_1 = n \times 10^{-3}$)	28 .0193	51* .538	33 .0313	32 .133	30 .0289
Alg II ($\rho_1 = 2n \times 10^{-3}$)	28 .00584	41* 1.19	33 .0262	32 .118	30 .00922
Broyden 2	32 .0300	Diverge	Diverge	35 .846	32 .0330
Alg I-S	30 .152	Osc 91+ 2.70	61* 11.2	34 12.5	31 .155
Alg I ($\rho_1 = n \times 10^{-3}$)	30 .0272	Osc 91+ .595	63 .411	34 2.08	31 .152
Alg I ($\rho_1 = 2n \times 10^{-3}$)	31 .0285	56* 1.74	106* .753	34 1.58	31 .211

*Converged to solution other than nominal one.

convergence to the vicinity of a solution but lack of convergence to a solution within the stated (large) number of function evaluations.

For each problem, the algorithm(s) providing convergence with the least number of function evaluations is (are) identified by the underlining of this number. Similarly, the quasi-Newton algorithm(s) yielding the smallest component error in $H^{k\ell}$ is (are) identified by the underlining of this error. In many problems more than one algorithm provided convergence with the minimum number of evaluations. On occasion, more than one quasi-Newton algorithm yielded both the minimum number of evaluations and the minimum component error in $H^{k\ell}$. When this occurred, or when more than one algorithm yielded the same two numbers regardless of whether they were minima or not, the reason was that these algorithms were following exactly the same path. Thus for example in Problem #21, both cases of Algorithms II as well as Algorithm II-S followed identical paths since the criteria associated with Algorithm II were initially satisfied at every step. In Problem #52, both cases of Algorithm II happened to follow the same step-by-step procedure (but not the same as in Algorithm II-S) even though the associated ρ_1 criteria differed by a factor of two.

The algorithms in Tables 4.1 through 4.4 are arranged in the order of discrete Newton's Method, then Group 1 algorithms and lastly Group 2 algorithms. A quick perusal of these tables reveals that the discrete Newton was generally the least efficient in terms of function evaluations. This was pretty much expected in view of the required $n + 1$ function evaluations per iteration. The discrete Newton converged to a solution in every problem except two (Problems #102 and #153). That divergence occurred twice is not surprising. The multiplicity

of roots in a small region implies that the Jacobian is singular at many points in this region. A chance landing close to one of these points could throw the next iterate far out with no chance of recovery. This apparently occurred in Problems #102 and #153. When convergence occurred, the discrete Newton always provided the most accurate estimate of H^{k_f} (generally within 10^{-6} of $J(\bar{x})$), as would be expected.

A further perusal of these tables reveals that as a group, the Group 1 algorithms were generally more effective than Group 2, this behavior becoming more pronounced as n increased. Moreover this is the case in an algorithm by algorithm comparison of corresponding algorithms in each group. Thus Broyden 1 was generally more effective than Broyden 2, Algorithm II more effective than Algorithm I, and Algorithm II-S more effective than Algorithm I-S. In the case of Broyden 1 and Algorithm II, their greater effectiveness (over the corresponding versions in Group 2) may be due in part to their inherently avoiding H^k becoming singular, as previously shown in this report. In the case of Algorithm II-S vs. I-S, neither algorithm insures against the singularity of H^k. However when $k < n$, H^k is less likely to become singular in Algorithm II-S than in I-S. This is shown by the following.

Assume that H^k is non-singular. Then as shown in Section I, H^{k+1} in Algorithm II-S will become singular (for $\Delta x^k \neq 0$) if and only if

$$(z^k)^T \Delta x^k = 0. \tag{4.12}$$

This occurs only when Δx^k lies in the subspace of the previous vectors Δx^i, $0 < k - i < n$. In the case of Algorithm I-S, a similar development shows that H^{k+1} becomes singular (for $\Delta x^k \neq 0$) if and only if

$$(z^k)^T J^k \Delta x^k = 0. \tag{4.13}$$

Recall that in Algorithm I-S, z^k is formed to be orthogonal to the previous vectors Δf^i, $0 < k - i < n$, or

$$(z^k)^T \Delta f^i = 0, \quad 0 < k - i < n. \tag{4.14}$$

But Δf^i satisfies

$$\Delta f^i = J^k \Delta x^i, \quad 0 < k - i < n \tag{4.15}$$

which when substituted in (4.14) yields

$$(z^k)^T J^k \Delta x^i = 0, \quad 0 < k - i < n.$$

Thus the vector $(J^k)^T z^k$ in Algorithm I-S is always orthogonal to the vectors Δx^i, $0 < k - i < n$. If Δx^k lies in the subspace formed by these vectors, then $(J^k)^T z^k$ must be orthogonal to Δx^k as well, showing that (4.13) holds. In addition, when $k < n$, it is possible for the vector $(J^k)^T z^k$ to be orthogonal to Δx^k even if the latter lies outside the aforementioned subspace since this subspace would be of dimension less than $n - 1$. If one assumes that the vectors Δx^k in both algorithms are more or less randomly oriented, then there is greater likelihood of H^k becoming singular in Algorithm I-S than in II-S. This may be a factor in the better performance of Algorithm II-S compared to Algorithm I-S.

To facilitate the comparison of performance of algorithms within a group, Tables 4.5 and 4.6 have been prepared. For each set of problems, these tables list the number of times an algorithm reached convergence in the fewest evaluations, the number of times an algorithm yielded the most accurate matrix H^{kf}, and the number of times the algorithm failed to converge. The first two

Table 4.5

Comparative Performance of Group 1 Algorithms

1st entry: No. of times algorithm had fewest f evaluations in a series of problems.

2nd entry: No. of times algorithm had most accurate estimate of inverse Jacobian at solution.

3rd entry: No. of times algorithm failed to converge (in parentheses).

Algorithm	n = 2 Problems	n = 5 Problems	n = 10 Problems	n = 15 Problems
Alg II	4	4	4	4
($\rho_1 = n \times 10^{-3}$)	2	2	2	1
	(0)	(0)	(0)	(0)
Alg II	4	4	3	5
($\rho_1 = 2n \times 10^{-3}$)	1	2	1	2
	(0)	(0)	(0)	(0)
Alg II-S	4	2	1	3
	2	1	0	0
	(0)	(0)	(0)	(1)
Broyden 1	4	2	1	1
	1	0	3	1
	(0)	(0)	(0)	(1)
Br 1 - Alg II-S	3	1	0	2
	2	1	0	1
	(0)	(0)	(0)	(1)

Table 4.6

Comparative Performance of Group 2 Algorithms

1st entry: No. of times algorithm had fewest f evaluations in a series of problems.

2nd entry: No. of times algorithm had most accurate estimate of inverse Jacobian at solution.

3rd entry: No. of times algorithm failed to converge (in parentheses).

Algorithm	n = 2 Problems	n = 5 Problems	n = 10 Problems	n = 15 Problems
Alg I	3	3	3	3
($\rho_1 = n \times 10^{-3}$)	3	2	0	2
	(1)	(0)	(1)	(1)
Alg I	3	4	3	3
($\rho_1 = 2n \times 10^{-3}$)	3	2	2	1
	(1)	(0)	(1)	(0)
Alg I-S	3	2	2	4
	3	1	0	0
	(1)	(2)	(1)	(1)
Broyden 2	2	2	1	0
	1	1	2	2
	(1)	(1)	(1)	(2)

comparison numbers are based on the best performance in each problem of the algorithms in the group rather than of all algorithms tested.

Table 4.5 reveals that either case of Algorithm II was generally more effective than any of the other algorithms in the group. Besides yielding the fewest function evaluations in general, Algorithm II converged in all of the 20 random problems under test whereas the other algorithms in the group each failed to converge once (in Problem #152). In addition, the convergence of Algorithm II-S was very poor in one other problem (Problem #102) where it required 94 evaluations to reach a solution. The combination of Broyden 1 and Algorithm II-S did not perform any better on the average than the individual algorithms comprising it. In the aforementioned problem (#102) when Algorithm II-S had poor convergence, this combination did much better than Algorithm II-S but still much worse than Broyden 1. It would seem that Broyden 1 is to be preferred over this combination algorithm.

Table 4.6 reveals that Algorithm I with $\rho_1 = 2n \times 10^{-3}$ was generally more effective than any of the other algorithms in the group. Although it failed to converge twice (in Problems #23 and #102), none of the other algorithms in the group converged in these two problems. In addition, Algorithm I-S and Broyden 2 failed to converge three additional times. Algorithm I with $\rho_1 = n \times 10^{-3}$ was not quite as effective as with $\rho_1 = 2n \times 10^{-3}$ although it still performed better on the average than either Algorithm I-S or Broyden 2. Both cases of Algorithm I generally yielded smaller errors in H^{k_f} than did Algorithm I-S. In addition Algorithm I-S often had some strikingly large errors in H^{k_f} even when it converged (see tabulation for Problems #25, #104, #153, and #154). Broyden 2,

when it converged, yielded a fairly accurate H^{k_f}, almost as good on the average as did Algorithm I.

It is of interest to examine the cases where Algorithm I-S or II-S converged to a region near a solution but failed to converge to a solution (denoted by "Osc" in Tables 4.1 through 4.4), and also those cases where either of these algorithms required an excessively large number of evaluations to reach a solution. In problem #51, Algorithm I-S reached the vicinity of a solution within about 25 function evaluations but oscillated erratically thereafter. When terminated, the error in $(H^k - J^{-1}(\bar{x}))$ for either of the known solutions in the vicinity was an astounding 4430. This number also approximated the magnitude of the largest component of H^k since the components of $J^{-1}(\bar{x})$ were less than one in all of the problems. At the same time, the determinant of H^k measured only -1.3×10^{-9}, indicating a poorly conditioned, near singular matrix. The relative independence of the vectors Δx^i comprising the matrix ΔX^k, as measured by program COND described in Appendix 2, was only 6.0×10^{-13}. Thus the poor conditioning of this set of vectors was apparently the major factor in causing an erroneous, near-singular matrix H^k, and in preventing convergence.

In problem #102, Algorithm II-S required 94 function evaluations to reach the expected solution, with the error in H^{k_f} being quite large (30.9). Again H^{k_f} was near singular (det $H^{k_f} = 7.8 \times 10^{-20}$) and the conditioning of the vectors Δx^i comprising ΔX^{k_f} was extremely poor (6.0×10^{-32}). Periodic measurements during this run of the error in H^k, the determinant of H^k, and the conditioning of ΔX^k showed the same pattern. This same pattern was also in evidence in Problem #55 where Algorithm I-S exhibited continual oscillatory behavior in the vicinity of a solution, and in Problem #152 where Algorithms I-S and II-S both

exhibited erratic oscillation in the vicinity of a solution and failure to converge. Thus the inability of Algorithms I-S and II-S to prevent near singular matrices H^k appears to be a factor in delaying or preventing convergence at times.

Algorithm I exhibited a related type of behavior in two cases, Problems #152 and #153. In Problem #152, Algorithm I with $\rho_1 = n \times 10^{-3}$ failed to converge after 91 function evaluations although it was close to a solution. However its behavior near the solution was not at all erratic and the iteration would have undoubtedly converged to a solution if the run had not been terminated. There had been no change in H^k and D_1^k for about 25 iterations preceding termination since Δf^k at each of these steps was insufficiently independent of any set of $(m_k - 1)$ of the previous vectors Δf^i retained in D_1^k. The value of $\|f^k\|_2$ was geometrically decreasing with a ratio of between .91 and .94 over these 25 iterations indicating (slow) linear convergence. The unchanging matrix H^k was near singular and the conditioning of the vectors Δx^i corresponding to the aforementioned Δf^i was very poor. This situation is not altogether unexpected since only the independence of the vectors Δf^i retained in D_1^k is assured in Algorithm I. As noted in Chapter III, the independence of the corresponding vectors Δx^i would be assured only if these vectors had been formed from iterates sufficiently close to the solution. This apparently was not the case in this particular problem.

In Problem #153 with $\rho_1 = 2n \times 10^{-3}$, a similar situation occurred although here the iteration was allowed to continue to a solution (in 106 evaluations) because the linear rate of convergence was better (about 0.76). This rate of convergence existed over the 50 iterations preceding convergence, during which time H^k and D_1^k $(= \Delta F^k)$ was retained unchanged for the same reason as before. Again the matrix H^k was near singular and the conditioning of the vectors Δx^i, corresponding

to the vectors Δf^i retained in D_1^k, was very poor. Despite these two cases of very slow convergence, Algorithm I on the average showed better performance than Secant Algorithm I-S, both in the number of evaluations required for convergence and in the accuracy of H^{kf}.

A number of runs were made with Algorithms I and II in which the value of ρ_1 was either much larger or much smaller than the "nominal" values of $n \times 10^{-3}$ and $2n \times 10^{-3}$. Sometimes these runs produced better results than for the "nominal" values of ρ_1. More often, the results were not quite as good. In general, too small a value of ρ_1 simply resulted in Algorithms I and II degenerating into Algorithms I-S and II-S respectively. Too large a value of ρ_1 often delayed or prevented convergence. No thorough attempt was made to optimize the selection of ρ_1. The rule of thumb developed for selecting "nominal" values of ρ_1 proved adequate to demonstrate improved performance of Algorithm I over I-S and of Algorithm II over II-S in this series of problems. While the results from this one series of problems are not conclusive, these results do support the claim that Algorithms I and II are improvements over Secant Algorithms I-S and II-S respectively.

As noted earlier, Algorithm II showed better performance than did Algorithm I. This was true in almost every problem, in terms of ability to converge, the number of function evaluations required, and the accuracy of H^{kf}. As discussed earlier, this is attributed in large measure to the inherent ability of Algorithm II to avoid singular or near singular matrices H^k, which is lacking in Algorithm I. This characteristic is directly associated in Algorithm II with the selection and retention of "sufficiently" independent vectors Δx^i comprising C_1^k. At the same time Algorithm II insures that H^k is well defined (as does Algorithm I) through

the requirement that the associated vectors Δf^i be "sufficiently" independent (as determined by criterion (2.36)).

Criterion (2.36) utilizes the parameter ρ_2 which was set at $0.1\rho_1$ in this series of problems as previously discussed. Probably as a result of the much smaller value of ρ_2 compared to ρ_1, the relevant criterion associated with ρ_2 was generally not the determining criterion in terms of rejection of a trial vector a_j^k. That is, if criterion (2.35) was satisfied, then criterion (2.36) was usually (but not always) satisfied as well. Perhaps Algorithm II could function almost as well in the great majority of cases by utilizing only criterion (2.35) which assures good conditioning of the retained vectors Δx^i. Possibly the good conditioning of the corresponding vectors Δf^i is not quite as important in general. It might be useful in future research to compare the performance of Algorithm II in a number of cases with a simplified Algorithm II in which criterion (2.36) is dropped.

In this series of problems, Algorithm II showed better performance than Broyden 1 and Algorithm I better than Broyden 2. The ineffectiveness of Broyden 2 has already been noted by Broyden [4] so further comparison is not warranted. Consider the performance of Algorithm II vs Broyden 1. Algorithm II showed up somewhat better in terms of the primary criterion, the number of function evaluations required for convergence. However, the calculations per iteration in Broyden 1 are much simpler than in Algorithm II (or II-S for that matter). Thus the criterion of function evaluations is not quite fair to Broyden 1, since a small average increase in the number of function evaluations may be more than balanced by the decreased calculations at each step. About the only area where Algorithm II showed a decided edge over Broyden 1 is in the ability to reach a solution in this series of problems. Algorithm II converged in all of

the 20 problems under test whereas Broyden 1 failed once (in Problem #152). In view of the foregoing, further numerical experimentation in many types of problems would be required to establish the merit of Algorithm II relative to Broyden 1. The results of the present numerical work suggest that Algorithm II is competitive with Broyden 1 and may have an edge in the ability to reach a solution in many problems.

SECTION V

CONCLUSIONS

Based on the preceding developments, the following conclusions have been reached.

1. Algorithms I and II are improvements over Secant Algorithms I-S and II-S respectively, in terms of ability to converge to a solution $\bar{x}$ of $f(x) = 0$, the number of function evaluations required, and in the closeness of the final matrix H^{k_f} to $J^{-1}(\bar{x})$.

2. Secant Algorithms I-S and II-S have a tendency towards poor convergence, especially as the order of the system increases. This is related to their inability to insure that H^k exists and is non-singular. Algorithm II-S is judged to be the better of the two.

3. Algorithm II is considered to be quite effective and warrants further study. Algorithm I, while more effective than Secant Algorithm I-S, does not measure up to Algorithm II in general performance.

4. Algorithm I insures the existence of H^k while Algorithm II insures both the existence and non-singularity of H^k. These characteristics together with related factors are judged responsible for the better performance in general of Algorithm I compared to I-S, of Algorithm II compared to II-S, and of Algorithm II compared to I.

5. Algorithms I and II are only moderately more complex than Secant Algorithms I-S and II-S respectively. The added complexity consists of the determination at each step of whether the associated criteria are

satisfied. These criteria are relatively simple to apply. Algorithm II has two associated criteria, as opposed to one in Algorithm I, and thus requires somewhat more computations.

6. If the criteria associated with Algorithms I and II are initially satisfied at each step, then under certain additional assumptions these algorithms possess superlinear local convergence. If these criteria are not initially satisfied at each step, these algorithms are still likely to possess linear convergence.

7. Algorithm II appears to be competitive with Broyden's Method 1 in overall effectiveness. The simpler calculations per step in Broyden's Method 1 are offset to some extent by the somewhat faster convergence of Algorithm II. In addition, Algorithm II may have an edge in the ability to reach a solution in many problems.

The following are a few suggestions for additional research in this area.

1. Compare the performance of full step versions of Algorithm II, Algorithm II-S, and Broyden's Method 1 in a variety of problems including those of high order ($n \geq 10$). Experiment with other rules for selecting the parameters ρ_1 and ρ_2 in Algorithm II besides those used in the numerical part of this research.

2. Repeat for one or more norm reducing versions of these three methods.

3. Compare the performance of Algorithm II in item 1 and/or item 2 with that of a simplified version in which the criterion associated with ρ_2 is dropped.

4. Determine the effectiveness of Algorithm II when used in conjunction with the Freudenstein-Roth technique for solving some very difficult problems. The final estimates x^{kf} and H^{kf} of one stage in the process could serve as the initial estimates x^1 and H^1 for the next stage. Compare the overall performance with that obtained when Newton's Method or Broyden's Method 1 is used in this process.

REFERENCES

1. J. Barnes, "An Algorithm for Solving Nonlinear Equations Based on the Secant Method," Computer J., V8, 1965, pp. 66-72.

2. L. Bittner, "A Generalization of the Secant Process," Wissen. Zeit. der Technischen Hochschule Dresden, V9, 1959, pp. 325-329.

3. K. M. Brown and S. D. Conte, "The Solution of Simultaneous Nonlinear Equations," Proc. A.C.M. Nat. Mtg., 1967, pp. 111-114.

4. C. G. Broyden, "A Class of Methods for Solving Nonlinear Simultaneous Equations," Math. of Comput., V19, 1965, pp. 577-593.

5. C. G. Broyden, "Quasi-Newton Methods and their Application to Function Minimisation," Math. of Comput., V21, 1967, pp. 368-381.

6. C. G. Broyden, "A New Method of Solving Nonlinear Simultaneous Equations," Computer J., V12, 1969, pp. 94-99.

7. R. Fletcher and M. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer J., V6, 1963, pp. 163-168.

8. F. Freudenstein and B. Roth, "Numerical Solutions of Systems of Nonlinear Equations," J. Assoc. Comp. Mach., V10, 1963, pp. 550-556.

9. C. Fröberg, Introduction to Numerical Analysis, Addison-Wesley Publishing Co., Reading, Mass., 1965.

10. F. R. Gantmacher, Applications of the Theory of Matrices, Interscience Publishers, New York, 1959.

11. A. S. Householder, Principles of Numerical Analysis, McGraw-Hill, New York, 1953.

12. A. S. Householder, The Theory of Matrices in Numerical Analysis, Blaisdell Publishing Co., New York, 1964.

13. T. A. Jeeves, "Secant Modification of Newton's Method," Comm. Assoc. Comp. Mach., V1, 1958, pp. 9-10.

14. D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," J. Siam, V11, 1963, pp. 431-441.

15. J. Ortega, "Notes on Newton and Secant Methods in n Dimensions," Tech. Note, IBM Fed. Sys. Div., 1967.

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

17. A. M. Ostrowski, Solution of Equations and Systems of Equations, Second Edition, Academic Press, New York, 1966.

18. E. M. Rosen, "A Review of Quasi-Newton Methods in Nonlinear Equation Solving and Unconstrained Optimization," Proc. A.C.M. Nat. Mtg., 1966.

19. J. Rosen, "The Steepest Ascent/Descent Method," Brown University Computing Review, VI/No. 2, 1967, pp. 64-75.

20. J. Sherman and W. J. Morrison, "Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or a Given Row of the Original Matrix," Ann. Math. Statist., V20, 1949, p. 621.

21. L. Tornheim, "Convergence of Multipoint Iterative Methods," J. Assoc. Comp. Mach., V11, 1964, pp. 210-220.

22. J. Traub, Iterative Methods for the Solution of Equations, Prentice Hall, Englewood Cliffs, N. J., 1964.

23. M. Urabe, Nonlinear Autonomous Oscillations, Academic Press, New York, 1967, pp. 280-302.

24. R. G. Voigt, "Rates of Convergence for Iterative Methods for Nonlinear Systems of Equations," University of Maryland Computer Science Center TR 69-97, Aug. 1969.

25. P. Wolfe, "The Secant Method for Simultaneous Nonlinear Equations," Comm. Assoc. Comp. Mach., V2, 1959, pp. 12-13.

26. F. J. Zeleznik, "Quasi-Newton Methods for Nonlinear Equations," J. Assoc. Comp. Mach., V15, 1968, pp. 265-271.

APPENDIX 1

ALTERNATE ORTHOGONALIZATION PROCEDURE FOR ALGORITHMS IB AND IIB

Algorithm IB requires the formation of b_j^k, for possible use in Equations (2.4) and (2.12), so as to satisfy

$$(b_j^k)^T \Delta f_i^k = 0, \quad i = 1, 2, \ldots, m_k'; \; m_k < n \tag{A1.1}$$

where Δf_i^k are the previous independent vectors retained in D_1^k. Gram-Schmidt orthogonalization can be used to accomplish this. However, this can represent a considerable amount of work since each time Δf^k is used to replace some earlier vector in forming D_1^{k+1}, the formation of b_j^{k+1} at the next step must start at or near the beginning of the Gram-Schmidt computational cycle.

A simpler orthogonalization scheme can be developed by utilizing the vectors b_i^k, $i = 1$ to m_k, which are available at every step in Algorithm IB. As indicated previously, the vectors b_i^k and Δf_i^k are related by

$$(b_i^k)^T f_m^k = \delta_{im} \quad (\text{since } B_1^k D_1^k = I). \tag{A1.2}$$

Consider the following expression for forming b_j^k.

$$b_j^k = \Delta f^k - \sum_{i=1}^{m_k} c_i \, b_i^k \tag{A1.3}$$

where c_i are constants to be determined. Since Equation (A1.1) must be satisfied, one obtains

$$0 = (b_j^k)^T \Delta f_m^k = (\Delta f^k)^T \Delta f_m^k - \sum_{i=1}^{m_k} c_i \, (b_i^k)^T \Delta f_m^k$$

$$= (\Delta f^k)^T \Delta f_m^k - \sum_{i=1}^{m_k} c_i \, \delta_{im} \qquad \text{(using A1.2)}$$

$$= (\Delta f^k)^T \Delta f_m^k - c_m . \qquad \text{(A1.4)}$$

Thus

$$c_i = (\Delta f^k)^T \Delta f_i^k \qquad \text{(A1.5)}$$

and substituting in (A1.3) yields

$$b_j^k = \Delta f^k - \sum_{i=1}^{m_k} \left((\Delta f^k)^T \Delta f_i^k \right) b_i^k . \qquad \text{(A1.6)}$$

Equation (A1.6) represents an alternate method for forming b_j^k so as to be orthogonal to the previous vectors Δf_i^k retained in D_I^k. It can be expressed in matrix form as

$$b_j^k = \Delta f^k - (B_I^k)^T \, (D_I^k)^T \Delta f^k . \qquad \text{(A1.7)}$$

The relation between the vector b_j^k obtained by this alternate orthogonalization procedure (A1.6) and that obtained via Gram-Schmidt orthogonalization can be seen in the following derivation. The two different vectors b_j^k will be denoted by

$$z_G^k = b_j^k \text{ as obtained from G-S orthogonalization} \qquad \text{(A1.8)}$$

and

$$z_I^k = b_j^k \text{ as obtained from Equation (A1.6).}$$

Equation (A1.6) shows that z_I^k is a linear combination of Δf^k and the m_k vectors b_i^k. An examination of the operation of Algorithm IB reveals that each vector b_i^k is a linear combination of some row of B_I^{k-1} and the vector b_j^{k-1}. The vector b_j^{k-1} was formed either by orthogonalization (as z_I^{k-1}) or as the j-th row of B_I^{k-1}. In the latter case, the vector Δf^{k-1} replaced some earlier vector Δf^j in forming D_I^k.

Extrapolating this process back to the beginning reveals that z_I^k is a linear combination of Δf^k, some q earlier vectors $\Delta f^{j_1}, \Delta f^{j_2}, \ldots, \Delta f^{j_q}$ that had been replaced by q of the vectors Δf_i^k in D_I^k, and the remaining $(m_k - q)$ vectors Δf_i^k (which did not have to replace any earlier vectors). For convenience in notation, we can arbitrarily rearrange D_I^k so that the latter set of $(m_k - q)$ vectors occupy the first $(m_k - q)$ columns. Then the above described linear combination can be expressed as

$$z_I^k = \Delta f^k + \sum_{i=1}^{m_k - q} s_i \, \Delta f_i^k + \sum_{i=1}^{q} t_i \, \Delta f^{j_i} \qquad \text{(A1.9)}$$

The vector z_G^k as obtained by Gram-Schmidt orthogonalization is that component of Δf^k which is orthogonal to the m_k vectors Δf_i^k in D_I^k. Expressed another way, z_G^k equals Δf^k minus the orthogonal projection of Δf^k upon this subspace. The projection operator for this subspace (see e.g., [12]) is

$$P_D = D_I^k \left((D_I^k)^T \, D_I^k \right)^{-1} (D_I^k)^T. \qquad \text{(A1.10)}$$

Then z_G^k can be expressed as

$$z_G^k = \Delta f^k - P_D \, \Delta f^k = (I - P_D) \, \Delta f^k. \qquad \text{(A1.11)}$$

Since z_I^k is also orthogonal to this subspace,

$$P_D \, z_I^k = 0 \tag{A1.12}$$

and

$$z_I^k = z_I^k - P_D \, z_I^k = (I - P_D) \, z_I^k . \tag{A1.13}$$

From Equations (A1.11) and (A1.13)

$$(z_G^k - z_I^k) = (I - P_D) \, (\Delta f^k - z_I^k)$$

or

$$(z_G^k - z_I^k) = - (I - P_D) \left(\sum_{i=1}^{m_k - q} s_i \, \Delta f_i^k + \sum_{i=1}^{q} t_i \, \Delta f^{j_i} \right)$$

(using (A1.9)). (A1.14)

The first vector sum in (A1.14) lies in the subspace for which P_D is the projection operator. Hence (A1.14) reduces to

$$(z_G^k - z_I^k) = - (I - P_D) \sum_{i=1}^{q} t_i \, \Delta f^{j_i} . \tag{A1.15}$$

The form of (A1.15) shows that $(z_G^k - z_I^k) = 0$ (or $z_G^k = z_I^k$) if the q vectors Δf^{j_1}, ..., Δf^{j_q} all lie in the subspace formed by the vectors Δf_i^k comprising D_I^k. This condition is "almost" satisfied in the following sense. The vector Δf^{j_1} was replaced by some later vector Δf^{k_1} because Δf^{k_1} was "almost" a linear combination of the vectors comprising $D_1^{k_1}$. Expressed another way, Δf^{j_1} was "almost" a linear combination of the vectors retained to form $D_1^{k_1+1}$. Similarly Δf^{j_2} was replaced by a later vector Δf^{k_2} because Δf^{j_2} was nearly a linear combination

of the vectors retained to form $D_I^{k_2+1}$, etc. Then the vectors Δf^{j_1}, ..., Δf^{j_q} are "almost" in the subspace of the vectors comprising D_I^k so that $(z_G^k - z_I^k) \approx 0$. The smaller ρ_1 is in the criterion of (2.5), the closer is this approximation.

At k = p when n - 1 independent vectors Δf_i^p have been retained in D_I^p, z_I^p (if it is non-zero) will be the same as z_G^p (to within a scalar multiple which is irrelevant), regardless of whether or not the replaced vectors Δf^{j_i} are in the subspace spanned by the columns of D_I^p. This subspace would be of dimension n - 1, and since both z_I^p and z_G^p are orthogonal to this subspace, they both must lie along the same line in n space. More formally, referring to (A1.15), $\Sigma t_i \Delta f^{j_i}$ lies in n space and can be expressed as a linear combination of Δf^p and the n - 1 vectors comprising D_I^p. But $(I - P_D)$ operating on this linear combination yields zero everywhere except for the component along Δf^p, or

$$(z_G^p - z_I^p) = -(I - P_D)\,\mu\,\Delta f^p$$

$$= -\mu\, z_G^p \quad \text{(using (A1.11))}$$

or

$$z_I^p = (1 + \mu)\, z_G^p . \tag{A1.16}$$

If no earlier vectors Δf^{j_i} have been replaced up to the k-th instant, then (A1.15) implies that $z_G^k = z_I^k$. This is always the case in (Secant) Algorithm IB-S. Hence the use of the alternate orthogonalization procedure of (A1.6) in Algorithm IB-S yields identical results to those obtained by Gram-Schmidt orthogonalization.

The same orthogonalization procedure just described can also be used in Algorithm IIB to form the trial vector a_j^k satisfying

$$(a_j^k)^T \Delta x_i^k = 0, \quad i = 1, 2, \ldots, m_k; \; m_k < n \tag{A1.17}$$

where Δx_i^k are the previous independent vectors retained in C_1^k. Following the same procedure as before yields the orthogonalization formula

$$a_j^k = \Delta x^k - \sum_{i=1}^{m_k} (\Delta x^k)^T \Delta x_i^k \; a_i^k \tag{A1.18}$$

or in matrix form,

$$a_j^k = \Delta x^k - (A_1^k)^T (C_1^k)^T \Delta x^k . \tag{A1.19}$$

The same comments generally apply as before, with regard to the relation of a_j^k as formed by (A1.18) to that obtained by Gram-Schmidt orthogonalization. Again at k = p, when n - 1 independent vectors Δx_i^p have been retained in C_1^p, the two procedures yield the same results (to within a scalar multiple). When used in (Secant) Algorithm IIB-S, the orthogonalization procedure of (A1.18) yields identical results to that of Gram-Schmidt orthogonalization, for the same reasons as previously discussed.

APPENDIX 2

PROGRAMMING

All of the numerical work was performed on an APL/360 computing system. This system was selected for the numerical part of this research because of the simplicity of its instructions (especially with regard to vector and matrix operations) and because of the convenience of remote terminal operation. While the size of the workspace available to the user is somewhat limited in this system, it was adequate for the range of problems considered in this study (up to and including 15th order systems). The computer operations are performed with a precision of about 16 decimal digits.

The programs listed at the end of this appendix were utilized for all of the numerical work. The programs were devised by the author except for INV (for calculating the inverse of a matrix) and DET (for calculating the determinant), which were borrowed from an APL/360 Public Library. For any given method, the programming is generally not the most efficient in terms of computer operations and storage requirements. Rather the programs were composed in any way that seemed convenient to the author. This in no way affects the validity of the results for any method nor the conclusions drawn therefrom. The programs can always be rewritten to minimize computer operations and storage requirements for a particular method.

As described in Chapter IV, the test problems considered are

$$f(x) = E - (A \sin x + B \cos x) = 0 \qquad \text{(A2.1)}$$

where the notation sin x (or cos x) refers to the n-vector whose i-th component

is sin x_i (or cos x_i), where x_i is the i-th component of x. The components of the $n \times n$ matrices A and B (denoted as AN and BN in programs F and FNR) are random integers in the range -100 to +100. The n-vector E (denoted as EN in programs F and FNR) is obtained from

$$E = A \sin \bar{x} + B \cos \bar{x} \qquad \text{(A2.2)}$$

where $\bar{x}$ (denoted as XN in program FNR) is an n-vector each of whose components is a random number between $-\pi$ and $+\pi$. Thus $\bar{x}$ is automatically one solution of (A2.1). The starting estimate x^1 for a particular problem is formed as

$$x^1 = \bar{x} + 0.1\delta \qquad \text{(A2.3)}$$

where δ (denoted as DN in program FNR) is also an n-vector whose components are each a random number between $-\pi$ and $+\pi$. The problems were generated as follows. For each value of n considered, program RM was used to generate two $5 \times n^2$ matrices each of whose components is a random integer between -100 and +100. These matrices are denoted in program FNR as A2, B2, A5, B5, A10, B10, A15, B15 corresponding to n = 2, 5, 10 and 15 respectively. Similarly, program RV was used to generate two $5 \times n$ matrices for each value of n, each of whose components is a random number between $-\pi$ and $+\pi$. These matrices are denoted in program FNR as X2, D2, X5, D5, X10, D10, X15 and D15 corresponding to the aforementioned values of n. This then provided the basis for five individual problems for each value of n, or a total of 20 problems. A particular problem was set up by running program FNR after specifying the problem (function) number via the variable FCT. Thus for example, setting FCT = 102 and running FNR established the second problem in the n = 10 series; the matrices AN and

BN would be formed from the second rows of A10 and B10 respectively while the vectors XN and DN would be formed from the second rows of X10 and D10 respectively. The initial value x^1 (denoted as X1 in FNR) would also be established as per (A2.3).

Once a problem was set up, an approximation H^1 to the inverse Jacobian at x^1 (denoted as JX1) was formed via program JINV, which formed a discrete approximation to the Jacobian via program F and calculated the inverse via INV. The discretization parameter h was specified as 10^{-4} (via DEL). The matrix denoted as E in JINV is the identity matrix of order n. Then a particular run was specified by assigning component values to the vector variable SEL, which selected among other things the particular algorithm to be used, the quasi-Newton step (full step was used throughout, i.e., $\alpha^k = 1$), the maximum number of iterations permitted, the convergence criterion for $\|f^k\|_2$ (set at 10^{-6} throughout), and the value of ρ_1 for use in Algorithm I or Algorithm II. The parameter ρ_2 in Algorithm II was arbitrarily fixed at $0.1\,\rho_1$.

The program BEGIN started a particular run. This program printed out several items for run identification purposes including the algorithm used, the function (problem) number, and the selected value of ρ_1 (when Algorithm I or Algorithm II was utilized). This program also incorporated the programs INITIAL and ITERATE. The program INITIAL established initial values for several variables and ITERATE performed the actual iterations. Included in ITERATE are programs AMEND and HNEW. The program AMEND formed Δx^k, x^{k+1}, f^{k+1}, and Δf^k in the usual way, for subsequent use. The program HNEW then formed H^{k+1} according to the particular algorithm that had been selected. For example in the discrete Newton's Method (ALG = 9), H^{k+1} was formed as an approximation to

$J^{-1}(x^{k+1})$ using the program JINV. For the other methods, HNEW selected the appropriate program for calculating H^{k+1}. After calculation of H^{k+1}, the execution returned to ITERATE where the iterate index k was incremented by one. After a check for convergence of $\|f^k\|_2$ and for determination of whether k exceeded the allowable limit, the process was repeated.

Besides a discrete Newton's Method, the other methods used on each problem were Broyden's Methods 1 and 2 (ALG = 7 and ALG = 8 respectively), Secant algorithms I-S and II-S (ALG = 5 and 6 respectively), a combination of Broyden's Method 1 and Algorithm II-S (ALG = 10), Algorithm I (ALG = 1), and Algorithm II (ALG = 2). There was also provision for using Algorithms I and II with Gram-Schmidt orthogonalization (ALG = 3 and 4 respectively) but this was only used on a few problems since the computations were more extensive and there was generally little if any improvement over the simpler alternate orthogonalization scheme. The program names for forming H^{k+1} in accordance with the aforementioned methods are BROYDEN, SECANT, BR1ALGIIS, ALGI, and ALGII.

Program SECANT includes Algorithms I-S and II-S. When Algorithm I-S (II-S) is used, Algorithm IB-S (IIB-S) is in effect until k = n + 1 at which time a switch is made to Algorithm IA-S (IIA-S). Similarly, in program ALGI (or ALGII), Algorithm IB (IIB) is in effect until m_k = n at which time a switch is made to Algorithm IA (IIA). The operation of programs ALGI and ALGII will be described in the following paragraphs, with the program variables identified in terms of the nomenclature used elsewhere in this report. The description should also suffice to clarify the operation of the similar but simpler program SECANT.

When Algorithm IB (IIB) is in effect, the vector Z, representing b_j^k (a_j^k), is formed first via an orthogonalization procedure. If the appropriate criteria are

satisfied, as implemented by program BETA, this value of Z is used to form ZMO, ZM, and HN, representing $B^k_{I,0}$ ($A^k_{I,0}$), B^{k+1}_I (A^{k+1}_I), and H^{k+1} respectively. If not, Z is formed as the j-th row of the previous ZM, j being the smallest integer as determined by program TEST such that the appropriate criteria are satisfied. This value of Z is then used to form the new matrices ZM and HN, i.e., B^{k+1}_I (A^{k+1}_I) and H^{k+1} If no value of j in the range of 1 to M, i.e., 1 to m_k, satisfies the criteria, then the previous matrices ZM, DM, and HN are retained unchanged. In each case the updated matrix DM represents D^{k+1}_I.

If and when $m_k = n$, Algorithm IA (IIA) takes over. Then Z is formed as the j-th row of the previous ZM, which now represents $(\Delta F^k)^{-1}$ ($(\Delta X^k)^{-1}$), provided the appropriate criteria are satisfied, with j determined as before. This value of Z is then used to update the matrices ZM and HN. If these criteria cannot be satisfied by any value of j in the range 1 to n, then the previous matrices ZM, DM, and HN are retained unchanged. The matrix DM, representing ΔF^k (ΔX^k), is not actually needed in Algorithm IA (IIA) but is retained as an aid in subsequent analysis. The program PM is utilized in ALGI and ALGII to form PJ, which represents the permuting matrix P_j of order m_k. Also required is the variable EM representing I_{m_k}.

The program COND is used to determine a condition number C_A of a matrix A. If A is $n \times n$, C_A is defined as $|\Delta^k|$ of (3.6) (for $A = \Delta X^k$). If A is $n \times m$, $m < n$, C_A is given by

$$C_A = \frac{|\det (A^T A)|^{1/2}}{\prod_{i=1}^{m} \|a^i\|_2}, \quad a^i \triangleq \text{i-th column of A.} \qquad \text{(A2.4)}$$

In either case, C_A measures the relative independence of the columns of A, for use in later analysis.

```
      ∇ RM [⎕] ∇
    ∇ A←RM N;MN
[1]   MN←(5,N*2)⍴201
[2]   A←(?MN)-101
    ∇

      ∇ RV [⎕] ∇
    ∇ A←RV N;I;LA;W1;W2;RD1
[1]   I←0
[2]   LA←(5×N)⍴0
[3]   W1←10*(-⍳16)
[4]  RV1:I←I+1
[5]   →(I>5×N)/RV2
[6]   W2←(?16⍴10)-1
[7]   RD1←+/W1×W2
[8]   LA[I]←(○2)×(RD1-0.5)
[9]   →RV1
[10] RV2:A←(5,N)⍴LA
    ∇

      ∇ FNR [⎕] ∇
    ∇ FNR
[1]   RES←10|FCT
[2]   N←0.1×FCT-RES
[3]   →(FCT>50)/F50
[4]   AN← 2 2 ⍴A2[RES;]
[5]   BN← 2 2 ⍴B2[RES;]
[6]   XN←X2[RES;]
[7]   DN←D2[RES;]
[8]   →F0
[9]  F50:→(FCT>100)/F100
[10]  AN← 5 5 ⍴A5[RES;]
[11]  BN← 5 5 ⍴B5[RES;]
[12]  XN←X5[RES;]
[13]  DN←D5[RES;]
[14]  →F0
[15] F100:→(FCT>150)/F150
[16]  AN← 10 10 ⍴A10[RES;]
[17]  BN← 10 10 ⍴B10[RES;]
[18]  XN←X10[RES;]
[19]  DN←D10[RES;]
[20]  →F0
[21] F150:AN← 15 15 ⍴A15[RES;]
[22]  BN← 15 15 ⍴B15[RES;]
[23]  XN←X15[RES;]
[24]  DN←D15[RES;]
[25] F0:EN←(AN+.×1○XN)+(BN+.×2○XN)
[26]  X1←XN+0.1×DN
    ∇

      ∇ F [⎕] ∇
    ∇ Y←F X
[1]   Y←EN-((AN+.×1○X)+(BN+.×2○X))
    ∇
```

```
      ∇ BEGIN [□] ∇
    ∇ BEGIN
[1]   ALG←SEL[1]
[2]   STEP←SEL[2]
[3]   FCT←SEL[3]
[4]   H1←SEL[4]
[5]   DEL←SEL[5]
[6]   RHO←SEL[6]
[7]   LIM←SEL[7]
[8]   CONV←SEL[8]
[9]   PRINT←SEL[9]
[10]  0ρ0
[11] N1:→N1+(2×ALG)-1
[12]  'ALGORITHM I; ALT. ORTHOG.'
[13]  →N2
[14]  'ALGORITHM II, ALT. ORTHOG'
[15]  →N2
[16]  'ALGORITHM I; G-S ORTHOG.'
[17]  →N2
[18]  'ALGORITHM II; G-S ORTHOG.'
[19]  →N2
[20]  'ALGORITHM I-S'
[21]  →N2
[22]  'ALGORITHM II-S'
[23]  →N2
[24]  'BROYDEN 1'
[25]  →N2
[26]  'BROYDEN 2'
[27]  →N2
[28]  'DISCRETE NEWTON; DELTA IS ';DEL
[29]  →N2
[30]  'COMBINATION ALG II-S AND BROYDEN 1'
[31] N2:→(STEP>1)/N3
[32]  'FULL STEP'
[33]  →N4
[34] N3:'REDUCED STEP; SCHEME ';STEP
[35] N4:'FUNCTION NO. ';FCT;';  N IS ';N
[36]  →(ALG=9)/N7
[37]  →(H1>1)/N5
[38]  'H1 IS JINV X1; DELTA IS ';DEL
[39]  →N6
[40] N5:'H1 IS I'
[41] N6:→(ALG>4)/N7
[42]  'RHO IS ';RHO
[43] N7:0ρ0
[44]  INITIAL
[45]  ITERATE
    ∇
```

```
      ∇ INITIAL [□] ∇
    ∇ INITIAL
[1]   NX←X1
[2]   FNX←F NX
[3]   MFNX←+/FNX×FNX
[4]   K←M←Q←L←0
[5]   DM←N⍴0
[6]   ZM←⍉DM
[7]   →(H1=1)/IN
[8]   HN←E
[9]   →0
[10] IN:HN←JX1
    ∇


      ∇ ITERATE [□] ∇
    ∇ ITERATE
[1]  ITER:X←NX
[2]   FX←FNX
[3]   H←HN
[4]   MAGF←MFNX
[5]   K←K+1
[6]   'K: ';K;';   NORM OF FX: ';MAGF*0.5
[7]   →(PRINT<3)/LT
[8]   →((5|K)≠0)/LT
[9]   'COND. NO. OF H: ';COND H
[10]  'DET OF H: ';DT
[11]  →(ALG>6)/LT
[12]  'COND. NO. OF DM: ';COND DM
[13]  'DET OF DM: ';DT
[14] LT:→((MAGF*0.5)<CONV)/SOL
[15]  →(PRINT<1)/LT1
[16]  'MAX DIFF FROM EXPECTED SOL: ';⌈/|X-XN
[17]  →(PRINT<2)/LT1
[18]  'MAX DIFF BETWEEN H AND JINV AT SOL: ';⌈/|(N×N)⍴(H-JXF)
[19] LT1:→(K>LIM)/TERM
[20]  P←-H+.×FX
[21]  AMEND
[22]  HNEW
[23]  →ITER
[24] TERM:'TERMINATED; NO. OF ITERATIONS EXCEEDS LIMIT'
[25]  →END
[26] SOL:0⍴0
[27]  →(N>5)/LT5
[28]  'SOLUTION IS: '
[29]  X
[30] LT5:'MAX DIFF FROM EXPECTED SOL: ';⌈/|X-XN
[31]  'MAX ABS ERROR IN APPR JINV AT SOL: ';⌈/|(N×N)⍴(H-JXF)
[32]  →(PRINT<4)/END
[33]  'APPROX JINV AT SOLUTION: '
[34]  H
[35] END: 3 0 ⍴0
    ∇
```

```
      ∇ AMEND [⎕] ∇
    ∇ AMEND
[1]   →(STEP>1)/RED
[2]   DX←P
[3]   NX←X+DX
[4]   FNX←F NX
[5]   MFNX←+/FNX×FNX
[6]   →MR21
[7]  RED:REDUCE
[8]  MR21:DF←FNX-FX
[9]   →(PRINT<3)/0
[10]  'MAGN OF DX: ';⌈/|DX
[11]  'MAGN OF DF: ';⌈/|DF
    ∇
```

```
      ∇ HNEW [⎕] ∇
    ∇ HNEW
[1]   →(ALG≠9)/NW1
[2]   HN←JINV NX
[3]   →OUT
[4]  NW1:→(ALG=10)/NW5
[5]   →(ALG<7)/NW2
[6]   BROYDEN
[7]   →OUT
[8]  NW2:→(ALG<5)/NW3
[9]   SECANT
[10]  →OUT
[11] NW3:→((ALG=2)∨(ALG=4))/NW4
[12]  ALGI
[13]  →OUT
[14] NW4:ALGII
[15] OUT:0⍴0
[16]  →0
[17] NW5:BR1ALGIIS
    ∇
```

```
      ∇ JINV [⎕] ∇
    ∇ HT←JINV XT;YT;VT;DYT;I;ZT;BT
[1]   YT←F XT
[2]   VT←⍳I←0
[3]  LOOP:ZT←XT+DEL×E[;I←I+1]
[4]   DYT←(F ZT)-YT
[5]   VT←VT,DYT
[6]   →(I<N)/LOOP
[7]   BT←(÷DEL)×(N,N)⍴VT
[8]   HT←INV(⍉BT)
    ∇
```

```
      ∇ SECANT [□] ∇
    ∇ SECANT;U;V;Z;ZDV;ZM0;PJ;DM0
[1]   U←DX-H+.×DF
[2]   →(ALG=6)/S2
[3]  S1:DV←DF
[4]   →S3
[5]  S2:DV←DX
[6]  S3:→(K>N)/S4
[7]   Z←DV-(⌹ZM)+.×(⌹DM)+.×DV
[8]   ZDV←+/Z×DV
[9]   ZM0←ZM-(÷ZDV)×(ZM+.×DV)∘.×Z
[10]  ZM←(K,N)ρ((N×K-1)ρZM0),Z÷ZDV
[11]  DM←⌹(K,N)ρ((N×K-1)↑⌹DM),DV
[12]  →S5
[13] S4:Z←ZM[1;]
[14]  EM←(N,N)ρ1,Nρ0
[15]  ZM0←ZM+(÷+/Z×DV)×(EM[1;]-ZM+.×DV)∘.×Z
[16]  M←N
[17]  PJ←PM 1
[18]  ZM←PJ+.×ZM0
[19]  DM0←DM+(DV-DM[;1])∘.×EM[;1]
[20]  DM←DM0+.×(⌹PJ)
[21] S5:→(ALG=6)/S6
[22]  V←Z
[23]  →S7
[24] S6:V←(⌹H)+.×Z
[25] S7:HN←H+(÷+/V×DF)×U∘.×V
    ∇

      ∇ BR1ALGIIS [□] ∇
    ∇ BR1ALGIIS;U;V;Z;ZDV;ZM0;PJ;DM0
[1]   U←DX-H+.×DF
[2]   DV←DX
[3]   →(K>N)/SB4
[4]   Z←DV-(⌹ZM)+.×(⌹DM)+.×DV
[5]   ZDV←+/Z×DV
[6]   ZM0←ZM-(÷ZDV)×(ZM+.×DV)∘.×Z
[7]   ZM←(K,N)ρ((N×K-1)ρZM0),Z÷ZDV
[8]   DM←⌹(K,N)ρ((N×K-1)↑⌹DM),DV
[9]   →SB5
[10] SB4:Z←ZM[1;]
[11]  EM←(N,N)ρ1,Nρ0
[12]  ZM0←ZM+(÷+/Z×DV)×(EM[1;]-ZM+.×DV)∘.×Z
[13]  M←N
[14]  PJ←PM 1
[15]  ZM←PJ+.×ZM0
[16]  DM0←DM+(DV-DM[;1])∘.×EM[;1]
[17]  DM←DM0+.×(⌹PJ)
[18] SB5:ZC←0.5×(Z+DV)
[19]  V←(⌹H)+.×ZC
[20]  HN←H+(÷+/V×DF)×U∘.×V
    ∇
```

```
      ∇ BROYDEN [⎕] ∇
    ∇ BROYDEN;U;V;DH
[1]   U←DX-H+.×DF
[2]   →(ALG=8)/BR2
[3]  BR1:V←(⌹H)+.×DX
[4]   →BR
[5]  BR2:V←DF
[6]  BR:DH←(÷+/V×DF)×U∘.×V
[7]   HN←H+DH
    ∇


      ∇ ALGI [⎕] ∇
    ∇ ALGI;U;V;Z;ZDV;ZM0;PJ;DM0
[1]   U←DX-H+.×DF
[2]   DV←DF
[3]   →(M≥N)/A12
[4]   →(ALG=3)/GR1
[5]   Z←DV-(   M)+.×(⌹DM)+.×DV
[6]   →A11
[7]  GR1:GRAM
[8]  A11:CHK←Z BETA DV
[9]   →(CHK≤RHO)/A15
[10]  ZDV←+/Z×DV
[11]  ZM0←ZM-(÷ZDV)×(ZM+.×DV)∘.×Z
[12]  ZM←((M+1),N)⍴((N×M)↑ZM0),Z÷ZDV
[13]  DM←⌹((M+1),N)⍴((N×M)↑⌹DM),DV
[14]  M←M+1
[15]  'NEXT M INCREASED TO: ';M
[16]  EM←(M,M)⍴1,M⍴0
[17]  →A14
[18] A15:'NO CHANGE IN NEXT M;  BETA= ';CHK
[19] A12:TEST
[20]  →(CHANGE)/A13
[21]  'NO CHANGE IN NEXT H, ZM, AND DM'
[22]  →0
[23] A13:Z←ZM[J;]
[24]  'Z IS ZM[';J;';]'
[25]  ZDV←+/Z×DV
[26]  ZM0←ZM+(÷ZDV)×(EM[;J]-ZM+.×DV)∘.×Z
[27]  PJ←PM J
[28]  ZM←PJ+.×ZM0
[29]  DM0←DM+(DV-DM[;J])∘.×EM[;J]
[30]  DM←DM0+.×(⌹PJ)
[31] A14:HN←H+(÷ZDV)×U∘.×Z
    ∇
```

```
      ∇ ALGII [□] ∇
    ∇ ALGII;U;V;Z;ZDV;ZM0;PJ;DM0
[1]   U←DX-H+.×DF
[2]   DV←DX
[3    →(M≥N)/A22
[4]   →(ALG=4)/GR2
[5]   Z←DV-(⍉ZM)+.×(⍉DM)+.×DV
[6]   →A21
[7]  GR2:GRAM
[8]  A21:CHK1←Z BETA DV
[9]   CHK2←((⍉H)+.×Z) BETA DF
[10]  →(CHK1≤RHO)/A25
[11]  →(CHK2≤0.1×RHO)/A25
[12]  ZDV←+/Z×DV
[13]  ZM0←ZM-(÷ZDV)×(ZM+.×DV)∘.×Z
[14]  ZM←((M+1),N)⍴((N×M)↑ZM0),Z÷ZDV
[15]  DM←⍉((M+1),N)⍴((N×M)↑⍉DM),DV
[16]  M←M+1
[17]  'NEXT M INCREASED TO: ';M
[18]  EM←(M,M)⍴1,M⍴0
[19]  →A24
[20] A25:'NO CHANGE IN NEXT M;  GAMMA= ';CHK1;', BETA= ':CHK2
[21] A22:TEST
[22]  →(CHANGE)/A23
[23]  'NO CHANGE IN NEXT H, ZM, AND DM'
[24]  →0
[25] A23:Z←ZM[J;]
[26]  'Z IS ZM[';J;';]'
[27]  ZDV←+/Z×DV
[28]  ZM0←ZM+(÷ZDV)×(EM[;J]-ZM+.×DV)∘.×Z
[29]  PJ←PM J
[30]  ZM←PJ+.×ZM0
[31]  DM0←DM+(DV-DM[;J])∘.×EM[;J]
[32]  DM←DM0+.×(⍉PJ)
[33] A24:V←(⍉H)+.×Z
[34]  →(PRINT<2)/A33
[35]  'I-ZM×DM: ';⌈/|(M×M)⍴(EM-ZM+.×DM)
[36] A33:HN←H+(÷+/V×DF)×U∘.×V
    ∇
```

```
      ∇ TEST [□] ∇
    ∇ TEST
[1]   J←0
[2]  T1:J←J+1
[3]   →(J>M)/T3
[4]   TJ←ZM[J;] BETA DV
[5]   →(TJ>RHO)/T2
[6]   →T1
[7]  T2:→((ALG=1)∨(ALG=3))/T4
[8]   VA←(⍉H)+.×ZM[J;]
[9]   TJA←VA BETA DF
[10]  →(TJA>0.1×RHO)/T4
[11]  'TJ[';J;';] O.K.;  TJA[';J;';]= ';TJA
[12]  →T1
[13] T3:CHANGE←0
[14]  →0
[15] T4:CHANGE←1
    ∇


      ∇ BETA [□] ∇
    ∇ SC←V1 BETA V2;MV1;MV2;M12
[1]   M12←+/V1×V2
[2]   MV1←(+/V1*2)*0.5
[3]   MV2←(+/V2*2)*0.5
[4]   SC←(*MV1×MV2)×|M12
    ∇


      ∇ PM [□] ∇
    ∇ A←PM J;VJ;PCJ;ZJ
[1]   VJ←(M⍴1)-EM[;J]
[2]   PCJ←VJ/[1] EM
[3]   ZJ←((M×M-1)⍴PCJ),EM[;J]
[4]   A←(M,M)⍴ZJ
    ∇


      ∇ GRAM [□] ∇
    ∇ GRAM;I;ZIM;DVI;ZI;MZI
[1]   I←0
[2]   ZIM←N⍴0
[3]  REP:→(I≥M)/LAST
[4]   I←I+1
[5]   DVI←DM+.×EM[;I]
[6]   ZI←DVI-ZIM+.×(⍉ZIM)+.×DVI
[7]   MZI←(+/ZI×ZI)*0.5
[8]   ZIM←⍉(I,N)⍴((N×I-1)⍴⍉ZIM),ZI÷MZI
[9]   →REP
[10] LAST:Z←DV-ZIM+.×(⍉ZIM)+.×DV
    ∇
```

```
      ∇ COND [□] ∇
    ∇ Y←COND A;RA;I;VM;VI;VIM
[1]   RA←ρA[1;]
[2]   →(RA<N)/CD0
[3]   DT←DET A
[4]   →CD3
[5] CD0:DT←(|DET(⊖A)+.×A)*0.5
[6] CD3:I←1-VM←1
[7] CD1:I←I+1
[8]   →(I>RA)/CD2
[9]   VI←A[;I]
[10]  VIM←(+/VI×VI)*0.5
[11]  VM←VIM×VM
[12]  →CD1
[13] CD2:Y←|DT÷VM
    ∇


      ∇ INV [□] ∇
    ∇ RB←INV RA;RK;RS;RP;RI
[1]   →((2=ρρRA)∧=/1,ρRA)⍴4
[2]   'NO INVERSE!'
[3]   →⍬RB←1
[4]   RK←⌊/ρRA
[5]   RS←RK
[6]   RP←⍳RK
[7]   RA←RA[;(⍳RS),1]
[8]   RA[;1+RS]←(⍳RS)≤1
[9]   RI←(|RA[⍳RK;1])⍳⌈/|RA[⍳RK;1]
[10]  RP[1,RI]←RP[RI,1]
[11]  RA[1,RI;⍳RS]←RA[RI,1;⍳RS]
[12]  →(1E¯30>|RA[1;1])⍴2
[13]  RA[1;]←RA[1;]÷RA[1;1]
[14]  RA←RA-((~(⍳RS)≤1)×RA[;1])∘.×RA[1;]
[15]  RA←RA[1+RS|⍳RS;(1+⍳RS),1]
[16]  RP←RP[1+RS|⍳RS]
[17]  →(0<RK←RK-1)/8
[18]  RB←RA[;RP⍳⍳RS]
    ∇


      ∇ DET [□] ∇
    ∇ C←DET Z;J;Q
[1]   →(1=ρ,Z)⍴0,C←,Z
[2]   →L2×⍳(2=ρρZ)∧=/ρZ
[3]   →0,ρ□←'ILLEGAL  STRUCTURE'
[4] L2:→0×⍳(1↑ρZ)<J←(Z[1;]=0)⍳C←,0
[5]   Z←(J-1)⌽Z
[6] L6:Z←Z-Z[;1]∘.×Z[1;]÷C←Z[1;1]
[7]   C←(¯1*J-1)×C×DET 1 1 ↓Z
    ∇
```