COMPUTATIONAL EXPERIMENTS WITH SYSTEMS OF NONLINEAR EQUATIONS

By

GUANG-NAY WANG

Bachelor of Science
National Taiwan Normal University
Taipei, Taiwan, Republic of China
1968

Master of Arts
Northeast Missouri State University
Kirksville, Missouri
1974

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF EDUCATION
December, 1978

COMPUTATIONAL EXPERIMENTS WITH SYSTEMS

OF NONLINEAR EQUATIONS

Thesis Approved:

Donald W. Grace
Thesis Adviser

Donald D. Fisher

James R. Choike

James Gerald Wright

Norman N. Durham
Dean of the Graduate College

ACKNOWLEDGMENTS

The author wishes to express his appreciation to his major adviser, Dr. Donald W. Grace, for his support, guidance, concern, and encouragement throughout this study. Appreciation is also expressed to the other committee members, Dr. Donald D. Fisher, Dr. James R. Choike, and Dr. James Y. Yelvington, for their assistance and cooperation.

A note of thanks is given to Dr. J. P. Chandler for his invaluable assistance in the preparation of the computer program.

Thanks are also extended to Mrs. M. F. Wang and Ms. Charlene Fries for the excellent typing of this manuscript.

Special gratitude is expressed to my wife, Yu-Shien, and to our daughter, Hsiu-Hung, for their understanding, love, and numerous sacrifices.

Finally, very special gratitude goes to the author's parents, Mr. and Mrs. Peng Wang, for their encouragement and support throughout the study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\phi(X)$, $X = (\xi_1, \xi_2, \ldots, \xi_n)^T$	function to be minimized
$F(X) = (f_1, f_2, \ldots, f_n)(X)$	vector function, of which a zero is sought
$J(X) = (\partial f_i / \partial \xi_j)$	Jacobian matrix of $F(X)$
$B_k(X)$	k^{th} approximation to $J(X)$
$H_k(X)$	k^{th} approximation to $[J(X)]^{-1}$
$F_k = F(X_k)$	vector function value at X_k
$d_k = -H_k F_k$	search direction
λ_k	step size from linear search
$S_k = \lambda_k d_k = X_{k+1} - X_k$	step

Vectors are denoted by lower case Latin letters and scalars are denoted by lower case Greek letters.

CHAPTER I

INTRODUCTION

In nonlinear unconstrained optimization, the necessary conditions for stationary points ($\frac{\partial \phi}{\partial \xi_j} = 0$) comprise systems of nonlinear equations. When derivatives do not exist or are costly to compute, closed form solutions may not be feasible, forcing analysts to devise iterative numerical algorithms. Every new journal contains suggested approaches--some new, some "warmed-over." Implementation is often left to the reader.

One such algorithm is proposed by David M. Gay and Robert B. Schnabel (38). The essence of the algorithm is to make use of steps (in the trajectory of the solution vector) from prior iterations to help determine the next step. These prior steps, S_j, $j = 0, 1, \ldots, i - 1$, are used in the calculation of an approximation to the Hessian matrix, needed in the quasi-Newton determination of the next step, S_i.

Broyden (13) uses the prior step S_{i-1} in determining a new step S_i. Gay and Schnabel propose forcing the linear independence of the S_j's by making $\hat{S}_i$ orthogonal to the subspace spanned by $\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_{i-1}$, where $\hat{S}_0 = S_0$, the initial step, then using this $\hat{S}_i$ to help determine a new step S_{i+1}. This method is different from, but related to, Broyden's method.

This dissertation implements the Gay-Schnabel method on the IBM 370/158 computer and compares its performance with that of other published algorithms, such as Broyden's method, Brown's method for a system

of nonlinear equations, and the Davidon-Fletcher-Powell (DFP) method for minimization problems with analytic gradient vector. Criteria used for comparison will be convergence rates, precision, number of function evaluations, user effort required, etc. Basic concepts and detailed descriptions of these algorithms will be presented in later chapters.

Two variations of Gay and Schnabel's proposal are also discussed, implemented, and evaluated.

The Source of the Problem

We are concerned with the problem of "computing" a solution of the system of n nonlinear equations in n variables. Let the system be given as

$$f_j(\xi_1, \xi_2, \ldots, \xi_n) = 0, \; j = 1, 2, \ldots, n. \tag{1.1}$$

where each f_j, $j = 1, 2, \ldots, n$ is differentiable. These may be written more concisely as

$$F(X) = 0 \tag{1.2}$$

where X is the column vector of independent variables and F the column vector of functions f_j. The problem can be stated as:

Find $X^* \in R^n$ such that $F(X^*) = 0$

where $F : R^n \to R^n$ is differentiable.

If the n nonlinear Equation (1.2) is obtained by setting the first partial derivatives of some functional of n variables, $\phi : R^n \to R$, equal to zero, then solving these equations is equivalent to finding a stationary point of $\phi(X)$, since a necessary condition for the point X^* to be a local minimum of $\phi(X)$ is that $F(X^*) = 0$, where $F(X)$ is the gradient

vector of $\phi(X)$ defined on R^n. The Jacobian of $F(X)$ is in this case the Hessian matrix of $\phi(X)$ and is symmetric, provided $\phi(X)$ and its first partial derivatives are continuous. If $\phi(X)$ is convex, then any stationary point of $\phi(X)$ is a minimum and the Hessian matrix is either positive definite or semidefinite. Hence, any method used to solve a system of nonlinear equations can be applied to the minimization problem.

On the other hand, there is a way of converting the problem of solving the system $F(X) = 0$ into a minimization problem. Let g be a functional defined on R^n such that the point $X = 0$ is the unique global minimum of g. For instance, we might choose $g(X) = ||X||$, with some norm in R^n, then define $\phi(X) = g(F(X))$, for $X \in R^n$, i.e., $\phi(X) = ||F(X)||$, for $x \in R^n$. Any solution $X^* \in R^n$ of $F(X) = 0$ is a global minimum of ϕ, and hence we may find X^* by minimizing ϕ. In case of $g(X) = X^TX$, the functional $\phi : R^n \to R$ to be minimized has the form,

$$\phi(X) = \sum_{i=1}^{n} (f_i(X))^2, \quad X \in R^n,$$

and a global minimum of $\phi(X)$ is called a least-squares solutions of the system $F(X) = 0$.

For the reason mentioned above, the survey of literature in a later section includes both the algorithms for solving systems of nonlinear equations and those of minimization problems.

If the system of Equation (1.1) has solutions, we will be concerned with how to approximate them. Many works in this area are the methods for approximating one solution of Equation (1.1); few papers describe methods for constructing sets containing solutions, that is, using any estimating method to define an interval or intervals in which the roots

of Equation (1.1) must lie and the methods for finding most or all solutions of Equation (1.1).

Mathematical Background

Generally, because the direct methods for solving Equation (1.1) are not feasible, a class of iterative processes for solving Equation (1.1) or minimization problems in R^n has been considered frequently in recent years. One of its members is a quasi-Newton method which is a special case of the general update method or modification method (30).

Intuitively, an iterative process Ω is a rule starting at an initial point X_0 in the domain $D \varepsilon R^n$ to obtain an improved point $X_1 \varepsilon D \varepsilon R^n$. Repeat the process and a sequence of ever-improving points $\{X_k\}$ is generated that approaches a solution X^* of the given problem. Usually, the process is terminated when a point sufficiently close to the solution point is obtained.

Let $\phi: R^n \to R$ be a functional to be minimized such that $F : R^n \to R^n$ is the gradient vector of $\phi(X)$; suppose F is differentiable. Expand $\phi(X)$ as a Taylor series in the neighborhood of a point $X_0 \varepsilon R^n$

$$\phi(X) = \phi(X_0) + (F(X_0))^T (X - X_0) + \frac{1}{2} (X - X_0)^T J(X_0)(X - X_0)$$

$$+ \text{Higher Order Terms}$$

where $J(X_0)$ is the Jacobian of $F(X)$ evaluated at X_0 and is the Hessian matrix of $\phi(X)$ at X_0.

If X is sufficiently close to X_0 such that the higher order terms can be ignored when compared with the first three terms, then we can approximate $\phi(X)$ by the following form,

$$\phi(X) = \phi(X_0) + (F(X_0))^T (X - X_0) + \frac{1}{2} (X - X_0)^T J(X_0)(X - X_0)$$

Since we want to minimize $\phi(X)$, we can do so by differentiating the above form with respect to X and setting the result to zero. This implies

$$F(X_0) + J(X_0)(X - X_0) = 0.$$

If $J(X_0)$ is nonsingular, we can solve X from the above form,

$$X = X_0 - (J(X_0))^{-1} F(X_0).$$

It says that the stationary point X of $\phi(X)$ can be taken from X_0 in the direction of the negative gradient modified by $(J(X_0))^{-1}$. This gives a basic iterative procedure for approximating a solution of Equation (1.1) or a minimum of $\phi(X)$.

The general iterative formula known as Newton's method can be formed by taking $X = X_{k+1}$ and $X_0 = X_k$ as follows:

$$X_{k+1} = X_k - (J(X_k))^{-1} F(X_k), \quad k = 0, 1, 2, 3, \ldots . \tag{1.3}$$

It is essential to the convergence of this method that the inverse of the Hessian, $(J(X_k))^{-1}$, of $\phi(X)$ be positive definite. The sufficient condition is that the $\phi(X)$ is convex and has continuous second partial derivatives.

The advantages of this algorithm are that if it works, then it works extremely well; convergence is rapid and in general is Q-quadratic; if a sufficiently good initial estimate of the solution can be made, it probably is the best method, and, if F(X) is linear with a nonsingular Jacobian matrix (or $\phi(X)$ is quadratic with nonsingular Hessian matrix), then $X_1 = X^*$.

Newton's method suffers from two serious disadvantages from the point of view of practical calculation. The first of these is that it

often fails to converge to a solution if one starts from a poor initial point. To overcome this problem, the implementations are often of the form

$$X_{k+1} = X_k - \lambda_k (J(X_k))^{-1} F(X_k), \quad k = 0, 1, 2, \ldots \tag{1.4}$$

where λ_k is a scalar multiplier. This parameter is determined by a "linear search" method (15)(16), such that X_{k+1} is a better approximation to the solution of the problem than X_k, i.e.,

$$\phi(X_{k+1}) < \phi(X_k) \quad \text{or} \quad ||F(X_{k+1})|| < ||F(X_k)||$$

for some norm. It has been observed, however, that this can inhibit convergence if continued when the iterates are close to the solution (Broyden, 1970b).

The second disadvantage of Newton's method is the difficulty of computing the Jacobian matrix if F(X) is a complicated function. In the majority of practical problems it is impossible to obtain the partial derivatives analytically, and even if it were possible it would be an extremely laborious and time-consuming operation. In order to compute the Jacobian matrix numerically (finite differences method) the vector function F(X) must be evaluated for at least n + 1 sets of independent variables.

To overcome this difficulty, many authors use a matrix $B(X_k)$ to approximate the Jacobian matrix $J(X_k)$ in such a way that it is modified at each iteration so that is possesses, to some extent, the properties of the Jacobian matrix $J(X_k)$. Some other authors use a matrix $H(X_k)$ to approximate the inverse of the Jacobian $(J(X_k))^{-1}$ if this Jacobian is nonsingular. It is recognized that this matrix $H(X_k)$ is to approximate

the inverse of the Hessian of the functional $\phi(X)$, when applied to a minimization problem.

From now on, we use H_k, B_k, J_k, F_k to denote $H(X_k)$, $B(X_k)$, $J(X_k)$, and $F(X_k)$, respectively. Define

$$S_k = X_{k+1} - X_k, \tag{1.5}$$

$$Y_k = F_{k+1} - F_k. \tag{1.6}$$

Consider $F(X)$ as a Taylor series expansion in the neighborhood of $X_k \in R^n$ such that the higher terms can be ignored. Then

$$F(X) = F_k + J_k(X - X_k) \tag{1.7}$$

Let $X = X_{k+1}$, the point for the next iteration, then Equation (1.7) will be in the following form:

$$F_{k+1} - F_k = J_k(X_{k+1} - X_k)$$

or

$$Y_k = J_k S_k. \tag{1.8}$$

Since B_k is used to approximate J_k, it is desirable that B_k satisfies $Y_k = B_k S_k$. By Equation (1.6), we know that Y_k depends on F_{k+1}, which depends on X_{k+1}, and this, by Equation (1.4), in turn depends on B_k. So this equation $Y_k = B_k S_k$ cannot be used to determine B_k, but we can expect, in the next iteration, that B_{k+1}, a modification of B_k by a correction matrix ΔB_k of rank m ($1 \leq m \leq 2$), satisfies the equation

$$Y_k = B_{k+1} S_k. \tag{1.9}$$

We conclude the above discussion and make the following definition of the quasi-Newton method to solve the system of equations defined as

Equation (1.2). Equation (1.9) is usually called the quasi-Newton equation.

Definition 1.1 (Direct update method of rank m): Given $F : R^n \to R^n$ is differentiable. A quasi-Newton method when applied to solve $F(X) = 0$ is an iterative process which generates a sequence $\{X_k\}$, $k = 0, 1, 2, \ldots$ of approximations to the zero of F. At each iteration, given $X_k \in R^n$ and $B_k \in R^{nxn}$, the next approximation is defined by

$$X_{k+1} = X_k - \lambda_k B_k F_k$$

where λ_k is chosen to reduce $F(X_k)$, and

$$B_{k+1} = B_k + \Delta B_k \qquad \text{rank } \Delta B_k = m \geq 1$$

is chosen to satisfy the quasi-Newton Equation (1.9).

Instead of storing and updating B_k, the approximation to the Jacobian, at each iteration, one would store and update $H_k = B_k^{-1}$ if this matrix exists for all K. Applying the Sherman-Morrison-Woodbury formula, we derive $\Delta H_k = H_{k+1} - H_k$ from $B_{k+1} = B_k + \Delta B_k$, where ΔH_k is still of rank m. Therefore, another version of Definition 1.1 can be stated as follows:

Definition 1.2 (Inverse update method of rank m): Given $F : R^n \to R^n$ is differentiable. A quasi-Newton method when applied to solve $F(X) = 0$ is an iterative process which generates a sequence $\{X_k\}$, $k = 0, 1, 2, \ldots$ of approximations to the zero of F. At each iteration, given $X_k \in R^n$ and $B_k \in R^{nxn}$, the next approximation is defined by

$$X_{k+1} = X_k - \lambda_k H_k F_k$$

where λ_k is chosen to reduce $||F(X_k)||$

and

$$H_{k+1} = H_k + \Delta H_k \qquad \text{rank } \Delta H_k = m \geq 1$$

is chosen to satisfy

$$S_k = H_{k+1} Y_k.$$

In practice, the rank m is at most 2. The matrix H_k is an approximation to J_k^{-1}, the inverse of the Jacobian.

In the following chapters we will state several different algorithms and compare their performance. One of the important criteria to be used is the asymptotic rate of convergence of the process at X^*. Here we state several fundamental definitions about this concept.

Definition 1.3: A sequence $\{X_k\} \in R^n$ converges to $X^* \in R^n$ if and only if, for each $\varepsilon > 0$, there exists an integer, N_0, such that for all $k > N_0$, $||X_k - X^*|| < \varepsilon$ with respect to some norm. X^* is called the limit of this sequence X_k.

Definition 1.4: Let $\{X_k\} \in R^n$ be any convergent sequence with limit X^*. Then the quantities

$$Q_p\{X_k\} = \begin{cases} \limsup_{k \to \infty} \dfrac{||X_{k+1} - X^*||}{||X_k - X^*||^p}, & \text{if } X_k \neq X^*, \text{ for all but finitely many } k, \\ 0, & \text{if } X_k = X^*, \text{ for all but finitely many } k, \\ +\infty & \text{otherwise} \end{cases}$$

defined for all $p \in [1, \infty)$, are the quotient convergence factors, or Q-factors of $\{X_k\}$ with respect to some norm $||.||$ in R^n. Since from now

on, $||.||$ will denote the L_2 vector norm defined by

$$||V|| = (\sum_{i=1}^{n} V_i^2)^{1/2}$$

for $V = (V_1, V_2, \ldots, V_n)^T \in R^n$.

Definition 1.5: Let $C(\Omega, X^*)$ denote the set of all sequences with limit X^* generated by an iterative process Ω. Then

$$Q_p(\Omega, X^*) = \sup\{Q_p\{X_k\} | \{X_k\} \in (\Omega, X^*)\}, \quad 1 \leq p < \infty,$$

are the Q-factors of Ω at X^* with respect to the norm in which the $Q_p\{X_k\}$ are computed.

The following theorem, proved by Ortega and Rheinholt (47), shows that Q_p is an isotone function of p which takes on only the values 0 and ∞ except at possibly one point.

Theorem 1.1: Let $Q_p(\Omega, X^*)$, $P \in [1, \infty)$, denote the Q-factors of an iterative process at X^* in some fixed norm on R^n. Then exactly one of the following conditions holds:

(a) $Q_p(\Omega, X^*) = 0, \forall\, p \in [1, \infty)$;

(b) $Q_p(\Omega, X^*) = \infty, \forall\, p \in [1, \infty)$;

(c) there exists a $p_0 \in [1, \infty)$ such that $Q_p(\Omega, X^*) = 0, \forall$ $p \in [1, p_0)$, and $Q_p(\Omega, X^*) = \infty, \forall\, p \in (p_0, \infty)$.

Definition 1.6: Let Ω_1 and Ω_2 denote two iterative processes with the same limit point X^*, and let $Q_p(\Omega_1, X^*)$ and $Q_p(\Omega_2, X^*)$ be the corresponding Q-factors computed in the same norm on R^n. Then Ω_1 is Q-faster than Ω_2 at X^* if there is a $p \in [1, \infty)$ such that $Q_p(\Omega_1, X^*) < Q_p(\Omega_2, X^*)$.

The above concept of "Q-faster" depends on the norm in R^n, since the magnitude of $Q_p\{X_k\}$ for an arbitrary convergent sequence $\{X_k\}$ such that $0 < Q_p\{X_k\} < +\infty$ is dependent on the norm. Now we state a property which is independent of the norm.

Definition 1.7: Let $Q_p(\Omega, X^*)$ be the Q-factors of the iterative process Ω at X^* in some norm on R^n. Then

$$O_Q(\Omega, X^*) = \begin{cases} +\infty & \text{if } Q_p(\Omega, X^*) = 0, \ \forall\, p \in [1, \infty) \\ \inf\{p \in [1, \infty) \mid Q_p(\Omega, X^*) = +\infty\}, & \text{otherwise} \end{cases}$$

is the Q-order of Ω at X^*.

Ortega and Rheinbolt proved the following theorem (47).

Theorem 1.2: Let $Q_p(\Omega, X^*)$ be the Q-factors of the iterative process Ω at X^*. Then the three relations $Q_p(\Omega, X^*) = 0$, $0 < Q_p(\Omega, X^*) < +\infty$ are independent of the norm on R^n. Hence, the Q-order of Ω at X^* is also independent of the norm.

From this theorem and Definition 1.6 we can list the following consequences.

Corollary 1: Let Ω_1 and Ω_2 be iterative processes with limit X^*. If

$$O_Q(\Omega_1, X^*) > O_Q(\Omega_2, X^*),$$

then Ω_1 is Q-faster than Ω_2 at X^* in every norm.

Corollary 2: Let Ω be an iterative process with limit X^*. If $Q_p(\Omega, X^*) < +\infty$ for some $p \in [1, \infty)$, then $O_Q(\Omega, X^*) \geq p$. If $Q_q(\Omega, X^*) > 0$

for some $q \in [1, \infty)$, then $O_Q(\Omega, X^*) \leq q$. Hence, if $0 < Q_p(\Omega, X^*) < +\infty$ for some $p \in [1, \infty)$, then $O_Q(\Omega, X^*) = p$.

When we compare two iterative processes Ω_1 and Ω_2 with the same limit X^*, we can do the following two stages. First, we compare the Q-orders $O_Q(\Omega_1, X^*)$ and $O_Q(\Omega_2, X^*)$; if they are different, the process with the larger Q-order is Q-faster than the other one in every norm. If $O_Q(\Omega_1, X^*) = O_Q(\Omega_2, X^*) = p$, then we compare the two Q-factors. If, say, $Q_p(\Omega_1, X^*) = 0 < Q_p(\Omega_2, X^*)$, or, if $Q_p(\Omega_1, X^*) < Q_p(\Omega_2, X^*) = +\infty$, then Ω_1 is Q-faster than Ω_2 in every norm. If $0 < Q_p(\Omega_1, X^*) < Q_p(\Omega_2, X^*) < +\infty$ in some norm, then Ω_1 is Q-faster than Ω_2 in that norm, but there may exist other norms in which the relation is reversed.

Definition 1.8: Let Ω be an iterative process with limit X^*.

(i) If $Q_1(\Omega, X^*) = 0$, then the process is Q-superlinearly convergent at X^*.

(ii) If $0 < Q_1(\Omega, X^*) < 1$ in some norm, the convergence is called Q-linear at X^*.

(iii) If $Q_2(\Omega, X^*) = 0$, then the process is Q-superquadratically convergent at X^*.

(iv) If $0 < Q_2(\Omega, X^*) < 1$ in some norm, the convergent is called Q-quadratic at X^*.

Review of Literature

Two basic quasi-Newton methods have been seriously proposed to solve general systems of nonlinear simultaneous equations. The secant method was proposed by J. G. P. Barnes (3) in 1965. The updating formula for H_k in Definition 1.2 is

$$H_k = \frac{(S_k - H_k Y_k) q_k^T}{q_k^T Y_k}$$

where q_k is given by

$$q_k^T Y_j = 0, \qquad k - n + 1 \leq j \leq k - 1$$

in solving the system of Equation (1.1).

Barnes (3) has proved that this method is equivalent to the generalized secant method described by Bittner (4) and Wolfe (58). This algorithm also possesses the property of linear termination (18); that is, it will solve a set of n linear equations in at most n + 1 steps. Computer experiments have shown that for suitable problems the method is considerably superior to the Newton-Raphson method (3).

In practice, for all but the most trivial problems, the n consecutive Y's in the secant method will become linearly dependent. This makes the secant method notoriously unstable.

C. G. Broyden (13) proposed another quasi-Newton method to solve Equation (1.1) in 1965. This method, unlike the secant method, possesses no termination property and thus can at most give an approximate solution to a linear system. The rate of convergence is Q-superlinear (19). If, on the other hand, B_k approximates the Jacobian J_k sufficiently well, the method is numerically stable. The detailed algorithm of Broyden's method and some other properties will be presented in the next chapter.

In 1968, C. G. Broyden (15) combined a particular form of Broyden's method (1965) with a particular form of Davidenko's method (22) to develop another method for the solution of Equation (1.1). Essentially, an auxiliary function $g(X, \theta)$, where θ is some scalar parameter, is constructed such that

$$g(X, 0) = F(X)$$

and the equation

$$g(X, 1) = 0$$

has a known solution. If X is a solution of

$$g(X, \theta) = 0$$

then X is a function of θ and by reducing θ incrementally from 1 to 0 a series of intermediate problems is constructed. This method uses a quasi-Newton method to solve the intermediate problems and suggests an improvement to Broyden's method made possible by the knowledge that a good initial estimate of the solution is available. Since X is not a continuous function of θ, for some value of θ the Jacobian of F of some problems may become singular. One such example is the polynomial equation due to Freudenstein and Roth (1963) with θ between 0.418 and 0.368.

Another algorithm was proposed by Brown (5) in 1966. This method is a variation of Newton's method incorporating Gaussian elimination in such a way that the most recent information is always used at each step of the algorithm. Basically the technique consists in expanding the first equation in a Taylor series about the starting guess, retaining only linear terms, equating to zero and solving for one variable, say X_k, as a linear combination of the reaming n - 1 variables. In the second equation, X_k is eliminated by replacing it with its linear representation found above, and again the same process is performed. One continues in this fashion, eliminating one variable per equation, until for the n^{th} equation, we are left with one equation in one unknown. A single Newton Step is now performed, followed by back-substitution in the triangularized linear system generated for the X_i's. This method is roughly

quadratically convergent and requires only $(n^2 + 3n)/2$ function evaluations per iterative step as compared with $n^2 + n$ evaluations for Newton's method. Details of the algorithm will be presented in Chapter III.

In 1975, W. C. Davidon (25) provided a variable metric algorithm for minimization calculations. Numerical experiments with Davidon's algorithm indicate that it may be the best numerical method for calculating the least value of a differentiable function of several variables. M. J. D. Powell (52) gave a new and elementary proof of the quadratic termination property without line search in 1976. Powell's proof does not require the frequent use of projection operations, i.e., "updating H with projections of the change in the gradient and the change in X," which is the part of the new algorithm that achieves quadratic termination with line searches.

In 1977, D. M. Gay and R. B. Schnabel (38) applied this "projection" concept to modify Broyden's algorithm to solve systems of nonlinear equations. The convergence rate is Q-superlinear. This algorithm will be one of the major parts to be discussed in Chapter II. The performance compared with some other algorithms will be presented in Chapter IV.

Davidon's original variable metric method (23) was proposed in 1959. This is the first quasi-Newton method for minimizing functions. With an initial point X and a positive definite trial matrix H, Davidon defined a new point $X^* = X - \lambda H\phi'(X)$, where $\phi'(X)$ is the gradient of $\phi(X)$ to be minimized, the scalar $\lambda > 0$ is chosen to minimize ϕ in the direction $- H\phi'(X)$. After making the change in X, the trial matrix H is improved by the relation between changes in X and changes in the gradient. Hence a sequence of points is generated to approach the minimum by repeating the iterative process.

The searching direction of Davidon's method is actually a downhill direction, i.e., the direction of steepest descent from X modified by the positive definite matrix H. The method of steepest descent for unconstrained minimization can be traced back to the work of the well-known French mathematician, A. Cauchy (20), in 1847. Characteristically the steepest descent path consists of a long first step followed by a sequence of short zig-zag steps. Davidon used some empirical devices to update the matrix H at each iteration in an effort to make the direction of steepest descent toward a minimum.

While Davidon's method was not widely publicized, R. Fletcher and M. J. D. Powell (34) published a powerful method with rapid convergence which is a simplified version of Davidon's method. This method is known as the Davidon-Fletcher-Power (DFP) method.

As in Davidon's method, DFP uses the same searching method to generate the next point X^* from the current point X, but the positive definite matrix H is updated by adding a symmetric correction matrix of rank two, defined in terms of H, the change in X and the change in the gradient. Numerical examples have shown that this method is generally successful in practice.

In 1967, C. G. Broyden (14) developed a family of quasi-Newton methods for minimization problems. Broyden updated the inverse Hessian matrix by a correction matrix such that this new approximation to the inverse Hessian satisfies the quasi-Newton equation. Since the correction matrix cannot be uniquely determined, families of methods can be obtained.

A similar development was given by D. F. Shanno (55) in 1970. At the same time, D. Goldfarb (39) used a variational approach to combine

two correction matrices, which belong to a family derived by J. Greenstadt (40), to form a family of update formula. These families are equivalent to Broyden's family (1967) and usually regarded as the general case of the DFP method. R. Fletcher (33) published another family of correction matrices in the same year. This algorithm was developed by the combination of the DFP correction matrix and the one from its inverse update form.

The convergence property of the members of Broyden's family has been studied by M. J. D. Powell (1972) and L. C. W. Dixon (32).

The organization of this study will be as follows: Broyden's method and the modification of Broyden's method, Gay-Schnabel's Algorithm, will be presented in Chapter II. Chapter III will discuss the development, properties of Brown's method and the DFP method whose performance will be compared with that of Gay-Schnabel's method. The result of this comparison with selected problems will be presented in Chapter IV. The conclusions considered in this study will be summarized in Chapter V.

Henceforth, $||.||_F$ will denote the Frobenius matrix norm;

$$||A||_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2} \quad \text{for} \quad A = (a_{ij})_{nxn} \in R^{nxn}.$$

CHAPTER II

GAY-SCHNABEL ALGORITHM--BROYDEN'S METHOD WITH PROJECTED UPDATES

A new algorithm to solve a system of n nonlinear equations in n unknowns was proposed by Gay and Schnabel (38) in 1977. This algorithm is a modification of the class of Broyden's method (13) which is mainly to approximate the Jacobian of Equation (1.1) to determine a searching direction at each iteration and then to generate a new point to form a sequence $\{X_k\}$, $k = 0, 1, 2, \ldots$, of approximations to a solution X^* of Equation (1.1). Actually, this Gay-Schnabel algorithm belongs to a class of quasi-Newton methods and has close relation to Broyden's method. We discuss Broyden's method first.

Broyden

The first quasi-Newton method was introduced by W. C. Davidon (23) in 1959 and simplified and published by R. Fletcher and M. J. D. Powell (34) in 1963, but Broyden (13) was the first in this area to deal with the solution of general sets of nonlinear Equation (1.1). The basic concepts of the quasi-Newton method, discussed in Chapter I, can also be applied to Broyden's method.

The Broyden class of single rank methods is defined according to the following algorithm.

Algorithm 2.1 (Broyden, 1965): Let $X_0 \in R^n$, $B_0 \in R^{nxn}$, $F : R^n \to R^n$, $\varepsilon > 0$ be given.

For $k = 0, 1, 2, \ldots,$

Choose nonzero $S_k = -\lambda_k H_k F(X_k)$

$$X_{k+1} = X_k + S_k \tag{2.1}$$

If $||F(X_{k+1})|| < \varepsilon$ then stop.

$$Y_k = F(X_{k+1}) - F(X_k) \tag{2.2}$$

Choose $d_k \varepsilon R^n - S_k$,

where S_k is the orthogonal complement of S_k.

$$B_{k+1} = B_k - \frac{(B_k S_k - Y_k) d_k^T}{d_k^T S_k} \tag{2.3}$$

Clearly, Equation (2.3) satisfies Equation (1.9) and hence Algorithm 2.1 is a direct update method of rank 1. The λ_k, $k = 0, 1, \ldots,$ is usually chosen to reduce $||F(X_k)||$ (15) or is set to unity (16) and $||.||$ denotes the L_2 norm.

Broyden singled out two specific choices of d_k. One method is $d_k = S_k$, a rather obvious choice in view of the requirement that $d_k^T S_k = 0$. This is known as Broyden's first method with direct update (15).

For computing purposes, it is preferable to store H_k, the inverse of B_k, in the computer and it is possible, using Householder's modification formula, to compute H_{k+1} with very little labor from H_k. Householder's formula states that if A is a nonsingular n x n matrix and X and Y are two n x 1 vectors such that $A + XY^T$ is nonsingular, then

$$(A + XY^T)^{-1} = A^{-1} - \frac{A^{-1} X Y^T A^{-1}}{1 + Y^T A^{-1} X} \tag{2.4}$$

Replacing d_k by S_k in Equation (2.3) and applying Equation (2.4) gives

$$H_{k+1} = H_k + \frac{(S_k - H_k Y_k)\ S_k^T H_k}{S_k^T H_k Y_k}. \tag{2.5}$$

This is usually called Broyden's first method with inverse update (15).

The second choice of Broyden's class is made by setting $H_k^T S_k = Y_k$ in Equation (2.5). This gives

$$H_{k+1} = H_k + \frac{(S_k - H_k Y_k)\ Y_k^T}{Y_k^T Y_k}. \tag{2.6}$$

If we replace Equation (2.3) by Equation (2.6), then Algorithm 2.1 is in another version, which is called Broyden's second method with inverse update (15).

According to Broyden's experiments (13), Broyden's first method is superior to Broyden's second method, if a reasonably good initial estimate of the solution is available.

Assume $A = J(X^*)$, the Jacobian matrix of $F(X)$ evaluated at the root X^*, is nonsingular; define the matrix error

$$E_k = B_k - A \tag{2.7}$$

where B_k is the current approximation to A. If we consider applying this algorithm (e.g., first method with direct update) to a system of linear equations $F(X) = AX - b = 0$, where A, the Jacobian of F, is a constant $n \times n$ matrix and b is a n vector, then we can recognize that the error matrix E_k decreases monotonically with respect to the spectral norm. It is trivial to see, since by Equations (2.1) and (2.2)

$$Y_k = AS_k,$$

that if B_{k+1} is obtained from B_k using Equation (2.3) taking $d_k = S_k$, then

$$E_{k+1} = E_k\left(I - \frac{S_k S_k^T}{S_k^T S_k}\right). \tag{2.8}$$

But the matrix

$$\left(I - \frac{S_k S_k^T}{S_k^T S_k}\right)$$

is symmetric and has n - 1 eigenvalues equal to unity and one equal to zero. Its spectral norm is thus unity since the norm of the product of two matrices is less than or equal to the product of their norms. In this sense B_{k+1} is not a worse approximation to A than B_k.

Convergence

A significant property of Newton's method,

$$X_{k+1} = X_k - J_k^{-1} F_k \qquad k = 0, 1, 2, \ldots \tag{2.9}$$

is the rapid convergence if one starts from a good initial point X_0. This fact is based on the following theorem proved by Dennis (29).

Theorem 2.1: If $F : R^n \to R^n$ is continuously differentiable in a neighborhood of some $X^* \in R^n$ for which $F(X^*) = 0$ and $J(X^*)$, the Jacobian of F at X^*, is nonsingular, then there exists a positive constant ε such that if $||X_0 - X^*|| < \varepsilon$ then the sequence $\{X_k\}$ defined by Equation (2.9) exists and converges to X^*. Moreover, if there is some $M \geq 0$ for which $||X - X^*|| < \varepsilon$ implies $||J(X) - J(X^*)||_F \leq M||X - X^*||$, then

$$\lim_{k \to \infty} ||X_{k+1} - X^*||/||X_k - X^*||^2 < \infty$$

i.e., the sequence $\{X_k\}$ converges Q-quadratically.

Q-quadratic convergence is quite fast; it implies roughly a doubling of the number of significant digits in the approximate solution at each iteration. One can go from an approximation accurate to just 1 digit to an approximation accurate to 16 digits in only four applications of the iterative process.

As one of the practical problems of using Newton's method is its frequent failure to converge from a poor initial estimate of the solution, people usually implement Newton's method in the form of Equation (1.4) by introducing a scalar parameter λ_k to enlarge the domain of convergence. Since Broyden's method uses Equation (1.4) as a major form to approximate the solution of $F(X) = 0$, except that the Jacobian J_k will be approximated by an appropriate form updated at each iteration as in Algorithm 2.1, we discuss several suggestions for the choice of λ_k.

Broyden (13) first suggested choosing λ_k by minimizing the norm of F_{k+1}. This choice of λ_k gives the greatest immediate reduction of the norm and hence the greatest improvement to the approximate solution. Making this choice, one needs to evaluate the vector function F a number of times. This means an increase in the amount of computation required compared with an alternative strategy of choosing a value λ_k which merely reduces the norm. Although norm reduction does not give as good an immediate improvement to the solution as norm minimization, the result of Broyden's tests (13) really means that less work is involved and that norm reduction is a better strategy than norm minimization.

Unfortunately, the strategy to obtain a monotonically decreasing sequence of norms of F may seriously inhibit convergence to the solution (16), and there is no obvious way of detecting the transition from the one state to the other. Experimental evidence reported in Reference (15)

shows that this does indeed occur when using Broyden's algorithm. Broyden found that the choice of λ_k to be unity was far superior provided that good initial approximations both to the solution and to the Jacobian could be found (16). Also at least two other algorithms, proposed by Barnes (24) and Davidon (31), suggested choosing λ_k to be unity. Dennis (27) has shown that Brown's algorithm is locally convergent for $\lambda_k = 1$.

Theorem 2.2: Let X^* be a zero of $F : R^n \to R^n$ which is continuously differentiable. Let the first partial derivatives satisfy a Lipschitz condition of order 1 in some open set containing X^* and $J(X^*)$, the Jacobian of F at X^*, is nonsingular. Under these hypotheses, there is an $\varepsilon > 0$ and a $\delta > 0$ such that if $X_0 \varepsilon R^n$ and $B_0 \varepsilon R^{nxn}$ satisfying $||X_0 - X^*|| < \varepsilon$ and $||B_o - J(X_0)||_F < \delta$, then the sequence $\{X_k\}$ defined by Algorithm 2.1 with $\lambda_k \equiv 1$ for all k converges to X^* from X_0 with $||X_{k+1} - X^*|| < \alpha ||X_k - X^*||$ for some $\alpha > 1$ and every k.

It would seem reasonable to suggest the strategy of choosing the step-size parameter λ_k by norm reduction until one feels he has a good approximate root and then switch to $\lambda_k \equiv 1$.

The rate of convergence revealed by this theorem was only Q-linear. In practice the iteration of Algorithm 2.1 seemed much better. The rate of convergence of Broyden's method is Q-superlinear. This fact has been proved by Broyden, Dennis and Moré (19).

Theorem 2.3: Let $F : R^n \to R^n$ be continuously differentiable in an open convex set D and $X^* \varepsilon D$ be a root of $F(X) = 0$. Assume that $j(X^*)$, the Jacobian of F at X^*, is nonsingular and that for some $M > 0$ and $r > 0$, $J(X)$ satisfies a Lipschitz condition,

$$||J(X) - J(X^*)||_F < M||X - X^*||^r$$

of order r in a neighborhood of X^*. Under these conditions there exists an $\varepsilon_1 > 0$ and an $\varepsilon_2 > 0$ such that if $||X_0 - X^*|| < \varepsilon_1$ and $||B_0 - J(X^*)||_F < \varepsilon_2$, then the sequence $\{X_k\}$ generated by Algorithm 2.1 exists and converges Q-superlinearly to X^*.

The class of Broyden's method converges Q-superlinearly but does not have the consistency property; i.e., let $\{X_k\}$, $\{B_k\}$ be sequences as defined by Algorithm 2.1 and X^* be a zero of F, if $\{X_k\}$ converges to X^* then $\{B_k\}$ converges to $J(X^*)$, the Jacobian of F at X^*.

This consistency condition has been a standard technique for proving that a method is Q-superlinearly convergent. It is known that this consistency condition is sufficient but not necessary (30). According to Powell's suggestion, Dennis and Moré (30) outlined a way to construct a counter example to support the "sufficient but not necessary" statement.

Define $F : R^n \to R^n$ such that $f_2, f_3, \ldots, f_n$ are independent of ξ_1 and $f_1(X) = \xi_1$. Let X_0 have a zero in its first coordinate and B_0 have zeros in its first row and first column except for the (1, 1) element. If $\{X_k\}$ and $\{B_k\}$ are generated by Algorithm 2.1, then the first row and first column of B_k remains unchanged while the rest of B_k is the matrix generated by the Broyden method when applied to $f_2, \ldots, f_n$ as a function of the n - 1 variables $(\xi_2, \ldots, \xi_n)$. In particular, the sequence $\{B_k\}$ does not converge to $J(X)$ for any $X \in R^n$. A numerical example is shown below.

Example 2.1: Let $X = (\xi_1, \xi_2)^T$, $F(X) = (\xi_1, 2\xi_2)^T$, and $X^* = (0, 0)^T$. Thus,

$$J(X) = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \qquad J(X^*) = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

consider Algorithm 2.1, with $S_k = -B_k^{-1}F_k$, starting from

$$X_0 = (0, \varepsilon)^T, \qquad B_0 = \begin{bmatrix} 1+\varepsilon & 0 \\ 0 & 2+\varepsilon \end{bmatrix}$$

Apply Theorem 2.3 to this problem. Clearly it satisfies the condition

$$||X - X^*|| < \delta \Rightarrow ||J(X) - J(X^*)||_F \leq M||X - X^*||^r \text{ with } M = r = 1$$

for arbitrary $\delta > 0$ since $||J(X) - J(X^*)||_F = 0$. Furthermore, we choose $\varepsilon_1 > \varepsilon$ and $\varepsilon_2 > \sqrt{2}\,\varepsilon$, then

$$||X_0 - X^*|| = \varepsilon < \varepsilon_1 \quad \text{and} \quad ||B_0 - J(X^*)||_F = \sqrt{2}\,\varepsilon < \varepsilon_2$$

and hence the sequence $\{X_k\}$ generated by Algorithm 2.1 converges to X^* Q-superlinearly. Actually, the intermediate steps are follows:

Iteration 1:

$$S_0 = \left(0, \frac{-2\,\varepsilon}{2+\varepsilon}\right)^T, \quad X_1 = \left(0, \frac{\varepsilon^2}{2+\varepsilon}\right)^T, \quad B_1 = \begin{bmatrix} 1+\varepsilon & 0 \\ 0 & 2 \end{bmatrix}$$

Iteration 2:

$$S_1 = \left(0, -\frac{\varepsilon^2}{2+\varepsilon}\right)^T, \quad X_2 = (0, 0)^T, \quad B_2 = \begin{bmatrix} 1+\varepsilon & 0 \\ 0 & 2 \end{bmatrix}$$

Clearly, $X_2 = X^*$ but $B_2 \neq J(X^*)$.

Broyden's update form of single-rank methods given by Equation (2.3) is

$$B_{k+1} = B_k - \frac{(B_k S_k - Y_k) d_k^T}{d_k^T S_k}$$

for any vector $d_k \in R^n$ such that $d_k^T S_k \neq 0$.

Gay and Schnabel propose to modify Broyden's update form to build a better approximation B_{k+1} to the Jacobian matrix from a given matrix B_k at each iteration.

This new approximation B_{k+1} not only satisfies the quasi-Newton equation $B_{k+1} S_k = Y_k$, where S_k and Y_k are defined as Equations (1.5) and (1.6), respectively, but also retains some good information learned through previous iterations, such as

$$B_{k+1} S_j = Y_j \quad \text{and} \quad B_k S_j = Y_j \quad \text{for all } j < k. \tag{2.10}$$

The reason that Equation (2.10) is good can be seen by considering a linear problem $F(X) = AX + b$ where A is nonsingular. After the first iteration, while applying Broyden's method, we will have $B_1 S_0 = Y_0$ $(= AS_0$ for a linear problem); after the next iteration we will have $B_2 S_1 = Y_1$ $(= AS_1)$, but not in general $B_2 S_0 = Y_0$, since B_i may never equal A, even though $F'(X) = A$ for all X_i. If one can retain those properties of Equation (2.2), while updating the new matrix B_{k+1}, it will be consistent, to some extent at least, with the properties that A $(= F'(X))$ possessed.

Gay and Schnabel accomplished this by a proper choice of d_k in Equation (2.3) denoted by $\hat{S}_k$. The updated B_{k+1} will then minimize the

Frobenius norm of $B_{k+1} - B_k$ among all B_{k+1} satisfying $B_{k+1} = Y_k$ and $(B_{k+1} - B_k)S_j = 0$ for $j < k$. The solution is based on the following theorem. Its proof is similar to Dennis and More's (31) proof that Broyden's method is the least-change update among all B_{k+1} satisfying $B_{k+1}S_k = Y_k$.

Theorem 2.4: Let $B \in R^{nxn}$ and S, Y be non-zero vectors in R^n with $BS \neq Y$. Let Z be an m-dimensional subspace of R^n, $m < n$. Then for $||.||_F$, the Frobenius norm, a solution to

$$\min \{||\bar{B} - B||_F : \bar{B}S = Y, \ (\bar{B} - B)\, z = 0 \ \text{ for all } z \in Z\}$$

is

$$\hat{B} = B - \frac{(BS - Y)\,\hat{S}}{\hat{S}\,S}$$

where $\hat{S}$ is the orthogonal projection of S onto the orthogonal complement of Z, i.e.,

$$S = S - \sum_{i=1}^{m} \frac{S^T Z_i}{Z_i^T Z_i} z_i$$

with $Z_1, Z_2, \ldots, Z_m$ an orthogonal basis for Z. The solution is unique in the Frobenius norm.

Based on the above theorem, Gay and Schnabel choose the vector $\hat{S}_k$ to be the orthogonal projection of S_k onto the orthogonal complement of the subspace spanned by $\{\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_{k-1}\}$, defined by

$$\hat{S}_k = \begin{cases} S_0 & \text{if } k = 0 \\ S_k - \sum_{j=0}^{k-1} \frac{S_k^T S_j}{S_j^T S_j} S_j & \text{if } k > 1 \end{cases} \qquad (2.11)$$

As a matter of fact, the projection $\hat{S}_k$ of S_k orthogonal to the subspace spanned by $\hat{S}_0, \hat{S}_i, \ldots, \hat{S}_{k-1}$ may be the zero vector for some $k \leq n$, the dimension of the space. Gay and Schnabel use a "restart criterion" by setting $\hat{S}_k = S_k$ if $\hat{S}_k$ is too small compared with S_k, i.e., when S_k is sufficiently close to lying entirely within the subspace spanned by $\{\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_{k-1}\}$. This must happen at least every n steps.

Algorithm 2.2 (Gay-Schnabel, 1977): Let $X_0 \in R^n$, $B_0 \in R^{nxn}$, $F : R^n \to R^n$, $\varepsilon > 0$, $\tau > 1$ be given.

Set $\ell_{-1} = 0$.

For K = 0, 1, 2, ...

Choose non-zero $S_k \in R^n$ (likely $S_k = -\lambda_k B_k^{-1} F(X_k)$)

$X_{k+1} = X_k + S_k$

IF $||F(X_{k+1})|| < \varepsilon$ then stop.

$Y_k = F(X_{k+1}) - F(X_k)$

$$Q' = \sum_{j=\ell_{k-1}}^{k-1} \frac{\hat{S}_j \hat{S}_j^T}{\hat{S}_j^T \hat{S}_j}$$

IF $||S_k|| \geq \tau \, ||S_k - Q_k' S_k||$

THEN ($\hat{S}_k = S_k$ and $\ell_k = k$)

ELSE ($\hat{S}_k = S_k - Q_k' S_k$ and $\ell_k = \ell_{k-1}$)

$$B_{k+1} = B_k - \frac{(B_k S_k - Y_k)\hat{S}_k^T}{\hat{S}_k^T S_k} \qquad (2.12)$$

Figure 1 shows an example of the relation between the first three

steps $\{S_0, S_1, S_2\}$ and $\{\hat{S}_0, \hat{S}_1, \hat{S}_2\}$ generated by Algorithm 2.2 in a three-dimensional space.

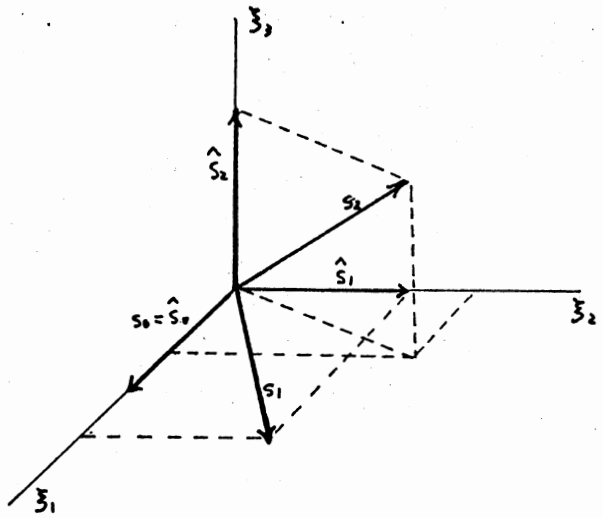

Figure 1. Relation Between $\{S_0, S_1, S_2\}$ and $\{\hat{S}_0, \hat{S}_1, \hat{S}_2\}$ in a Three-Dimensional Space

Several basic properties of these $\hat{S}_k$'s and S_k's have been developed from this algorithm.

Each $\hat{S}_k$ is defined to be the orthogonal projection of S_k onto the orthogonal complement of the subspace spanned by $\{\hat{S}_{\ell_1}, \ldots, \hat{S}_{k-1}\}$ since the last restart. So $\hat{S}_k$ is orthogonal to $\hat{S}_j$ for $j = \ell_1, \ldots, k - 1$.

$\hat{S}_k$, being a linear combination of $\{S_{\ell_1}, \ldots, S_{k-1}, S_k\}$ since the last restart, lies entirely in the subspace spanned by $\{S_{\ell_1}, \ldots, S_{k-1}, S_k\}$ for $k = 0, 1, 2, \ldots$. $\hat{S}_k$ is also orthogonal to S_{k-1}, and consequently $\hat{S}_k$ is orthogonal to S_j for each $j = \ell_1, \ldots, k - 1$.

$\hat{S}_k$ is the orthogonal projection of S_k onto the orthogonal complement of the subspace spanned by $\{\hat{S}_{\ell_1}, \ldots, \hat{S}_{k-1}\}$ since the last restart. The magnitude of $\hat{S}_k$ equals $||S_k||\cos\theta_k$, where θ_k is the angle between $\hat{S}_k$ and S_k. Thus, by vector analysis, $\hat{S}_k^T S_k = ||\hat{S}_k|| \cdot ||S_k|| \cos\theta_k = ||\hat{S}_k|| = \hat{S}_k^T \hat{S}_k$ for each k.

If we define the sequence $\{\hat{Y}_j\}_0^k$ to be

$$\hat{Y}_j = \begin{cases} Y_j & \text{if } \hat{S}_j = S_j \\ Y_j - B_j Q_j S_j & \text{otherwise} \end{cases} \tag{2.13}$$

for j = 0, 1, 2, . . ., k, an additional property, $B_{k+1}\hat{S}_j = \hat{Y}_j$ for $j = \ell_k, \ldots, k$ since the last restart, can be proved by Equation (2.12) and previous properties, i.e.,

$$B_{k+1}\hat{S}_k = B_k\hat{S}_k - \frac{(B_kS_k - Y_k)\,\hat{S}_k^T\hat{S}_k}{\hat{S}_k^T S_k}$$

$$= B_k\hat{S}_k - \frac{(B_kS_k - Y_k)\hat{S}_k^T\hat{S}_k}{\hat{S}_k^T S_k}$$

$$= B_k\hat{S}_k + Y_k - B_kS_k$$

If $\hat{S}_k = S_k$, then $\hat{Y}_k = Y_k$ and the above form gives $B_{k+1}\hat{S}_k = \hat{Y}_k$ with a restart. Otherwise, $\hat{S}_k = S_k - Q_kS_k$

$$B_{k+1}\hat{S}_k = B_k\hat{S}_k + (Y_k - B_kQ_kS_k) - B_k(S_k - Q_kS_k)$$

$$= B_k\hat{S}_k + \hat{Y}_k - B_k\hat{S}_k = \hat{Y}_k \tag{2.14}$$

By $\hat{S}_k^T\hat{S}_j = 0$ for $j = \ell_k, \ldots, k-1$ and Equation (2.12) we have

$$B_{k+1}\hat{S}_j = B_k\hat{S}_j \quad \text{for } j = \ell_k, \ldots, k-1. \tag{2.15}$$

Applying Equations (2.14) and (2.15) recursively gives

$$B_{k+1}\hat{S}_j = \hat{Y}_j \quad \text{for } j = \ell_k, \ldots, k-1.$$

We summarize the above properties in the following theorem. The analytical proof was given by Gay and Schnabel.

Theorem 2.5: Given $X_0 \in R^n$, $B_0 \in R^{nxn}$, $F: R^n \to R^n$, $\varepsilon > 0$, $\tau > 1$, let the sequences $\{S_i\}_0^k$, $\{B_i\}_0^{k+1}$ be generated by Algorithm 2.2. Define $\{\hat{S}_i\}_0^k$ as in Algorithm 2.2; let $\hat{Y}_j = Y_j$ if $\hat{S}_j = S_j$ and $\hat{Y}_j = Y_j - B_j Q_j S_j$; otherwise, $j = 0, 1, \ldots, k$. Then at each iteration k, $S_{\ell_k}, \ldots, S_k$ are linearly independent, B_{k+1} is well-defined, and

$$\hat{S}_k^T \hat{S}_j = 0 \qquad j = \ell_k, \ldots, k-1$$

$$\hat{S}_k^T S_j = 0 \qquad j = \ell_k, \ldots, k-1$$

$$\hat{S}_k^T \hat{S}_k = \hat{S}_k^T \hat{S}_k$$

$$B_{k+1} S_j = Y_j \qquad j = \ell_k, \ldots, k$$

$$B_{k+1} \hat{S}_j = \hat{Y}_j \qquad j = \ell_k, \ldots, k$$

$$||S_k|| < \tau \, ||\hat{S}_k||$$

$$k - \ell_k < n$$

As a conclusion of Theorem 2.5, $\hat{S}_k$ can be regarded as the projection of S_k orthogonal to the subspace spanned by $S_{\ell_k}, \ldots, S_{k-1}$ since the last restart.

Following Broyden's second method, Gay and Schnabel proposed an algorithm which updates approximations H_k to $J(X_k)^{-1}$, and choose $\hat{Y}_k$ to be the projection of Y_k ($Y_k = F_{k+1} - F_k$) orthogonal to the previous Y_j's in the following update form

$$H_{k+1} = H_k + \frac{(S_k - H_k Y_k)\hat{Y}_k^T}{\hat{Y}_k^T Y_k} . \tag{2.16}$$

Algorithm 2.3: Let $X_0 \in R^n$, $H_0 \in R^{nxn}$, $F : R^n \to R^n$, $\varepsilon > 0$, $\tau > 1$ be given.

Set $\ell_{-1} = 0$.

For k = 0, 1, 2, ...

Choose non-zero, $S_k \in R^n$ (likely $S_k = -\lambda_k H_k F_k$)

$X_{k+1} = X_k + S_k$

IF $||F(X_{k+1})|| < \varepsilon$ then stop.

$Y_k = F(X_{k+1}) - F(X_k)$

$$Q_k' = \sum_{j=\ell_{k-1}}^{k-1} \frac{\hat{Y}_j \hat{Y}_j^T}{\hat{Y}_k^T \hat{Y}_k}$$

IF $||Y_k|| \geq \tau \, ||Y_k - Q_k' Y_k||$

THEN $(\hat{Y}_k = Y_k$ and $\ell_k = k)$

ELSE $(\hat{Y}_k = Y_k - Q_k' Y_k$ and $\ell_k = \ell_{k-1})$

$$H_{k+1} = H_k + \frac{(S_k - H_k Y_k)\hat{Y}_k^T}{\hat{Y}_k^T Y_k}$$

Using Algorithm 2.3 we can prove a theorem analogous to Theorem 2.5. We state the results as follows.

Theorem 2.6: Given $X_0 \in R^n$, $H_0 \in R^{nxn}$, $F : R^n \to R^n$, $\varepsilon > 0$, $\tau > 1$, let the sequence $\{S_i\}_0^k$, $\{Y_i\}_0^k$, $\{H_i\}_0^{k+1}$ be generated by Algorithm 2.3.

Define $\{\hat{Y}_i\}_0^k$ as in Algorithm 2.3; let $\hat{S}_j = S_j$ if $\hat{Y}_j = Y_j$ ans $\hat{S}_j = S_j - H_j Q_j' Y_j$ otherwise, $j = 0, 1, \ldots, k$. Then at each iteration k, Y_{ℓ_k}, $\ldots$, Y_k are linearly independent, H_{k+1} is well defined, and

$$\hat{Y}_k^T \hat{Y}_j = 0, \qquad j = \ell_k, \ldots, k-1$$

$$\hat{Y}_k^T Y_j = 0, \qquad j = \ell_k, \ldots, k-1$$

$$\hat{Y}_k^T \hat{Y}_k = \hat{Y}_k^T \hat{Y}_k$$

$$H_{k+1} Y_j = S_j \qquad j = \ell_k, \ldots, k$$

$$H_{k+1} \hat{Y}_j = \hat{S}_j \qquad j = \ell_k, \ldots, k$$

$$||Y_k|| < \tau \, ||\hat{Y}_k||$$

$$k - \ell_k < n.$$

Convergence

Gay and Schnabel have analytically proved that their new algorithm is locally Q-superlinearly convergent under the same conditions used in the Broyden method. Incidentally, they found that their algorithm will always locate a zero for those n equations in n unknowns $F(X) = 0$ in $n+1$ or fewer iterations if any one or all of those n equations are linear with non-singular Jacobian matrix. Furthermore, if $k+1$ iterations are required, then $B_{k+1} - J(X^*)$, X^* is a zero of $F(X)$, has rank $n-k$. Broyden's method may take 2n stpes to locate the zero for a linear system of n equations with n variables and $B_{2n} - J(X^*)$ may have rank $n-1$ (37). As to Newton's method, $X_1 = X^*$ if F is linear with non-singular Jacobian. We first consider F as a linear case.

Theorem 2.7: Let $A \in R^{nxn}$ be non-singular; $b \in R^n$, and $F : R^n \to R^n$ defined by $F(X) = AX + b$. Consider Algorithm 2.2 acting on F, starting from any $X_0 \in R^n$ and $B_0 \in R^{nxn}$. If $S_0, \ldots, S_{n-1}$ are linearly independent, then $B_n = A$; and if $S_n = -B_n^{-1} F(X_n)$, then $F(X_{n+1}) = 0$. Moreover, if for some $k < n$, $S_0, \ldots, S_{k-1}$ are linearly independent, B_k^{-1} exists and $B_k^{-1} F(X_k) \in (S_0, \ldots, S_{k-1})$, and if $S_k = -B_k^{-1} F(X_k)$, then $F(X_{k+1}) = 0$.

From the above theorem, it can be recognized that if Algorithm 2.2 is acting on a linear problem with $n - m$ iterations required, and if $S_0, \ldots, S_{n-m-1}$ are linearly independent and no restarts have occurred, then B_{n-m} will agree with A ($= J(X_k)$, for each k) in $n - m$ directions, i.e., $A - B_{n-m}$ will have rank m.

The next theorem concerns the case that when some but not necessarily all of the component functions of F are linear, the number of iterations required to locate a zero X^* is also less than or equal to $n + 1$.

For ease of notation we use $(V_1, \ldots, V_k)$ to denote the subspace spanned by vectors $V_1, \ldots, V_k \in R^n$ and assume that the first m component functions of F are linear. The Jacobian of F will be constant in its first m rows, and we denote the approximations B_k by $\left(\frac{C_k}{D_k}\right)$, $C_k \in R^{mxn}$, $D_k \in R^{(n-m)xn}$.

Theorem 2.8: Let $A \in R^{mxn}$, $1 \leq m \leq n$; $b \in R^m$

$$F(X) = \begin{bmatrix} F_1(X) \\ F_2(X) \end{bmatrix} : R^n \to R^n \text{ with } F_1(X) = AX + b : R^n \to R^m$$

and $F_2 : R^n \to R^{n-m}$. Consider Algorithm 2.2 acting on F, starting from any $X_0 \in R^n$ and $B_0 \in R^{nxn}$. If for some $k \leq n$, $S_0, \ldots, S_{k-1}$ are linearly

independent, B_k^{-1} exists and $B_k^{-1} F(X_k) \in (S_0, \ldots, S_{k-1})$, then the choice $S_k = -B_k^{-1} F(X_k)$ leads to $F_1(X_{k+1}) = 0$. Furthermore, if $S_0, \ldots, S_{n-1}$ are linearly independent, then $C_n = A$.

Schnabel made a proof that the sequence $\{X_i\}$ generated by Algorithm 2.2 with $S_k = -B_k^{-1} F(X_k)$ converges Q-superlinearly to X^* if X_0 is close enough to X^* and if B_0 is close enough in norm to $J(X_0)$. This proof is analogous to the local superlinear convergence proof of Broyden, Dennis and Moré (1973) for Broyden's method, and the work of Dennis and Moré (1974) characterizing superlinear convergence.

Theorem 2.9: Let $F : R^n \to R^n$ be differentiable in an open convex set D, and assume for some $X^* \in D$ and $r > 0$, $C \geq 0$, that

$$||J(X) - J(X^*)||_F \leq C||X - X^*||^r$$

where $F(X^*) = 0$ and $J(X^*)$ is non-singular. Under these hypotheses, consider the sequences $\{X_i\}$, $\{B_i\}$, $X_i \in R^n$, $B_i \in R^{nxn}$ generated from X_0 and B_o by Algorithm 2.2. Then there exists $\varepsilon > 0$ and $\delta > 0$ such that for $||X_0 - X^*|| \leq \varepsilon$ and $||B_0 - J(X^*)||_F \leq \delta$, $\{X_i\}$ converges Q-superlinearly to X^* and $\{||B_k||\}$, $\{||B_k^{-1}||\}$ are bounded.

Using Algorithm 2.3 we can have the same convergence results for linear and general nonlinear functions F as are given in Theorems 2.7, 2.8, and 2.9.

The Gay-Schnabel Proposal

As we mentioned in the previous section, the Gay-Schnabel algorithms (Algorithm 2.2 and Algorithm 2.3) have been analytically proved to be locally Q-superlinearly convergent and exact on linear problems.

Now, we consider the Gay-Schnabel proposal. This proposal involves three parts: Extensions, Implementations, and Evaluations.

Extensions

The basic idea of Gay and Schnabel's modification of the Broyden method is the choice of a proper vector $\hat{S}_k \in R^n$ used to update a new approximation to the Jacobian at each iteration. Two variations of the Gay-Schnabel algorithm are derived by the different methods of setting $\hat{S}_k$ at each iteration. We identify the original Gay-Schnabel algorithm (Algorithm 2.2) as algorithm I and its inverse update method (Algorithm 2.3) as algorithm I'.

The first extension is to preserve the current and most recent quasi-Newton equation at each step by setting

$$\hat{S}_k = S_k - (S_{k-1}^T S_k / S_{k-1}^T S_{k-1})\, S_{k-1} \tag{2.17}$$

in Equation (2.12) of Algorithm I at each iteration.

This choice of $\hat{S}_k$ will cause the updated B_{k+1} to minimize the Frobenius norm of $B_{k+1} - B_k$ among all B_{k+1} satisfying $B_{k+1} S_k = Y_k$ and $(B_{k+1} - B_k)\, S_k = 0$. This extension can be proved to be of local Q-superlinear convergence without restarts. Gay and Schnabel never tested it. From now on we call this algorithm II.

Algorithm II is actually to choose $\hat{S}_k$ at each iteration to be the projection of S_k orthogonal to the previous step S_{k-1} instead of all the previous steps $S_0, S_1, \ldots, S_{k-1}$. This choice of $\hat{S}_k$ is a special case of the following extension.

The second extension is to choose $\hat{S}_k$ in the update form (Equation (2.12)) of B_{k+1} equal to the projection of S_k orthogonal to the previous

t S_j's, $t < n$, subject to the linear independence of S_{k-t}, $S_{k-t+1}, \cdots,$ S_k as in Algorithm I. Such an algorithm would require no restart. From now on, we call this algorithm III.

Algorithm II is then a special case of algorithm III with $t = 1$. We note that there is an inverse update form for each of algorithm II and algorithm III. These two inverse update forms will be denoted by algorithm II' and algorithm III', respectively.

Implementations

Since algorithm II and algorithm III have never been tested, we have implemented and tested these algorithms II, II', III, and III' and have done further tests of algorithms I and I'. The test problems will be covered in Chapter IV.

Evaluations

We will evaluate the performance of these algorithms by making a comparison between these algorithms and the conventional Broyden's method, as well as some other methods such as Brown's method for solving a system of equations and the DFP (Davidon-Fletcher-Powell) method for minimization problems. Criteria used will be convergence properties, function evaluations, computing time, and user's effort required. Brown's method and the DFP method will be covered in the following chapter.

CHAPTER III

OTHER ALGORITHMS AND COMPUTER EXPERIENCES

In this chapter the author will present two additional algorithms--Brown's method and the DFP (Davidon-Fletcher-Powell) method. The performance of these two algorithms will be compared with that of Gay-Schnabel's method in the next chapter. We first consider Brown's method to solve a system of n simultaneous nonlinear equations.

Brown

In order to reduce the amount of computational effort in solving the system of n nonlinear equations with n unknowns, Equation (1.1), Brown (5) through (8) proposed a local method which handles the equations $f_i = 0$ one at a time so that information obtained from working with f_1 can be incorporated when working with f_2, etc. The basic idea of Brown's method is to set up a successive substitution scheme by the expansion of previous f_i in the Taylor series about an initial point X_0. Then using this expansion form, set equal to zero, one solves a specified component of X of which the (approximate) partial derivative is the largest in absolute value, then substitutes the solved component into the next f_{i+1}. Repeating this process, one goes through all the f_i's, then uses a back-substitution method to solve the real values of those components which are now regarded as the components of a new point X_1. This is different from Newton's method, which treats all these f_i

simultaneously. Brown's method does not require the user to furnish any derivatives. The partial derivatives $\partial f_i/\partial \xi_j$ required in the Taylor expansion of f_i at a given point X_0 are furnished by the finite difference quotient approximation

$$\frac{f_i(X_0 + h^0 e_j) - f_i(X_0)}{h^0} \tag{3.1}$$

where e_j denotes the j^{th} unit vector and the scalar h^0 is usually chosen such that $h^0 = 0(||F(X_0)||)$. We will show the strategy of choosing this h^0 later so as to guarantee the convergence of Brown's method. Now we discuss the necessary steps that are required to derive a better approximation, $X_{k+1} = (\xi_1^{k+1}, \xi_2^{k+1}, \ldots, \xi_n^{k+1})$, to the solution $X^* = (\xi_1^*, \xi_2^*, \ldots, \xi_n^*)$ from a given point $X_k = (\xi_1^k, \xi_2^k, \ldots, \xi_n^k)$.

STEP 1: Express f_1 in an approximate Taylor series expansion about the point X_k and ignore terms of order 2 and higher,

$$f_1(X) = f_1(X_k) + \sum_{i=1}^{n} (\partial f_1/\partial \xi_i)(X_k)(\xi_i - \xi_i^k) \tag{3.2}$$

where $\partial f_1(X_k)/\partial \xi_i$ is defined by Equation (3.1). If X_k is close enough to X^*, $f_1(x) \sim 0$, and we can equate Equation (3.2) to zero and solve for one variable component, say ξ_n, whose corresponding approximate partial derivative, $\partial f_1(X_k)/\partial \xi_n$, is largest in absolute value.

$$\xi_n = \xi_n^k - \sum_{i=1}^{n-1} \left(\frac{\partial f_1(X_k)}{\partial \xi_i} \Big/ \frac{\partial f_1(X_k)}{\partial \xi_n}\right)(\xi_i - \xi_i^k) - f_1(X_k) \Big/ \frac{\partial f_1(X_k)}{\partial \xi_n} \tag{3.3}$$

Clearly, ξ_n is a linear function of the $n-1$ variable components $\xi_1, \xi_2, \ldots, \xi_{n-1}$ in Equation (3.3). We rename the left-hand side of Equation

(3.3) as $L_n(\xi_1, \xi_2, \ldots, \xi_{n-1})$. The constants $\frac{\partial f_1(X_k)}{\partial \xi_i} / \frac{\partial f_1(X_k)}{\partial \xi_n}$, $i = 1, 2, \ldots, n-1$, and $f_1(X_k) / \frac{\partial f_1(X_k)}{\partial \xi_n}$ are stored for future use in computer implementation.

STEP 2: Define $g_2(\xi_1, \xi_2, \ldots, \xi_{n-1}) = f_2(\xi_1, \xi_2, \ldots, \xi_{n-1}, L_n(\xi_1, \xi_2, \ldots, \xi_{n-1}))$. Expand g_2 about the point $(\xi_1^k, \xi_2^k, \ldots, \xi_{n-1})$, linearize (ignore higher order terms) and solve that variable component, say ξ_{n-1}, whose corresponding approximate partial derivative, $(\partial g_2/\partial \xi_{n-1})(\xi_1^k, \xi_2^k, \ldots, \xi_{n-1}^k)$, is largest in absolute value:

$$\xi_{n-1} = \xi_{n-1}^k - \sum_{i=1}^{n-2} \left(\frac{\partial g_2}{\partial \xi_i}(\xi_1^k, \ldots, \xi_{n-1}^k) / \frac{\partial g_2}{\partial \xi_{n-1}}(\xi_1^k, \ldots, \xi_{n-1}^k)\right) (\xi_i - \xi_i^k) - g_2(\xi_1^k, \ldots, \xi_{n-1}^k) / \frac{\partial f_2}{\partial \xi_{n-1}}(\xi_1^k, \ldots, \xi_{n-1}^k). \tag{3.4}$$

Here

$$\frac{\partial g_2}{\partial \xi_i}(\xi_1^k, \ldots, \xi_{n-1}^k)$$

is given by

$$\frac{g_2(\xi_1^k, \ldots, \xi_{i-1}^k, \xi_i^k + h^k \xi_{i+1}^k, \ldots, \xi_{n-1}^k) - g_2(\xi_1^k, \ldots, \xi_{n-1}^k)}{h^k}$$

Again, we denote the right-hand side of the linear function (Equation (3.4)) as $L_{n-1}(\xi_1, \xi_2, \ldots, \xi_{n-2})$. The ratios

$$\frac{\partial g_2}{\partial \xi_i} / \frac{\partial g_2}{\partial \xi_{n-1}} \quad \text{at} \quad (\xi_1^k, \ldots, \xi_{n-1}^k) \quad \text{for } i = 1, 2, \ldots, n-2$$

and

$$g_2(\xi_1^k, \ldots, \xi_{n-1}^k)/\frac{\partial g_2}{\partial \xi_{n-1}}(\xi_1^k, \ldots, \xi_{n-1}^k)$$

are stored for future use in any computer implementation.

STEP 3: Define $g_3(\xi_1, \xi_2, \ldots, \xi_{n-2}) = f_3(\xi_1, \xi_2, \ldots, \xi_{n-2}, L_{n-1}(\xi_1, \ldots, \xi_{n-2}), L_n(\xi_1, \ldots, \xi_{n-2}, L_{n-1}(\xi_1, \ldots, \xi_{n-2})))$. Expand g_3 with Taylor expansion about $(\xi_1^k, \xi_2^k, \ldots, \xi_{n-2}^k)$, linearization of the resulting expansion, equating to zero and solving for one variable component, say ξ_{n-2}, whose corresponding approximate partial derivative $\partial g_3(\xi_1^k, \xi_2^k, \ldots, \xi_{n-2}^k)/\partial \xi_{n-2}$ is largest in magnitude. Now, ξ_{n-2} is a linear combination of the remaining $n-3$ variable components; i.e., $\xi_{n-2} = L_{n-2}(\xi_1, \xi_2, \ldots, \xi_{n-3})$.

We continue in this fashion, eliminating one variable for each equation treated. Every time we obtain a new linear expression, L_{n-k} for one of the components, say ξ_{n-k}, in terms of the remaining $n-k-1$ variable components, $\xi_1, \xi_2, \ldots, \xi_{n-k-1}$, we use this linear expression wherever ξ_{n-k} had appeared in the previously defined linear expressions $L_{n-k+1}, L_{n-k+2}, \ldots, L_n$. On the other hand, we add one more linear expression to a linear system at each step. During the $(k+1)$st step of the algorithm, we need to evaluate g_{k+1}, i.e., f_{i+1} for various arguments. The values of the last k components of the variable X of f_{i+1} are obtained by the substitution of the linear system $L_{n-k+1}, L_{n-k+2}, \ldots, L_n$ which has been built up. The points needed are $(\xi_1^k, \ldots, \xi_{n-k}^k)$ and $(\xi_1^k, \ldots, \xi_{n-k}^k) + he_i$, $i = 1, \ldots, n-k$, where e_i denotes the i^{th} unit vector. These points are required to determine the quantities $g_{k+1}(\xi_1^k, \ldots, \xi_{n-k}^k)$ and $\partial g_{k+1}(\xi_1^k, \ldots, \xi_{n-k}^k)/\partial \xi_i$ for $i = 1, \ldots, n-k$, needed for the elimination of the $(k+1)$st component, say ξ_{n-k}, by the

basic processes of expansion, linearization, and solution of the resulting expression. For each k, this results in the (k + 1)st components, say ξ_{n-k}, being expressed as a linear combination, L_{n-k}, of the remaining n - k + 1 components.

STEP N: At this stage we have $g_n = (\xi_1, L_2, L_3, \ldots, L_n)$ where L_i's are obtained by back-substitution in the n - 1 rowed triangular linear system which now has the form

$$L_i = \xi_j^n - \sum_{i=1}^{j-1} \left(\frac{\partial g_{n-j+1}}{\partial \xi_i}(\xi_1^k, \ldots, \xi_j^k) \Big/ \frac{\partial g_{n-j+1}}{\partial \xi_j}(\xi_1^k, \ldots, \xi_j^k)\right)(\xi_i - \xi_i^k)$$

$$- g_{n-j+1}(\xi_1^k, \ldots, \xi_j^k) \Big/ \frac{\partial g_{n-j+1}}{\partial \xi_j}(\xi_1^k, \ldots, \xi_j^k), \qquad (3.5)$$

for $j = n, n-1, \ldots, 2$, with $g_1 = f_1$, $L_1 = \xi_1$. Thus g_n is a function of a single variable component ξ_1, giving

$$\xi_1 = \xi_1^k - g_n(\xi_1^k) \Big/ \frac{\partial g_n}{\partial \xi_1}(\xi_1^k). \qquad (3.6)$$

We use ξ_1 thus obtained as the next approximation, ξ_1^{k+1}, to the first component, ξ_1^*, of the solution vector X^*. We rename ξ_1 as L_1 in Equation (3.6) and back-solve the L_i system (Equation (3.5)) to get improved approximations to the other components of X^*.

Now, we use the notation

$$X_{k+1} <= \text{PROC}(X_k;\ S1 - SN)$$

to denote the above procedures that derive the point $X_{k+1} = (\xi_1^{k+1}, \ldots, \xi_n^{k+1})$ from a given point $X_k = (\xi_1^k, \ldots, \xi_n^k)$, by going through STEP 1 to STEP N. Brown's algorithm can be listed as follows.

Algorithm 3.1 (Brown, 1967): Let $X_0 \in R^n$, $F : R^n \to R^n$, $\varepsilon > 0$ be given. For k = 0, 1, 2, . . .

IF $||F(X)|| < \varepsilon$ then stop.

X_{k+1} <= PROC(X_k; SL - SN).

In applying Algorithm 3.1 to F = 0, an ordering strategy must be considered. The equations should be pre-ordered so that the linear ones, or most nearly linear ones, come first and then equations become progressively more nonlinear. This is because Brown's method works with one equation, $f_i = 0$, at a time and uses information thus obtained immediately when dealing with the next equation $f_{i+1} = 0$.

One question might be raised: how can we be sure that the solution procedure (Equation (3.3)) in STEP 1 will be defined? Brown (7) gave a proof in 1969 that under the usual hypotheses for Newton's method there will always be at least one non-zero partial derivative; consequently, the approximate partial Equation (3.1) will also be non-vanishing.

The choice of h^k in forming the approximate partial derivative needed in X_{k+1} <= PROC(X_k; Sl - SN) will affect the convergence of Brown's method. This fact is from the following local convergence theorem, which was proved by Brown and Dennis (10). The notation $||.||_\infty$ used in the theorem is defined as follows:

If $X = (\xi_1, \xi_2, \ldots, \xi_n)$,

then $||X||_\infty = \max\{|\varepsilon_i| : i = 1, 2, \ldots, n\}$

If $A = (a_{ij})_{nxn}$

then $||A||_\infty = \max\{\sum_{j=1}^{n} |a_{ij}| : i = 1, 2, \ldots, n\}$

Clearly,

$$||X||_2 < ||X||_\infty \qquad ||A||_F \leq ||A||_\infty$$

Theorem 3.1: Let $F : R^n \to R^n$ be continuously differentiable in an open convex set D, $X^* \in D$, $F(X^*) = 0$. Assume the Jacobian $J(X)$ of F is continuous in $\bar{N}(X^*; r) = \{X \in R^n : ||X - X^*||_\infty \leq r\}$ and $J(X^*)$ is non-singular. Under these hypotheses there exist positive numbers ε and δ, such that if $X_0 \in N(X^*; \delta)$ and $\{h^k\}$ is bounded in modulus by ε, then the sequence $\{X_k\}$ generated by Algorithm 3.1 exists and converges to X^*. Furthermore, if there is an $M \geq 0$ such that $||J(X) - J(X^*)||_\infty \leq k||X - X^*||_\infty$ for $||X - X^*||_\infty < r$ and $\{h^k\}$ is $O(\{|f_1(X^*)|\})$, then the convergence is Q-quadratic.

In an actual computer implementation, h_j^k, by which one increments ξ_j^k when working with f_i, is chosen according to the following strategy (8)

$$h_j^k = \max\{\alpha_{ij}^k; \ 5 \times 10^{-\beta+2}\}$$

where

$$\alpha_{kj}^k = \min\{\max(|f_1(\xi_1^k, \ldots, \xi_n^k)|, |g_2(\xi_1^k, \ldots, \xi_{n-1}^k)|, \ldots,$$

$$g_i(\xi_1^k, \ldots, \xi_{n-i})); \ 0.001 \times |\xi_j^k|\}$$

and β is the number of significant digits carried by the machine.

This choice of the amount of h^* can prevent the size of h^* from being absurd when compared to the magnitude of $||X^k||_\infty$, (suppose, for example, that $||X^k||_\infty = 0.001$ but $||f_1(X^k)||_\infty = 1000$; this would provide poor approximations to the partial derivatives) while at the same time satisfy the conditions of the theorem as the solution X^* is approached.

If we examine the process X_{k+1} <= PROC(X_k; S1 - SN) to count the number of fucntion evaluations, we can see that the first step requires n + 1 evalutions of f_1, the second step n evaluations of f_2, the third step n - 1 evaluations of f_3, etc., so the total number of function evaluations is

$$\sum_{k=2}^{n+1} k = (N^2 + 3N) / 2.$$

On the other hand, Brown's method adds a number of other functions to be evaluated, namely the linear functions L_k. The evaluations of the L_k are not included in the count above. Thus Brown's method is suitable to solve problems when the functions f_i are expensive to evaluate in terms of the amount of computation.

Computer experience (8) reveals that the method seems to be extremely stable locally in practice and has been used very successfully on at least 100 different problems, even in cases where the f_i are easy to evaluate. The implemented program used in the next chapter to compare the performance of this algorithm with that of Gay-Schnabel's algorithm is called from the IMSL (43) library. The name of the subroutine is ZSYSTM.

Davidon-Fletcher-Powell

The Davidon-Fletcher-Powell (DFP) method is a technique for finding an unconstrained minimum of a differentiable function $\phi(X)$ of n real variables. It was first proposed by Davidon (23) and later reformulated by Fletcher and Powell (34).

Basically, the DFP method is an iterative procedure. With a given point X and a positive definite symmetric matrix H, the DFP method

searches along a line from X in a direction of steepest descent, modified by this matrix H, to a new point X^* such that the function value is decreased. Then a new matrix H^* is updated according to a defined rule in terms of H, the change of X, and the change in the gradient. Now, X^* and H^* can be used to begin the next iteration. This generates a sequence of points which converge to a local minimum of $\phi(X)$. We use g to denote $V\phi$ and state Fletcher and Powell's original proposal as the following algorithm.

Algorithm 3.2 (Davidon-Fletcher-Powell, 1959, 1963): Given an initial point X_0 and an initial matrix $H_0 = I$ or any symmetric positive definite matrix.

For k = 0, 1, 2, . . . ,

If $g(X_k) = 0$, then stop.

Else, set $d_k = -H_k g(X_k)$ (3.7)

Obtain $\lambda_k > 0$ such that $f(X_k + \lambda_k d_k)$ is a minimum with respect to λ along $X_k + \lambda d_k$,

set $S_k = \lambda_k d_k$, (3.8)

$X_{k+1} = X_k + S_k$

$Y_k = g(X_{k+1}) - g(X_k)$ (3.9)

$H_{k+1} = H_k + A_k - B_k$ (3.10)

where

$$A_k = \frac{S_k S_k^T}{S_k^T Y_k} \tag{3.11}$$

and

$$B_k = \frac{H_k Y_k Y_k^T H_k}{Y_k^T H_k Y_k} \tag{3.12}$$

In the above algorithm, the quitting rule $g(X_k) = 0$ says that the current point X_k is a stationary point of the function ϕ to be minimized. This stationary point satisfies the necessary conditions for a local minimum. If ϕ has continuous second partial derivatives, then the stationary point X_k is a local minimum if the Hessian matrix of ϕ is positive definite. In the computer implementation, Fletcher and Powell recommended that the minimization be terminated if, on evaluating both the vectors--$H_k g(X_k)$ and $-\lambda_k H_k g(X_k)$--either of the following occurs:

1. Every component of the two vectors is less than a prescribed value.
2. The predicted lengths of each of the vectors from the minimum are less than a prescribed value.

As to the line search procedure to obtain a minimum along a direction, the algorithm uses cubic interpolation which is first given by Davidon (23).

For ease of notation, we use g_k to denote $g(X_k)$.

Two significant properties were derived by Fletcher and Powell (34). The first one is the stability of the algorithm, i.e., the value of the function to be minimized is decreased at each step. This property is desirable for descent methods, usually.

In Algorithm 3.2, g_k is the direction of steepest ascent. The direction d_k will be downhill if and only if

$$-d_k^T g_k = (H_k g_k)^T g_k = g_k^T H_k g_k > 0$$

i.e., the symmetric matrix H_k is positive definite for all nontrivial vectors g_k.

The above relation will be clear if we observe the first order Taylor series form for ϕ with a sufficiently small step $\lambda > 0$

$$\phi(X_k + \lambda d_k) = \phi(X_k) + \lambda d_k^T g_k.$$

Since $\lambda > 0$, this implies $\phi(X_k + \lambda d_k) < \phi(X_k)$ if and only if $-d_k^T g_k > 0$.

Fletcher and Powell have proved, by an inductive argument, that H_k in the DFP method defined in Algorithm 3.2 is positive definite and consequently the DFP method is stable.

The second property is the quadratic convergence which is usually known as quadratic termination, i.e., the algorithm, when applied to a strictly convex quadratic function of n variables, will find the minimum in at most n iterations. A termination property certainly guarantees efficiency when solving quadratic problems. It has often been regarded as a desirable property for an algorithm to possess. Let the function ϕ_q be given by

$$\phi_q = (1/2)X^T AX + b^T x + C, \tag{3.13}$$

where A is positive definite. ϕ_q is a strictly convex quadratic function and has a unique minimum. Clearly, A is the Hessian matrix of ϕ_q. If $H_k = A^{-1}$ for some $k \leq n$, then the search at the k^{th} iteration will find the minimum.

Fletcher and Powell proved this property by proving that the steps $S_0, S_1, \ldots, S_k$ generated by Algorithm 3.2 are linearly independent eigenvectors of $H_{k+1}A$ with eigenvalue unity. Therefore, it will follow that $H_n = A^{-1}$. The detailed proof involves establishing

$$S_i^T A S_j = 0, \qquad 0 \leq i < j < k \tag{3.14}$$

and

$$H_k A S_i = S_i, \qquad 0 \leq i < k, \tag{3.15}$$

for $1 \leq k \leq n$ by induction. Equations (3.9) and (3.14) give that the searching directions $d_0, d_1, \ldots, d_{n-1}$ are conjugate with respect to A, i.e.,

$$d_i^T A d_j = 0 \qquad 0 < i < j \leq n-1.$$

Then, by the definition of $g_k = V\phi_q(X_k) = AX_k + b$ and the fact that g_{k+1} is orthogonal to S_k and hence d_k in Algorithm 3.2, gives that

$$d_i^T g_n = 0 \qquad \text{for } 0 \leq i \leq n-1.$$

The linear independence of $d_0, d_1, \ldots, d_{n-1}$ in R^n forces $g_n = 0$. That X_n is a minimum follows the positive definiteness of A.

Next, if we consider $H_{k+1}Y_k$ by Equations (3.10), (3.11), and (3.12), then

$$H_{k+1}Y_k = H_k Y_k + \frac{S_k S_k^T}{S_k^T Y_k} Y_k - \frac{H_k Y_k Y_k^T H_k}{Y_k^T H_k Y_k} Y_k = S_k. \tag{3.16}$$

Equation (3.16) is what we called the quasi-Newton equation and hence the DFP method is a special case of the quasi-Newton method. Therefore, the DFP method possesses the properties which are commonly possessed by the quasi-Newton method.

Convergence

The DFP method has become one of the most popular and most successful techniques for finding the minimum of a differentiable function of

several variables since 1963. But until 1971, the convergence was not proved to be Q-superlinear.

Powell (51) has shown that, if

(1) the objective function ϕ is twice continuously differentiable,

(2) the objective function ϕ is strongly convex, i.e., the eigenvalues of the Hessian are bounded below by a positive constant,

(3) the line search performed at each iteration locates the exact minimum of the function along that line, i.e., λ_k is chosen so that

$$\phi(X_k - \lambda_k P_k) = \min\{\phi(X_k + \lambda P_k) : \lambda \geq 0\}$$

where $P_k = -H_k \nabla\phi(X_k)$, then the sequence of function values generated by the method converges to the minimum at a linear rate.

If, in addition,

(4) the second derivatives of the objective function ϕ satisfy a Lipschitz condition at the location of the minimum X^*, i.e.,

$$||\nabla^2\phi(X) - \nabla^2\phi(X^*)||_F \leq M||X - X^*||$$

for some constant $M > 0$ and for all $X \in S = \{X : \phi(X) \leq \phi(X_0)\}$,

then the rate of convergence is Q-superlinear.

The set S in hypothesis (4) is reasonable since the stability of the DFP method implies that the sequence $\{X_k\}$ generated by the method is contained in this set. We can conclude the above and make a theorem related to Algorithm 3.2 as follows.

Theorem 3.2: If Algorithm 3.2 is applied to an objective function which satisfies (1), (2), and (4), then the sequence $\{X_k\}$ generated by

this algorithm converges Q-superlinearly to X^*, the global minimum of ϕ, from X_0 and a positive definite matrix $H_0 = I$; i.e.,

$$\lim_{k\to\infty} ||X_{k+1} - X^*|| / ||X_k - X^*|| = 0.$$

Computing experiments for the DFP method were successful; Powell has used this method to solve a system of 100 nonlinear simultaneous equations successfully. Numerical difficulties also have been reported. These are mainly caused by the breakdown of the positive definiteness of H_k. Broyden (14) notes that negative steps have to be taken occasionally. This implies that some of the H_k's are not positive definite. Broyden attributes this failure to excessive rounding error. Bard (2) pointed out that poor scaling could cause H_k to become singular. Abbott (1) indicated that a more probable cause of loss of positive definiteness is failure to perform an exact line search.

In order to improve the performance, a variation on the original method was suggested by McCormick and Pearson (45). For an objective function of n variables the procedure is "restarted" every n (or n + 1) iterations by ignoring the usual matrix updating formula (Equation (3.10)) and instead setting the matrix H_k (or H_{k+1}) to be the identity matrix and taking the search direction to be the negative of the gradient.

McCormick (46) has shown that , given all the hypotheses (1) through (4), the rate of convergence of the "restarted" DFP method is quadratic when the decrease in error is measured over the interval of n iterations between restarting steps.

In practice, obtaining the exact solution in a linear search is costly (e.g., consuming most of the computation time), and since, in any case, the "exact" solution is never obtained, there have also been some

results on the convergence properties of the DFP method without exact line searches.

Broyden, Dennis and Moré (19) proved the following: Assume the objective function ϕ is twice differentiable in an open convex set D containing the minimum point X^*, and the Hessian matrix of ϕ at X^* is symmetric and positive definite and satisfies the one-sided Lipschitz condition

$$||\nabla^2\phi(X) - \nabla^2\phi(X^*)|| \leq M||X - X^*||^p$$

for some $p > 0$, $M > 0$, and all X in D. If (X_0, H_0) is sufficiently close to $(X^*, \nabla^2\phi(X^*))$, then the DFP method with $\lambda_k = 1$ for all k is Q-superlinearly convergent to X^*.

Dennis and Moré (30) showed that the sequence of step sizes $\{\lambda_k\}$ converges to one if and only if the method converges superlinearly.

Recently, convergence theory for the DFP method has been extended to cover the practical situation. Lenard (44) gave conditions on the error incurred in the line search performed at each iteration under which the order of convergence of the DFP method is linear or superlinear for the original method and n-step quadratic for the restarted method.

Even though the rate of convergence of the restarted version seems faster than that of the original proposal, we use the original algorithm in the next chapter to compare the performance with that of the Gay-Schnabel algorithm.

CHAPTER IV

QUALITY CONSIDERATIONS OF THE ALGORITHMS

The aim of this chapter is to compare the performance of the algorithms in the Gay-Schnabel proposal with that of other algorithms mentioned in the previous chapters--Broyden's method and the DFP method. Numerical tests were carried out on the IBM 370/158 computer at Oklahoma State University. The machine has a 64-bit word in double precision, with 56 bits devoted to the mantissa, giving an accuracy of 16 decimal digits.

Among all these algorithms, algorithms I and I' are Gay-Schnabel's original implementation. Brown's method and the DFP method are embodied in existing computer programs in IMSL (43), a fact which leads to some differences in comparison. Algorithms II, II', III, and III'--the extensions of the Gay-Schnabel algorithm--as well as Broyden's first method, are programmed by the author in standard FORTRAN.

In the implementation of algorithms II and III, the step size $S_k = -\lambda_k B_k^{-1} F_k$ is determined by choosing λ_k according to the nonlinear search described in Reference (13) with the added restriction that $||S_k|| \leq 1$. Instead of storing B_k, we actually store and update $H_k = B_k^{-1}$. The computation of $\hat{S}_k$ is carried out by the Gram-Schmidt orthonormalization process.

The quality of the algorithms will be assessed by the following criteria: convergence properties, computational complexity, and effort

required. All those algorithms have been analytically proved to be Q-superlinearly convergent except Brown's method, which converges Q-quadratically sometimes. These merits will be explained in the computer output if possible. But instead of testing the convergence properties directly, we will examine the accuracy of the solution obtained. The second criterion is often related to the time taken to reach the solution and the amount of computer storage space used in computation. Frequently, the time taken is in conflict with the accuracy of the solution obtained and the storage required, so some arbitrary assumptions will be made before a comparison of two algorithms.

Since time taken depends heavily on the strategy of program implementation and also the difficulty in comparing the times when two methods are tested on two different machines, we assume that the total amount of computation is directly proportional to the number of function evaluations to obtain the solution and that the constant of proportionality is the same for all methods. This is more true as the complexity of the functions increases and the constant of proportionality approaches unity. In the case of Brown's method "equivalent function evaluations" are quoted since in that method some elements of F are computed more often than others. The term "equivalent function evaluations" is defined as the total number of evaluations of the individual function components, which is simply the number of iterations multiplied by $(n^2 + 3n)/2$. If we divide this number, "equivalent function evaluations," by n and round it to an integer, then this integer can be used to approximate the number of function calls. Since the amount of labor involved in an iteration of different methods may vary tremendously, the number of iterations will

primarily be used in the comparison of the member of Gay-Schnabel's class.

In order to compare the performance of the various methods under as nearly identical conditions as possible, they are used to solve a set of test problems, starting from the same initial point. The accuracy of the final solution will be regarded as obtained when the Euclidean norm of the vector F becomes less than a tolerance arbitrarily chosen to be 10^{-10}, although there are some other criteria for the termination rule used in Brown's method and the DFP method as mentioned in Chapter III.

In order to keep the linear independence of the steps S_i generated by Gay-Schnabel's original algorithm, the criterion parameter τ used to judge the restart procedure will be set to 10. This is a good choice according to Reference (38).

We note that the Broyden algorithm chosen here to compare with other algorithms is Broyden's first method which is superior to Broyden's second method (13).

A table is given for each of the following selected problems and the computing results are reported for the following items: convergence, number of iterations, number of function evaluations, and accuracy (final norm of F). For reference purposes, the normalized number of function evaluations (Gay-Schnabel, 1977) and the mean convergence rate (Broyden, 1965) are also reported. The normalized number of function evaluations is defined by dividing the actual number by the minimum of the numbers for that problem and rounding to two decimal places while the mean convergence rate R is given by

$$R = (1/m)\ \ln(N_1/N_m) \tag{4.1}$$

where m is the total number of function evaluations and N_1, N_m the initial and final Euclidean norm of F. This R, as noted by Broyden, does not tend to an asymptotic value as $m \to \infty$ and is merely intended to be a concise index of the performance of a method applied to a particular problem in a particular situation.

Experimental Results

This section is devoted to a discussion of the behavior of the various methods when applied to a number of test cases. The table immediately following each problem indicates the computing results of each of the comparison criteria for the algorithms. The entries on the "convergence" item accompanied by a character "a" indicates that Broyden's (1965) quadratic interpolation technique failed to reduce $||F||$ and a character "b" indicates that the norm of F cannot reach the preassigned arbitrary number 10^{-10} owing to exceeding the limit (= 200) of iterations. The corresponding reports on this row are then the information at the time of failure. Algorithms III and III' are not tested until $N > 3$.

Notation used in each table:

Conv. - Is the algorithm convergent?

Eval. - Number of function evaluations.

N - Normalized function evaluations.

Iter. - Number of iterations.

T - Execution time in seconds.

$||F_0||$ - Initial norm of F.

$||F||$ - Final norm of F.

R - Mean convergence rate (see Equation (4.1)).

Problem 1

This problem is given by J. P. Chandler (21) to study the case when the algorithms are applied to a single nonlinear equation.

$$F(X) = \arctan X \qquad X \in R^1$$

with $X_0 = 3.0$, $X^* = 0.0$, and $||F_0|| = 1.249$. The results are given in Table I. We note that algorithms I, I', and II' have the same results. This is desirable since the orthogonal projection of S_k onto the orthogonal complement of S_{k-1} for $k \geq 1$ is a zero vector in R^1 and hence the updated H_k will not be changed after the first iteration for those algorithms.

TABLE I

COMPUTER RESULTS FOR PROBLEM 1

Method	Conv.	Iter.	Eval.	N	T (sec)	$\|\|F\|\|$	R
I	yes	6	9	1.00	0.15	0.6596D-11	2.885
I'	yes	6	9	1.00	0.14	0.6596D-11	2.885
II	yes	6	9	1.00	0.09	0.6596D-11	2.885
II'	yes	6	9	1.00	0.09	0.6596D-11	2.885
Broyden	yes	32	82	9.11	0.55	0.7833D-11	0.329
Brown	no	6	12	---	0.02	1.5710D 00	---

Problem 2

This problem is taken from the derivatives of Rosenbrock's function (54),

$$\phi(X) = 100\ (\xi_2 - \xi_1^2)^2 + (\xi_1 - 1)^2$$

given by

$$f_1(X) = 10\ (\xi_2 - \xi_1^2)$$

$$f_2(X) = 1 - \xi_1$$

with $X_0 = (-1.2,\ 1.0)^T$, $X^* = (1,\ 1)^T$, and $||F_0|| = 4.919$. The computer results are given in Table II.

TABLE II

COMPUTER RESULTS FOR PROBLEM 2

Method	Conv.	Iter.	Eval.	N	T (sec)	$\|\|F\|\|$	R
I	yes	5	15	1.88	0.16	0.755D-13	2.121
I'	no	14	61	---	0.46	1.875D 00	0.016
II	yes	29	78	9.75	0.60	0.138D-15	0.488
II'	(no)[a]	8	42	---	---	1.394D 00	0.300
Broyden	yes	25	71	8.88	0.53	0.666D-14	0.482
Brown	yes	3	8	1.00	0.03	0	∞

In the test of this problem, algorithm I' failed to converge due to an extremely small step size taken after the tenth iteration (step

size less than 0.5594D-06). This caused the new step S_i to lie almost entirely in the space spanned by the previous steps. Therefore, the linearly independent property was destroyed by the adding of a new step S_i to the previous steps. Sometimes this phenomenon can be avoided by adjusting the parameter EPS2 in the program (Appendix B) but a risk of underflow then occurs in the nonlinear search.

The author examined the output and listed the property of Q-superlinear convergence of Broyden's method, algorithms I and II in Tables III, IV, and V, respectively.

The trajectory of the search to the solution of Problem 2 by Broyden's method, algorithms I and II, and Brown's method are given in Figures 2, 3, 4, and 5, respectively. From these we can examine the status of convergence of these algorithms.

TABLE III

Q-SUPERLINEAR CONVERGENCE OF BROYDEN'S METHOD WHEN APPLIED TO PROBLEM 2

K	$\|\|X_k - X^*\|\|$
0	0.2200D-01
21	0.1228D-01
22	0.8090D 00
23	0.2107D 00
24	0.1089D-01
25	0.6661D-15

TABLE IV

Q-SUPERLINEAR CONVERGENCE OF ALGORITHM I WHEN APPLIED TO PROBLEM 2

K	$\|\|x_k - x^*\|\|$
0	0.2200D-01
1	0.2089D-01
2	0.2007D-01
3	0.1330D-01
4	0.8347D-02
5	0.7550D-14

TABLE V

Q-SUPERLINEAR CONVERGENCE OF ALGORITHM II WHEN APPLIED TO PROBLEM 2

K	$\|\|x_k - x^*\|\|$
0	0.2200D-01
16	0.1119D-01
17	0.8178D 00
26	0.1541D 00
27	0.7153D-02
28	0.1084D-02
29	0.1388D-16

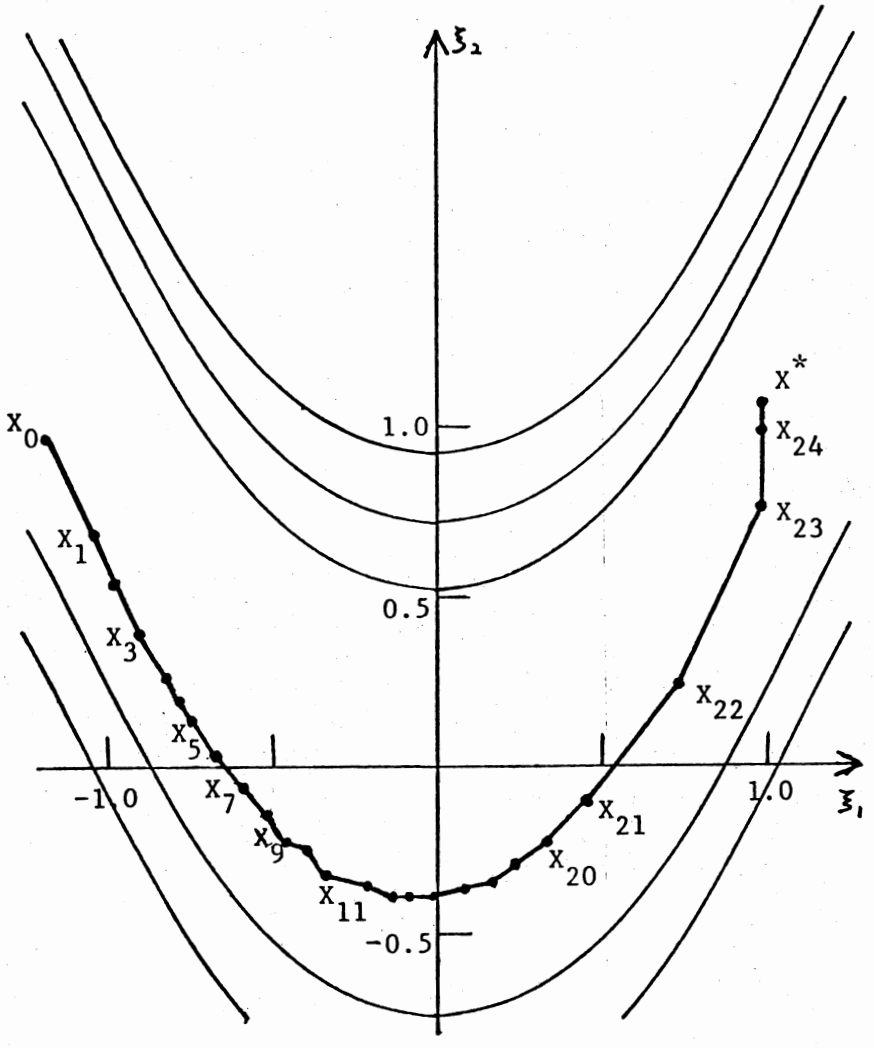

Figure 2. Trajectory of the Search to Solve Problem 2 by Broyden's Method

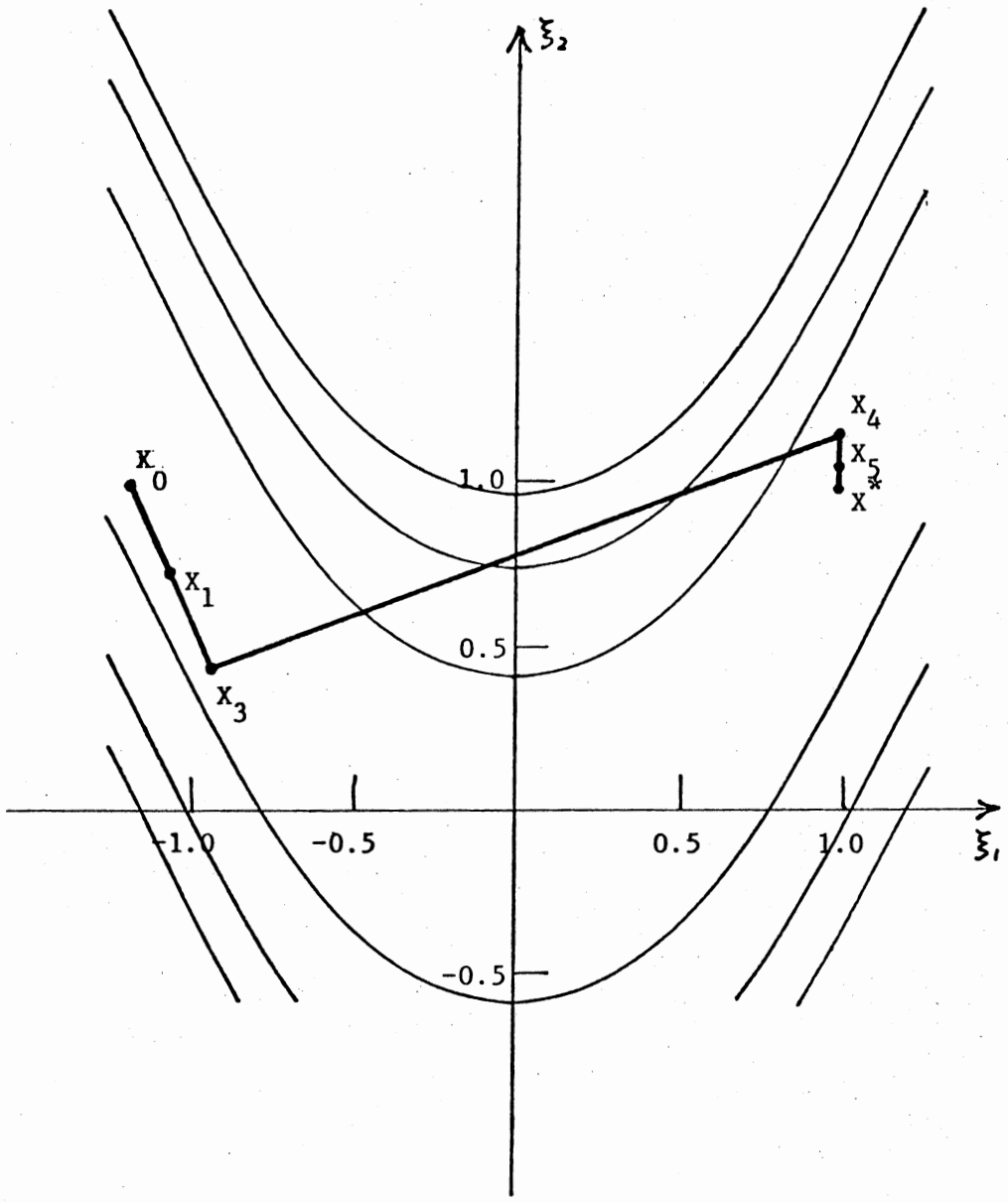

Figure 3. Trajectory of the Search to Solve Problem 2 by Algorithm I

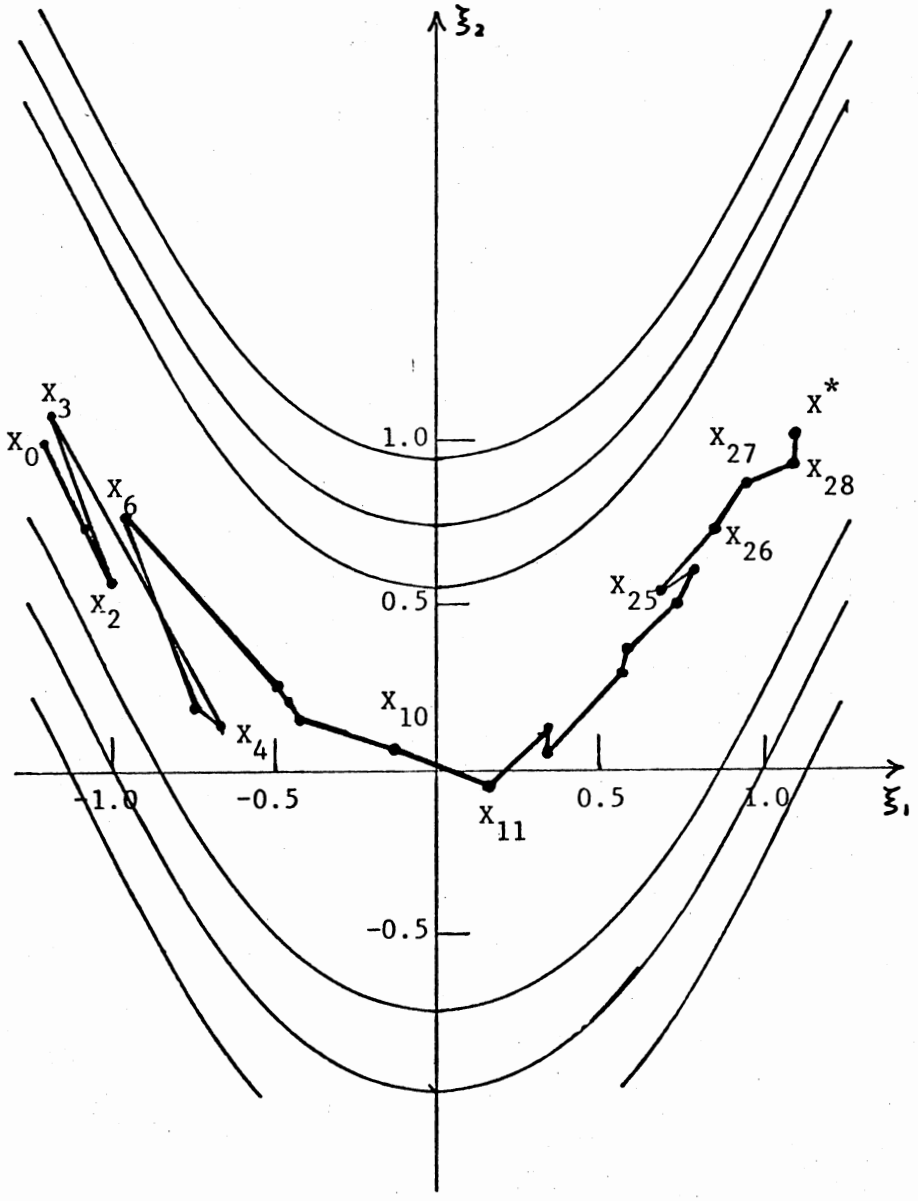

Figure 4. Trajectory of the Search to Solve Problem 2 by Algorithm II

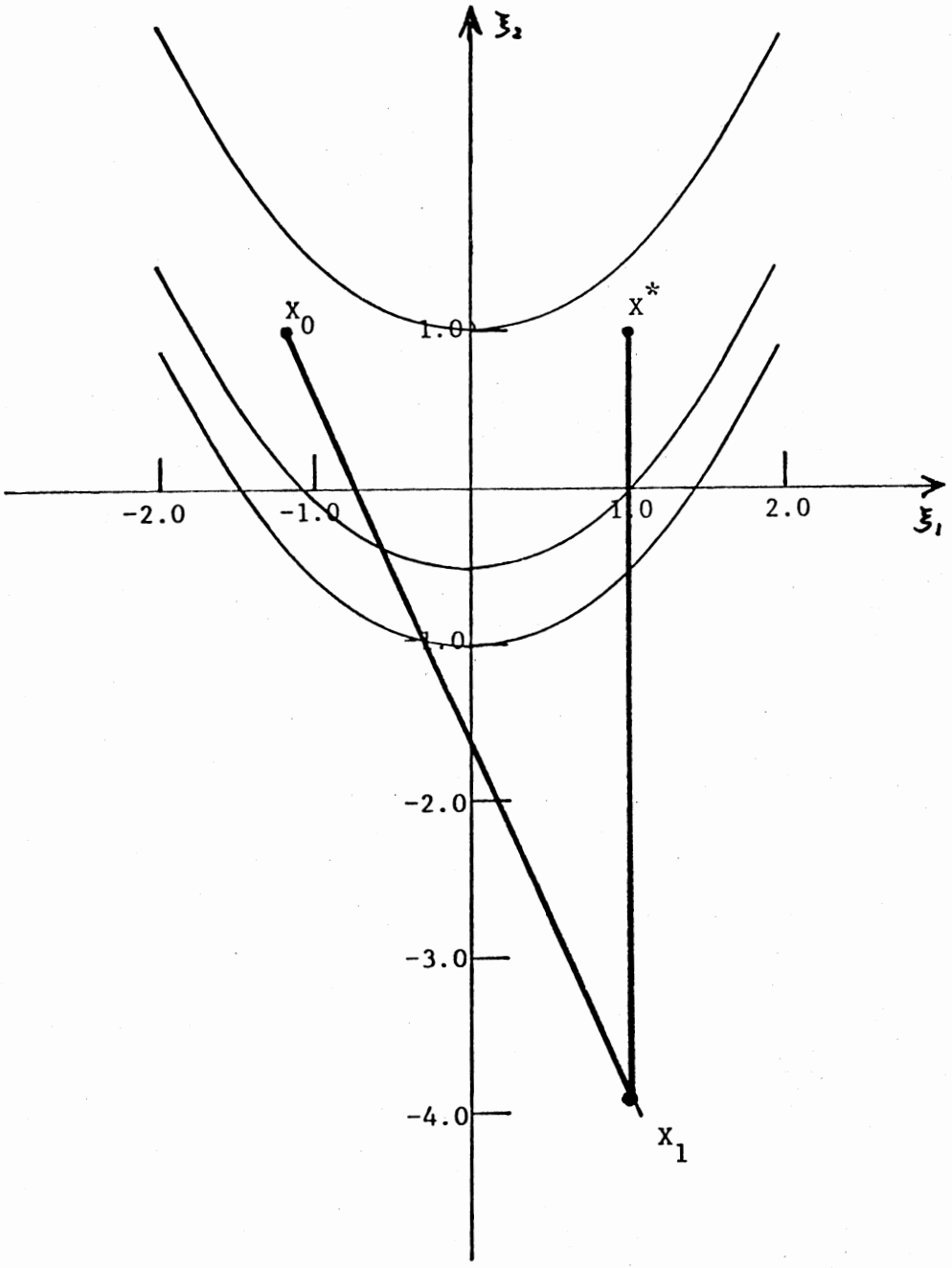

Figure 5. Trajectory of the Search to Solve Problem 2 by Brown's Method

Problem 3

The following system is given by Brown (7):

$$f_1(X) = \xi_1^2 - \xi_2 - 1$$

$$f_2(X) = (\xi_1 - 2)^2 + (\xi_2 - 0.5)^2 - 1$$

This system has two roots:

$$\gamma_1 \simeq (1.54634288,\ 1.39117631)^T;$$

and

$$\gamma \simeq (1.06734609,\ 0.13922767)^T.$$

The initial point is $(0.1,\ 2.0)^T$ with $||F_0|| = 5.706$. From the computer experiments, all the algorithms converged to γ_2, i.e., $X^* = \gamma_2$. The experimental results are given in Table VI.

TABLE VI

COMPUTER RESULTS FOR PROBLEM 3

Method	Conv.	Iter.	Eval.	N	T (sec)	$\|\|F\|\|$	R
I	yes	14	29	1.61	0.48	0.6240D-14	1.188
I'	yes	14	20	1.11	0.42	0.4163D-16	1.973
II	yes	13	20	1.11	0.29	0.6695D-14	1.719
II'	yes	14	20	1.11	0.31	0.5551D-16	1.959
Broyden	yes	14	18	1.00	0.23	0.7554D-14	1.903
Brown	yes	8	20	1.11	0.09	0.4923D-12	1.504

Problem 4

The following system was first studied by Freudenstein and Roth (35) and later by Broyden (13).

$$f_1(X) = -13 + \xi_1 + ((-\xi_2 + 5)\xi_2 - 2)\xi_2$$

$$f_2(X) = -29 + \xi_1 + ((\xi_2 + 1)\xi_2 - 14)\xi_2$$

with $X_0 = (15, -2)^T$, $X^* = (5, 4)^T$.

According to Broyden (13) and Brown (7), Broyden's method failed to converge to X^*. This is a reason why the author was interested in testing this problem.

The author tested all the algorithms with different initial points. Except for Brown's method, the members of Gay-Schnabel's class and Broyden's method failed to converge with the following initial points: $(15, -2)^T$, $(7.5, -1)^T$, $(3, 2)^T$. The failure occurred because the nonlinear search technique could not reduce the norm of F sufficiently. For instance, algorithm II, with $X_0 = (3, 2)^T$, reduced $||F||$ to 7.4635 in 11 iterations with 78 function evaluations, but in iteration 12, it made 100 more function evaluations (178 in total) and could not improve anything at all. Most of these function evaluations were spent in nonlinear search process. The information in Table VII is given with $X_0 = (3, 2.5)^T$ and $||F_0|| = 39.13$. The solution by Brown's method is exact. Algorithm II' seems superior to any other members in the Gay-Schnabel class.

TABLE VII

COMPUTER RESULTS FOR PROBLEM 4
$X_0 = (3, 2.5)^T, ||F_0|| = 39.13$

Method	Conv.	Iter.	Eval.	N	T (sec)	$\|\|F\|\|$	R
I	yes	9	17	1.06	0.27	0.4578D-14	2.158
I'	yes	9	17	1.06	0.25	0.2914D-13	2.049
II	yes	9	17	1.06	0.18	0.7127D-12	1.861
II'	yes	8	16	1.00	0.16	0.1862D-11	1.917
Broyden	yes	9	18	1.13	0.18	0.2739D-11	1.683
Brown	yes	10	25	1.56	0.11	0	∞

Problem 5

This transcendental equation was first studied by Brown and Conte (9).

$$f_1(X) = \frac{1}{2} \sin(\xi_1 \xi_2) - \xi_2/(4\pi) - \xi_1/2$$

$$f_2(X) = (1 - 1/(4\pi))(\exp(2\xi_1) - e) + e\xi_2/\pi - 2e\xi_1$$

with $X_0 = (0.6, 3)^T$, $X^* = (0.5, \pi)^T$, and $||F_0|| = 0.1236$.

The computer results are given in Table VIII. All the algorithms under consideration worked well on this problem. The member of the Gay-Schnabel class is superior to Broyden's method.

Problem 6

This problem is taken from Powell's paper (50).

$$f_1(X) = 10000\ \xi_1 \xi_2 - 1$$

$$f_2(X) = \exp(-\xi_1) + \exp(-\xi_2) - 1.0001.$$

TABLE VIII

COMPUTER RESULTS FOR PROBLEM 5

Method	Conv.	Iter.	Eval.	N	T (sec)	$\|\|F\|\|$	R
I	yes	8	11	1.10	0.24	0.1348D-13	2.713
I'	yes	8	11	1.10	0.25	0.1310D-15	3.135
II	yes	7	10	1.00	0.15	0.6228D-12	2.601
II'	yes	7	10	1.00	0.13	0.9658D-12	2.558
Broyden	yes	9	12	1.20	0.16	0.1046D-11	2.125
Brown	yes	10	25	2.50	0.11	0.1265D-13	1.196

Starting at the estimate $X_0 = (0, 1)^T$ with $||F_0|| = 1.065$. The solution is given approximately by $X^* \sim (0.1099 \times 10^{-4}, 9.096)^T$.

All the algorithms of Gay-Schnabel's class and Broyden's method failed to reduce the norm of F to be less than 10^{-3}. Only algorithm II can reach a point $(0.1388D\text{-}04, 7.204)^T$ with $||F|| = 0.6933D\text{-}03$ by 167 function evaluations. If we use $(0.1, 1.0)^T$ as a starting guess, then all the algorithms, except for Broyden's method, can reduce $||F|| < 10^{-4}$ and algorithm II' converged in 57 function evaluations. In both initiative cases, Brown's method converges in 33 function calls (65 individual function component evaluations). We omit the reports for this problem.

Problem 7

We start to test algorithms III and III' with this problem which comes from Brown and Gearhart (11).

$$f_1(X) = \xi_1^2 + 2\xi_1^2 - 4$$

$$f_2(X) = \xi_1^2 + \xi_2^2 + \xi_3 - 8$$

$$f_3(X) = (\xi_1 - 1)^2 + (2\xi_2 - \sqrt{2})^2 + (\xi_3 - 5)^2 - 4$$

with initial point $X_0 = (1, 0.7, 5)^T$, $||F_0|| = 4.729$ and $X^* = (0, \sqrt{2}, 6)^T$.

The author tested this problem and discovered that all the algorithms were convergent. Brown's method converged to $X^* = (0, \sqrt{2}, 6)^T$, while Broyden's method and the member of Gay-Schnabel's class converged to another root at $(2, 0, 4)^T$. This is different from the Gay-Schnabel report. Gay and Schnabel claimed that their original algorithm also converged to $(0, \sqrt{2}, 6)^T$.

The author tested this problem with another initial point $(1, 1, 5)^T$ which is closer to $(0, \sqrt{2}, 6)^T$ rather than to $(2, 0, 4)^T$. This time, all algorithms under consideration converged to $(0, \sqrt{2}, 6)^T$ as desired.

Please note that the initial point $(1, 0.7, 5)^T$ is almost equidistant from $(0, \sqrt{2}, 6)^T$ and $(2, 0, 4)^T$ but is closer to $(2, 0, 4)^T$. The computer results with $(1, 0.7, 5)^T$ as initial point are given in Table IX. Algorithm III' seems to be the most efficient one.

TABLE IX

COMPUTER RESULTS FOR PROBLEM 7

Method	Conv.	Iter.	Eval.	N	T (sec)	$\|\|F\|\|$	R
I	yes	12	17	1.00	0.49	0.1817D-12	1.817
I'	yes	12	17	1.00	0.46	0.1087D-12	1.847
II	yes	11	22	1.30	0.29	0.4537D-13	1.467
II'	yes	11	17	1.00	0.25	0.3093D-13	1.920
III (t=2)	yes	10	24	1.41	0.32	0.2941D-12	1.267
III' (t=2)	yes	10	17	1.00	0.27	0.6324D-14	2.015
Broyden	no	11	17	1.00	0.32	0.1083D-10	1.577
Brown	yes	12	36	2.11	0.24	0.1088D-14	1.000

Problem 8

The following system was first studied by Brown (7). This is a system in which every equation is linear except for the very last one. For n = 5, 10:

$$f_i(X) = -(n + 1) + 2\xi_i + \sum_{\substack{j=1 \\ j\neq 1}}^{n} \xi_j \qquad i = 1, \ldots, n - 1,$$

$$f_n(X) = -1 + \prod_{j=1}^{n} \xi_j$$

$$X_0 = (0.5, \ldots, 0.5)^T; \; X^* = (1, \ldots, 1)^T.$$

The author also used this problem to test algorithm III (and algorithm III') with t = 2, 3, 4, i.e., set $\hat{S}_k$ (or $\hat{Y}_k$) equal to the projection of S_k (or Y_k) orthogonal to the previous t S_j's (or Y_j's), t < n in the extension of Gay-Schnabel's algorithm.

For n = 5, the author tested all the algorithms with three different initial points: $r = (0.5, \ldots, 0.5)^T$, $s = (0.75, \ldots, 0.75)^T$, and $w = (1.5, \ldots, 1.5)^T$. The computer results are given in Tables X, XI, and XII, respectively. We note that Brown's method converged in each case to the root $X^* = (1, 1, 1, 1, 1)^T$. Broyden's method also converged to X^* (but slower) with given initial point r or s, but converged to the root given approximately by (-0.579, -0.579, -0.579, -0.579, 8.90) with initial point W. Taking r as initial point, the members of the Gay-Schnabel class failed to converge except for algorithm III with t = 2 and t = 4. Both of these two cases are not better than Broyden's method. As we take S or W as an initial point, the Gay-Schnabel algorithm and its extensions seem superior to Broyden's method.

For n = 10, the author also tested all the algorithms with three different initial points, the points all of whose components are 0.5, 0.75, and 1.5, respectively. The computer results are given in Tables XIII, XIV, and XV. We note that Brown's method converged exactly to $(1, \ldots, 1)^T$ and reduced $||F|| = 0$ for the cases of $X_0 = (0.5, \ldots, 0.5)^T$ and $(0.75, \ldots, 0.75)^T$. The Brown method also worked well for

$X_0 = (1.5, \ldots, 1.5)^T$, i.e., converged in 10 iterations. Broyden's method never converged in each case. As to the member of Gay-Schnabel's class, the case t = 2 in algorithm III, with $X_0 = (0.5, \ldots, 0.5)^T$, converged in 25 iterations with 111 function calls. Algorithm II', with $X_0 = (0.75, \ldots, 0.75)^T$, converged in 33 iterations with 173 function calls. Taking $(1.5, \ldots, 1.5)^T$ as an initial point, all members in algorithm III converged. These are the only convergent cases.

TABLE X

COMPUTER RESULTS FOR PROBLEM 8
$N = 5$, $X_0 = (0.5, \ldots, 0.5)^T$,
$||F_0|| = 6.078$

Method	Conv.	Iter.	Eval.	N	Root	$\|\|F\|\|$	R
I	(no)[a]	7	21	---	---	5.5830D 00	0.004
I'	(no)[a]	5	17	---	---	5.5541D 00	0.005
II	(no)[a]	4	14	---	---	5.5700D 00	0.006
II'	(no)[a]	4	14	---	---	5.6430D 00	0.005
III t=2	yes	17	35	1.25	r_1	0.7269D-13	0.916
III t=3	(no)[a]	13	32	---	---	4.3510D 00	0.010
III t=4	yes	18	36	1.28	r_1	0.2791D-12	0.853
III' t=2	(no)[a]	4	14	---	---	5.6640D 00	0.005
III' t=3	(no)[a]	5	82	---	---	5.6640D 00	0.001
III' t=4	(no)[a]	5	82	---	---	5.6640D 00	0.001
Broyden	yes	16	30	1.07	r_2	0.1388D-16	1.354
Brown	yes	7	28	1.00	r_2	0	∞

$r_1 = (-0.579, -0.579, -0.579, -0.579, 9.895)^T$.

$r_2 = (1, 1, 1, 1, 1)^T$.

TABLE XI

COMPUTER RESULTS FOR PROBLEM 8
$N = 5$, $X_0 = (0.75, \ldots, 0.75)^T$,
$||F_0|| = 3.095$

Method	Conv.	Iter.	Eval.	N	Root	$\|\|F\|\|$	R
I	yes	11	21	1.00	r_1	0.1625D-13	1.566
I'	yes	11	21	1.00	r_1	0.1489D-13	1.570
II	yes	11	32	1.52	r_2	0.2054D-13	1.020
II'	yes	11	21	1.00	r_1	0.4720D-13	1.515
t=2	yes	11	32	1.52	r_2	0.1528D-13	1.029
III t=3	yes	11	32	1.52	r_2	0.7216D-15	1.125
t=4	yes	11	32	1.52	r_2	0.9714D-16	1.188
t=2	yes	12	23	1.09	r_1	0.5395D-13	1.177
III' t=3	yes	12	21	1.00	r_1	0.2916D-13	1.538
t=4	yes	11	21	1.00	r_1	0.1476D-13	1.570
Broyden	yes	11	32	1.52	r_2	0.3189D-15	1.150
Brown	yes	9	36	1.71	r_2	0	∞

$r_1 = (-0.579, -0.579, -0.579, -0.579, 8.895)^T$.

$r_2 = (1, 1, 1, 1, 1)^T$.

TABLE XII

COMPUTER RESULTS FOR PROBLEM 8
$N = 5$, $X_0 = (1.5, \ldots, 1.5)^T$,
$||F_0|| = 8.915$

Method	Conv.	Iter.	Eval.	N	Root	$\|\|F\|\|$	R
I	yes	9	17	1.06	r_1	0.2542D-13	1.970
I'	yes	9	17	1.06	r_1	0.2251D-14	2.113
II	yes	9	16	1.00	r_1	0.4965D-15	2.339
II'	yes	11	38	2.37	r_1	0.8882D-15	0.970
t=2	yes	9	16	1.00	r_1	0.5894D-13	2.041
III t=3	yes	9	16	1.00	r_1	0.3520D-13	2.073
t=4	yes	9	16	1.00	r_1	0.1955D-13	2.110
t=2	no	16	87	5.43	r_1	0.1241D-08	0.261
III' t=3	yes	10	18	1.11	r_1	0.4793D-15	2.081
t=4	yes	9	17	1.06	r_1	0.1940D-11	1.715
Broyden	yes	9	16	1.00	r_1	0.2449D-11	1.808
Brown	yes	7	28	1.75	r_2	0.2498D-11	1.032

$r_1 = (-0.579, -0.579, -0.579, -0.579, 8.895)^T$.

$r_2 = (1, 1, 1, 1, 1)^T$.

TABLE XIII

COMPUTER RESULTS FOR PROBLEM 8
$N = 10$, $X_0 = (0.5, \ldots, 0.5)^T$,
$||F_0|| = 16.53$

Method	Conv.	Iter.	Eval.	N	$\|\|F\|\|$	R
I	(no)[a]	1	12	---	16.5000D 00	0.000
I'	(no)[a]	1	12	---	16.5000D 00	0.000
II	(no)[a]	5	20	---	16.0800D 00	0.001
II'	(no)[a]	1	12	---	16.5000D 00	0.000
t=2	yes	25	111	1.71	0.1231D-11	0.272
III t=3	(no)[a]	4	17	---	16.1000D 00	0.002
t=4	(no)[a]	4	17	---	16.1000D 00	0.002
t=2	(no)[a]	1	12	---	16.5000D 00	0.002
III' t=3	(no)[a]	1	12	---	16.5000D 00	0.002
t=4	(no)[a]	1	12	---	16.5000D 00	0.002
Broyden	(no)[a]	9	23	---	12.0800D 00	0.014
Brown	yes	10	65	1.00	0	∞

TABLE XIV

COMPUTER RESULTS FOR PROBLEM 8
$N = 10$, $X_0 = (0.75, \ldots, 0.75)^T$,
$||F_0|| = 8.304$

Method	Conv.	Iter.	Eval.	N	$\|\|F\|\|$	R
I	(no)[a]	9	27	---	7.2510D 00	0.005
I'	(no)[a]	5	21	---	7.4310D 00	0.005
II	(no)[a]	5	20	---	6.7760D 00	0.010
II'	no*	33	173	2.66	0.3494D-09	0.138
III t=2	(no)[a]	12	29	---	1.0000D 00	0.073
III t=3	(no)[a]	6	71	---	6.7230D 00	0.003
III t=4	(no)[a]	5	22	---	6.7230D 00	0.009
III' t=2	(no)[a]	8	116	---	7.4200D 00	0.001
III' t=3	(no)[a]	4	16	---	7.4310D 00	0.007
III' t=4	(no)[a]	4	16	---	7.4310D 00	0.007
Broyden	(no)[a]	5	17	---	0.4913D 00	0.166
Brown	yes	10	65	1.00	0	∞

*It failed to reduce $||F|| < 10^{-10}$.

TABLE XV

COMPUTER RESULTS FOR PROBLEM 8
$N = 10$, $X_0 = (1.5, \ldots, 1.5)^T$,
$||F_0|| = 59.02$

Method	Conv.	Iter.	Eval.	N	$\|\|F\|\|$	R
I	(no)[a]	2	13	---	8.9930D 00	0.145
I'	(no)[a]	2	13	---	8.9650D 00	0.145
II	(no)[a]	3	17	---	11.6500D 00	0.095
II'	(no)[a]	2	13	---	8.9650D 00	0.145
t=2	yes	19	40	1.00	0.4246D-13	0.872
III t=3	yes	19	40	1.00	0.6876D-13	0.860
t=4	yes	21	44	1.10	0.4434D-11	0.687
t=2	(no)[a]	2	13	---	8.9650D 00	0.145
III' t=3	(no)[a]	2	13	---	8.9650D 00	0.145
t=4	(no)[a]	2	13	---	8.9650D 00	0.145
Broyden	(no)[a]	5	23	---	11.6300D 00	0.071
Brown	yes	10	65	1.625	0.9873D-14	0.559

Problem 9

This specially-constructed system was first studied by Broyden (13) and then by Gay and Schnabel (38).

$$f_1(X) = -(3 + \alpha\xi_1)\xi_1 + 2\xi_2 - \beta,$$

$$f_1(X) = \xi_{i-1} - (3 + \alpha\xi_i)\xi_i + 2\xi_{i+1} - \beta, \quad i = 2, 3, \ldots, n-1,$$

$$f_n(X) = \xi_{n-1} - (3 + \alpha\xi_n)\xi_n - \beta.$$

Values of α, β, and n are given as follows:

(1) $\alpha = -0.1$ $\beta = 1.0$ $n = 5$

(2) $\alpha = -0.5$ $\beta = 1.0$ $n = 5$

(3) $\alpha = -0.5$ $\beta = 1.0$ $n = 10$

and the initial estimates in all cases were

$$X_0 = (-1.0, -1.0, \ldots, -1.0)^T.$$

From the computing experiments of the algorithms with this problem, the author discovered that every algorithm considered in this paper converged in each case and that all versions and extensions of the Gay-Schnabel method are superior to the Broyden method. The computer reports are given Tables XVI, XVII, and XVIII.

Problem 10

This system of transcendental equations from Deist and Sefor (26) is defined by

$$f_i(X) = \sum_{\substack{j=1 \\ j \neq i}}^{6} \cot\beta_i \xi_j \qquad i < i < 6$$

where $(\beta_1, \ldots, \beta_6) = 10^{-2}$ (2.249, 2.166, 2.083, 2.000, 1.918, 1.835).

TABLE XVI

COMPUTER RESULTS FOR PROBLEM 9
$N = 5$, $\alpha = -0.1$, $\beta = 1.0$,
$||F_0|| = 1.910$

Method	Conv.	Iter.	Eval.	N	$\|\|F\|\|$	R
I	yes	8	14	1.08	0.4754D-13	2.241
I'	yes	8	14	1.08	0.4264D-13	2.245
II	yes	8	14	1.08	0.1158D-13	2.338
II'	yes	8	14	1.08	0.4119D-11	2.248
t=2	yes	7	13	1.00	0.2631D-11	2.138
III t=3	yes	7	13	1.00	0.2703D-12	2.263
t=4	yes	7	13	1.00	0.5954D-12	2.215
t=2	yes	7	13	1.00	0.3305D-11	2.082
III' t=3	yes	7	13	1.00	0.4789D-12	2.232
t=4	yes	7	13	1.00	0.4368D-11	2.062
Broyden	yes	7	13	1.00	0.5521D-11	2.044
Brown	yes	5	20	1.54	0.4440D-15	1.800

TABLE XVII

COMPUTER RESULTS FOR PROBLEM 9
N = 5, α = -0.5, β = 1.0,
$||F_0||$ = 1.803

Method	Conv.	Iter.	Eval.	N	$\|\|F\|\|$	R
I	yes	8	14	1.08	0.6248D-13	2.214
I'	yes	8	14	1.08	0.6256D-13	2.214
II	yes	7	13	1.00	0.8435D-11	2.007
II'	yes	8	14	1.08	0.3550D-14	2.419
t=2	yes	8	14	1.08	0.7573D-14	2.365
III t=3	yes	7	13	1.00	0.3434D-12	2.253
t=4	yes	7	13	1.00	0.6075D-12	2.209
t=2	yes	8	14	1.08	0.1867D-13	2.300
III' t=3	yes	7	13	1.00	0.1510D-11	2.139
t=4	yes	7	13	1.00	0.1895D-11	2.122
Broyden	yes	8	14	1.08	0.4842D-12	2.073
Brown	yes	5	20	1.54	0.5439D-15	1.787

TABLE XVIII

COMPUTER RESULTS FOR PROBLEM 9
$N = 10$, $\alpha = -0.5$, $\beta = 1.0$,
$||F_0|| = 2.121$

Method	Conv.	Iter.	Eval.	N	$\|\|F\|\|$	R
I	yes	10	21	1.05	0.1028D-11	1.350
I'	yes	10	21	1.05	0.1012D-11	1.351
II	yes	10	21	1.05	0.1473D-11	1.333
II'	yes	10	21	1.05	0.1065D-11	1.349
t=2	yes	9	20	1.00	0.1067D-10	1.300
III t=3	yes	9	20	1.00	0.1619D-10	1.280
t=4	yes	9	20	1.00	0.1428D-9	1.171
t=2	yes	10	21	1.05	0.1295D-11	1.339
III' t=3	yes	9	20	1.00	0.3925D-10	1.236
t=4	yes	10	21	1.05	0.8649D-12	1.358
Broyden	yes	10	21	1.05	0.1785D-10	1.214
Brown	yes	5	20	1.00	0.3966D-12	1.465

$$X_0 = (75, 75, \ldots, 75)^T, \qquad ||F_0|| = 1.397$$

$$X^* = (121.850, 114.161, 93.6488, 62.3186, 41.3219, 30.5027)^T.$$

The author tested this problem and discovered that all the algorithms under consideration could reduce the norm of F to be smaller than 1.0D-08 at least. The original Gay-Schnabel methods--algorithms I and I' --seem to be more suitable in solving this problem. It seems that each member of the Gay-Schnabel class worked better than Broyden's method. The computer reports are given in Table XIX.

From the previous experiments, the author learned that Brown's method could handle nine problems out of the given ten and converged rapidly and accurately 21 times while the 22 different test cases were given. Brown's method often reduced the norm of F to be less than 10^{-15} or even exactly to be 0. These merits are not reduced by the number of function calls. Brown's method has played a good role in the library subroutine to solve a nonlinear system of equations. Since our objective is to study the performance of the Gay-Schnabel method, the following three problems will be devoted to the comparison of the performance of the members of Gay-Schnabel's class with that of the DFP method when applied to finding a minimum of an unconstrained minimization problem.

Problem 11

This minimization problem is introduced by Powell (49).

$$\phi(x) = (\xi_1 + 10\xi_2)^2 + 5(\xi_3 - \xi_4)^2 + (\xi_2 - 2\xi_3)^4 + 10(\xi_1 - \xi_4)^4.$$

$(3, -1, 0, 1)^T$ is used as an initial point. The minimum occur at $(0, 0, 0, 0)^T$. This function is a severe test since the Hessian matrix of ϕ is singular at the minimum point.

TABLE XIX

COMPUTER RESULTS FOR PROBLEM 10
$N = 6$, $X_0 = (75, \ldots, 75)^T$,
$||F_0|| = 1.397$

Method	Conv.	Iter.	Eval.	N	$\|\|F\|\|$	R
I	yes	17	26	1.08	0.2589D-11	1.039
I'	yes	17	24	1.00	0.7793D-12	1.176
II	no*	18	29	1.21	0.1090D-9	0.803
II'	no*	19	30	1.25	0.1469D-10	0.843
t=2	yes	22	44	1.83	0.7446D-11	0.590
III t=3	yes	21	39	1.63	0.4620D-11	0.679
t=4	yes	23	52	2.17	0.2931D-11	0.517
t=2	no*	24	51	2.13	0.8346D-10	0.462
III' t=3	yes	25	51	2.13	0.2915D-13	0.618
t=4	yes	25	45	1.88	0.2217D-11	0.604
Broyden	no*	27	61	2.54	0.5984D-09	0.354
Brown	yes	15	68	2.83	0.1115D-12	0.444

*It failed to reduce $||F|| < 1.0D-10$.

The test was carried out with two different initial points $(3, -1, 0, 1)^T$ and $(3, 1, 0, 1)^T$. Broyden's method worked well but algorithms II', III, and III' seem to be more efficient. The DFP method is inferior to any other method in both cases.

Examining Tables XX and XXI, it can be realized that algorithm III with t = 2 is a superior algorithm in Table XX, but it becomes inferior to any other members of the Gay-Schnabel class in Table XXI. This reveals that the efficiency of an algorithm will be affected by the given initial point.

Problem 12

This function, credited to C. F. Wood and documented by Pearson (48), is given by

$$\phi(X) = 100\ (\xi_2 - \xi_1^2)^2 + (1 - \xi_1)^2 + 90\ (\xi_4 - \xi_3^2)^2 + (1 - \xi_3)^2$$

$$+ 10.1\ \{(\xi_2 - 1)^2 + (\xi_4 - 1)\}^2 + 19.8\ (\xi_2 - 1)(\xi_4 - 1)$$

with $X_0 = (-3, -1, -3, -1)^T$ and $X^* = (1, 1, 1, 1)^T$. The minimum of ϕ at X^* is zero.

Starting at $X_0 = (-3, -1, -3, -1)^T$, all the algorithms under consideration failed to converge. Some of those algorithms such as algorithm II', algorithm III with case t = 2, 3, and algorithm III' with case t = 2, 3 converged to a stationary point at $(-0.9679, 0.9471, -0.9695, 0.9512)^T$ successfully, but this was a nonoptimal stationary point. The function value at this nonoptimal point is 49.9218. Algorithms I, I', II and Broyden's method failed to locate any stationary point. The DFP method reduced the function value form 19192 to 1.6471 in 11 iterations with 58 function evaluations, but it diverged afterward.

TABLE XX

COMPUTER RESULTS FOR PROBLEM 11
$X_0 = (3, -1, 0, 1)$, $\phi_0 = 215$

Method	Conv.	Iter.	Eval.	N	ϕ	R
I	yes	31	36	1.71	0.8619D-13	0.985
I'	yes	31	36	1.71	0.8688D-13	1.013
II	yes	25	40	1.90	0.7457D-13	0.890
II'	yes	16	21	1.00	0.2452D-13	1.748
III t=2	yes	17	24	1.14	0.1876D-13	1.541
III t=3	yes	19	24	1.14	0.7712D-13	1.482
III' t=2	yes	20	25	1.19	0.1404D-12	1.399
III' t=3	yes	17	22	1.05	0.1575D-12	1.584
Broyden	yes	30	35	1.67	0.1070D-12	1.007
DFP	yes	31	162	7.71	0.2298D-13	0.227

TABLE XXI

COMPUTER RESULTS FOR PROBLEM 11
$X_0 = (3, 1, 0, -1)$, $\phi_0 = 1320$

Method	Conv.	Iter.	Eval.	N	ϕ	R
I	yes	33	38	1.46	0.1351D-12	0.988
I'	yes	33	38	1.46	0.1351D-12	0.988
II	yes	21	26	1.00	0.4684D-14	1.573
II'	yes	19	26	1.00	0.1796D-12	1.433
III t=2	yes	28	61	2.35	0.1716D-12	0.612
III t=3	yes	21	26	1.00	0.1591D-13	1.526
III' t=2	yes	24	31	1.92	0.1197D-12	1.215
III' t=3	yes	23	30	1.53	0.3356D-13	1.298
Broyden	yes	33	38	1.46	0.1269D-12	0.990
DFP	yes	24	164	6.30	0.6049D-14	0.248

The author also tried $X_0 = (3, 1, 3, 1)$ as a starting guess and found that four of the given ten algorithms converged. This means that the four algorithms located $(1, 1, 1, 1)^T$ as a stationary point. The DFP method happened to be the same divergent situation after four time iterations. The computer results are given in Table XXII.

TABLE XXII

COMPUTER RESULTS OF PROBLEM 12
$X_0 = (3, 1, 3, 1), \phi_0 = 12168$

Method		Conv.	Iter.	Eval.	N	ϕ	R
I		(no)[b]	20	49	---	0.3910D-01	---
I'		(no)[b]	26	114	---	0.3916D-02	---
II		(no)[a]	22	118	---	-0.5192D-01	---
II'		yes	50	132	1.00	-0.7047D-10	0.248
III	t=2	yes	88	332	2.52	-0.2823D-23	0.192
	t=3	(no)[a]	24	108	---	-0.5632D-02	---
III'	t=2	yes	60	173	1.31	-0.9855D-23	0.361
	t=3	yes	48	141	1.07	-0.3364D-20	0.450
Broyden		(no)[b]	58	253	---	0.6207D-01	---
DFP		no	96	905	---	-0.2506D-10	---

Problem 13

This problem given by Fletcher and Powell (34) is defined by

$$\phi(X) = 100 \{(\xi_3 - 10\,\theta(\xi_1, \xi_2))^2 + (r(\xi_1, \xi_2) - 1)^2\} + \xi_3^2,$$

where

$$2\pi\theta\ (\xi_1,\ \xi_2) = \begin{cases} \arctan\ (\xi_2/\xi_1), & \xi_1 > 0 \\ \pi + \arctan\ (\xi_2/\xi_1), & \xi_1 < 0 \end{cases}$$

and

$$r\ (\xi_1,\ \xi_2) = (\xi_1^2 + \xi_2^2)^{1/2}.$$

This function has a steep-sided helical valley in the ξ_3 direction with pitch 10 and radius 1. The starting point $X_0 = (-1,\ 0,\ 0)^T$ and a minimum occur at (1, 0, 0).

In 18 iterations the DFP method reduced ϕ from 2500 to 0.2396D-23. Only two members, algorithms I' and III', of the Gay-Schnabel class converged. Broyden's method failed to converge. The results are given in Table XXIII.

TABLE XXIII

COMPUTER RESULTS OF PROBLEM 13
$X_0 = (-1,\ 0,\ 0)$, $\phi_0 = 2500$

Method	Conv.	Iter.	Eval.	N	ϕ	R
I	no	9	91	---	0.5862D-01	---
I'	yes	20	55	2.04	0.9486D-20	0.981
II	(no)[b]	140	1766	---	0.1374D+02	---
II'	no	6	64	---	0.1422D+02	---
III (t=2)	no	95	1615	---	0.1851D 00	---
III'(t=2)	yes	15	27	1.00	0.1271D-20	2.072
Broyden	no	9	91	---	0.5864D-01	---
DFP	yes	18	45	1.67	0.2396D-23	1.382

The author also used $(1, 0, -1)^T$, $(1, 0, 1)^T$, $(1, 0, 0.5)^T$, $(1, 0, -0.3)^T$, and $(2, 3, 4)^T$ as starting guesses. None of the algorithms, except for the DFP method, converged. We omit the reports for these tests.

The Rosenbrock function, Problem 2, is also a good unconstrained minimization problem. This function is difficult to minimize on account of its having a steep-sided valley following the curve $\xi_1^2 = \xi_2$. The optimal point is at $(1, 1)^T$ with function value zero. If we start at $(-1.2, 1)^T$, the searching paths of several methods have been shown in Figures 2 through 5. It revealed in Table II that algorithm I is the most efficient method for finding the minimum of the Rosenbrock function among all the members of Gay-Schnabel's class and Broyden's method. It would be interesting to compare the performance of this method with that of the DFP method when applied to the Rosenbrock function. Table XXIV gives this report. $\phi_0 = 24.2$.

TABLE XXIV

COMPARISON OF THE DFP METHOD AND ALGORITHM I WHEN APPLIED TO THE ROSENBROCK FUNCTION

Method	Conv.	Iter.	Eval.	N	T (sec)	ϕ	R
I	yes	5	15	1.00	0.17	0.5700D-23	3.78
DFP	yes	28	129	8.60	0.58	0.9478D-16	0.31

Conclusions

From the numerical examples and discussion in the preceding section, it is clear that Brown's method is probably as satisfactory a method as any member of the Gay-Schnabel class or the Broyden method for solving systems of nonlinear equations in R^n. Actually, Brown's method is considerably superior to any other methods previously available, especially for more difficult problems. Within the 22 different test cases which covered the first ten problems in the preceding section, Brown's method failed only once and successfully reduced the norm of F to be zero six times at least. The effectiveness of other algorithms can also be judged by the number of success in the tests available. This is given in Table XXV.

From the evidence presented in the tables, the author also noticed that most of the members of the Gay-Schnabel class failed as Broyden's method did, but when this converged, the former turned out to be more efficient. Problem 8 was the only exceptional case. When $n = 5$ with $X_0 = (0.5, \ldots, 0.5)^T$, Broyden's method converged but most of the Gay-Schnabel member did not converge; however, if it did converge, such as $t = 2$, $t = 4$ in algorithm III (Table X), it converged slowly. In some difficult problems, such as Problems 4, 6, or 8, a reasonably good initial estimate of the solution will improve the performance of the algorithms. A failing algorithm will become successful or even efficient with a good starting point. It seems reasonable to judge the efficiency by the mean and standard deviation of the normalized function evaluations. This information is also given in Table XXV.

By definition (Equation (4.1)), the mean convergence rate can also be used to judge both the effectiveness and the efficiency. This is

TABLE XXV

A COMPARISON OF THE ALGORITHMS BASED ON THREE DIFFERENT FACTORS

Method		Number of Success*	Normalized Func. Eval.† Mean	Normalized Func. Eval.† S.D.	Mean Conv. Rate† Mean	Mean Conv. Rate† S.D.
I		13	1.167	0.278	1.462	0.981
I'		12	1.049	0.043	1.402	1.071
II		13	1.840	2.496	1.273	0.941
II'		14	1.305	0.573	1.292	1.015
III	t=2	(11)**	1.280	0.319	1.169	0.753
	t=3	(7)**	1.164	0.282	1.055	0.911
	t=4	(8)**	1.259	0.413	1.096	0.848
III'	t=2	(7)**	1.826	1.640	0.890	0.919
	t=3	(6)**	1.207	0.454	1.000	0.954
	t=4	(6)**	1.165	0.351	0.959	0.895
Broyden		14	2.429	2.944	1.152	0.795
Brown		21	1.552	0.575	∞	---

*Based on 22 observations.

†Based on 16 observations listed in the tables presented.

**t = 2 is based on 11 observations; t = 3 is based on 10 observations.

because both the number of function evaluations and the accuracy of an algorithm to a particular problem are involved in Equation (4.1).

Examining the tables and previous discussion, the author tentatively makes the following conclusions.

(1) Gay-Schnabel's algorithms (algorithms I and I') and the first extension (algorithms II and II') are superior to Broyden's method.

(2) In Gay-Schnabel's proposal, the modification of Broyden's first method (algorithms I, II, and II) is superior to the modification of Broyden's second method (algorithms I', II', and III', respectively).

(3) Algorithm III is somewhat superior to Broyden's method.

(4) Algorithm IV is inferior to both algorithms I and I' in general. For large N, an appropriate choice of t in algorithm III will render algorithm III more effective.

(5) There is no optimal strategy of choosing an appropriate number t in algorithm III.

(6) The initial point affects the performance of an algorithm.

As to the application of the member of Gay-Schnabel's class to an unconstrained minimization problem, the author observed the following results:

(1) If a reasonably good initial estimate of the minimum is available or if the problem is an "easy" problem, then the members of the Gay-Schnabel class are superior to the DFP method.

(2) If a reasonably good initial estimate is not available, then the DFP method is superior to Gay-Schnabel's algorithm.

(3) The Gay-Schnabel algorithms may converge to a nonoptimal stationary point.

(4) The analytic gradient vectors of ϕ must be given.

All of the conclusions are somewhat tentative. The term "reasonably good" is itself rather vague. Furthermore, the behavior of the various methods depends upon the nature of the problem they are attempting to solve. Thus it is unlikely that the few test problems selected give a sufficiently accurate picture of the overall behavior of the algorithms. However, at least some members in Gay-Schnabel's class have been more efficient than Broyden's method. It may prove to be a useful alternative as Gay and Schnabel desired.

CHAPTER V

SUMMARY AND SUGGESTIONS FOR FURTHER WORK

Summary

This study is based on the Gay-Schnabel proposal (38), which is a modification of Broyden's method for solving systems of n nonlinear equations in R^n. The objectives of this thesis are: (1) to study the essential features and convergent properties for the following algorithms--Broyden's method, Gay-Schnabel's method, Brown's method, and the DFP (Davidon-Fletcher-Powell) method; (2) to implement the algorithms in Gay-Schnabel's proposal and compare their performance with that of the algorithms mentioned above.

The introduction, statement of the problem, basic concepts of quasi-Newton methods, and pertinent definitions of convergence properties are given in Chapter I. The algorithm and convergence properties of Broyden's method, the technique of nonlinear search, the Gay-Schnabel proposal and its extensions, which deal with using the orthogonal projection of current step onto the orthogonal complement of previous steps to generate a new step, are contained in Chapter II. Chapter III presents two effective and efficient methods: one for solving systems of equations, and another for solving unconstrained minimization problems. The two methods are Brown's method and the DFP method, respectively. A set of test problems and numerical results are given in Chapter IV. In

addition, a library write-up and a program listing for the BROGAY package are shown in Appendices A and B, respectively.

Suggestions for Further Work

In the computational experience, we verified that a good choice of the initial data often improves the effectiveness of an algorithm. Such a choice depends strongly on the properties of the particular class of equations under consideration. So far we do not have any good strategy for choosing an initial datum and a random guess depends heavily on the probability. The first suggestion for further work is to develop a localization method for constructing sets containing solutions in order to make a good guess for the general system of equations.

All of the algorithms covered in this study are designed to find one solution of a given problem. This solution, if obtained, may not be the desired one. For instance, some of the algorithms in Gay-Schnabel's method converged to a point when applied to the Wood function, but this obtained point was a non-optimal stationary point. The second suggestion for further work is to develop a method to approximate "multiple" roots or to find further solutions once one has been obtained.

We tested two systems comprising ten equations (problem 9, problem 10). We learned that some cases of algorithm III were more efficient than other members of Gay-Schnabel's method. We conjecture that an appropriate choice of t can change the algorithm from slow to fast. Since we only tested the cases $t = 2, 3, 4$ for the above two problems, we do not know if we can get a better performance of the algorithm for $t > 4$. The third suggestion for further work is to do more tests for a large system of equations with $N > 5$ and $t > 4$. This will involve a

slight revision of the program BROGAY. The fourth suggestion will be to develop a strategy of making an optimal choice of t for algorithm III.

BIBLIOGRAPHY

(1) Abbott, J. M. "Instability in Quasi-Newton Methods for the Optimisation of Nonlinear Functions." (Unpublished M.S. thesis, University of Essex, 1971.)

(2) Bard, Y. "On a Numerical Instability of Davidon-Like Method." Math. Prog., Vol. 22 (1968), 665-666.

(3) Barnes, J. G. P. "An Algorithm for Solving Non-Linear Equations Based on the Secant Method." Comp. J., Vol. 8 (1965), 66-72.

(4) Bittner, L. "Eine Verallgemeinerung des Sekantenverfahrens (regular falsi) Eur Näherungsweisen Berechnung des Nullstellen Eines Nichtlinearen Gleichungssystems." Wissen. Zeit der Technischen Hochschule Dresden, Vol. 9 (1959), 325.

(5) Brown, K. M. "A Quadratically Convergent Method for Solving Simultaneous Nonlinear Equations." (Unpublished Ph.D. thesis, Purdue University, 1966.)

(6) _____. "Solution of Simultaneous Non-Linear Equations." Comm. ACM, Vol. 10 (1967), 728-729.

(7) _____. "A Quadratically Convergent Newton-Like Method Based Upon Gaussian Elimination." SIAM H. Numer. Anal., Vol. 6 (1969), 560-569.

(8) _____. "Computer Oriented Algorithm for Solving Systems of Simultaneous Nonlinear Algebraic Equations." Numerical Solution of Systems of Nonlinear Algebraic Equations. New York: Academic Press, 1973, 281-348.

(9) Brown, K. M., and S. D. Conte. "The Solution of Simultaneous Nonlinear Equations." Proc. 22nd Nat. Conf. ACM. Washington, D.C.: Thompson Book Co., 1967, 111-114.

(10) Brown, K. M., and J. E. Dennis, Jr. "On the Second Order Convergence of Brown's Derivative-Free Method for Solving Simultaneous Nonlinear Equations." Technical Report 71-7. Department of Computer Science, Yale University, New Haven, Conn., 1971.

(11) Brown, K. M., and W. B. Gearhart. "Deflation Techniques for the Calculation of Further Solutions of a Nonlinear System." Numer. Math., Vol. 16 (1971), 334-342.

(12) Box, M. J. "A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems." Comp. J., Vol. 9 (1966), 67-77.

(13) Broyden, C. G. "A Class of Methods for Solving Nonlinear Simultaneous Equations." Math. Comp., Vol. 19 (1965), 368-381.

(14) _____. "Quasi-Newton Method and Their Application to Function Minimization." Math. Comp., Vol. 21 (1967), 368-381.

(15) _____. "A New Method of Solving Nonlinear Simultaneous Equations." Comp. J., Vol. 12 (1969), 95-100.

(16) _____. "The Convergence of Single-Rank Quasi-Newton Method." Math. Comp., Vol. 24 (1970), 365-382.

(17) _____. "Recent Developments in Solving Nonlinear Simultaneous Equations." Numerical Methods for Nonlinear Algebraic Equations. Ed. P. Rabinowitz. London: Gordon and Breach, 1970.

(18) _____. "Quasi-Newton Method." Numerical Methods for Unconstrained Optimization. Ed. W. Murray. New York: Academic Press, 1972, 87-106.

(19) Broyden, C. G., J. E. Dennis, and J. J. Moré. "On the Local and Superlinear Convergence of Quasi-Newton Method." J. Inst. Math. Appl., Vol. 12 (1973), 223-246.

(20) Cauchy, A. "Methods Generale pour La Resolution des Systems d'Equations Simultanees." C. R. Acad. Sci. Paris, Vol. 25 (1874), 536.

(21) Chandler, J. P. Private Communication, Oklahoma State University, 1978.

(22) Davidenko, D. F. "On a New Method of Numerical Solution of Systems of Nonlinear Equations." Dokl, Akad. Nauk. SSSR, Vol. 88 (1953), 601-602. (Russian), MR 14, 906.

(23) Davidon, W. C. "Variable Metric Method for Minimization." AEC Research and Development Report. ANL-5990 (Rev.). Lemont, Ill.: Argonne National Laboratory, 1959.

(24) _____. "Variance Algorithm for Minimization." Comp. J., Vol. 10 (1967-68), 406-410.

(25) _____. "Optimally Conditioned Optimization Algorithms With Line Searches." Math. Prog., Vol. 9 (1975), 1-30.

(26) Deist, F. H., and L. Sefor. "Solution of Systems of Nonlinear Equations by Parameter Variation." Comp. J., Vol. 10 (1967), 77-82.

(27) Dennis, J. E. "On the Convergence of Broyden's Method of Nonlinear System of Equations." Math. Comp. Vol. 25 (1971), 559-567.

(28) _____. "A Brief Survey of Convergence Results for Quasi-Newton Method." SIAM-AMS, Vol. 9 (1976), 185-199.

(29) _____. "Toward a Unified Convergence Theory for Newton-Like Method." Nonlinear Functional Analysis and Applications. Ed. L. Rall. New York: Academic Press, 1971, 425-472.

(30) Dennis, J. E., and J. J. Moré. "A Characterization of Superlinear Convergence and Its Application to Quasi-Newton Method." Math. Comp., Vol. 28 (1974), 549-559.

(31) _____. "Quasi-Newton Method, Motivation and Theory." SIAM REV., Vol. 19 (1977), 46-89.

(32) Dixon, L. C. W. "Quasi-Newton Techniques Generate Identical Points II: The Proofs of Four New Theorems." Math. Prog., Vol. 3 (1972), 345-358.

(33) Fletcher, R. "A New Approach to Variable Metric Algorithms." Comp. J., Vol. 13 (1970), 317-322.

(34) Fletcher, R., and M. J. D. Powell. "A Rapid Convergent Descent Method for Minimization." Comp. J., Vol. 6 (1963), 163-168.

(35) Freudenstein, F., and B. Roth. "Numerical Solutions of Systems of Nonlinear Equations." J. Assoc. Comput. Mach., Vol. 10 (1963), 550-556.

(36) Garcia-Palomares, U. M. "Superlinearly Convergent Quasi-Newton Methods for Nonlinear Programming." (Unpublished Ph.D. dissertation, University of Wisconsin, 1973.)

(37) Gay, D. M. "Convergence Properties of Broyden-Type Methods on Linear Systems of Equations." 1977 (to appear).

(38) Gay, D. M., and R. B. Schnabel. "Solving Systems of Nonlinear Equations by Broyden's Method With Projected Updates." 1977 (to appear).

(39) Goldfarb, D. "A family of Variable-Metric Methods Derived by Variational Means." Math. Comp., Vol. 24 (1970), 23-26.

(40) Greenstadt, J. "Variations on Variable-Metric Method." Math. Comp., Vol. 24 (1970), 1-18.

(41) Himmelblau, D. M. "Applied Nonlinear Programming." New York: McGraw-Hill, 1972.

(42) Huang, H. Y. "A Family of Variable-Metric Methods Derived by Variational Means." Math. Comp., Vol. 24 (1970), 23-26.

(43) International Mathematical and Statistical Libraries, Inc. LMSL Library 1, Reference Manual. 5th ed. Document IMSL LIB1-005 (Rei.). Houston: International Mathematical and Statistical Libraries, Inc., 1975.

(44) Lenard, M. L. "Practical Convergence Conditions for the Davidon-Fletcher-Powell Method." Math. Prof., Vol. 9 (1975), 69-86.

(45) McCormick, G. P. "On the Convergence and Rate of Convergence of the Reset Davidon Variable Metric Method." Technical Summary Report No. 1012. Mathematics Research Center, University of Wisconsin, 1969.

(46) McCormick, G. P., and J. D. Pearson. "Variable Metric Methods and Unconstrained Optimization." Optimization. Ed. R. Fletcher. New York: Academic Press, 1969, 307-325.

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

(48) Pearson, J. D. "Variable Metric Methods of Minimization." Comp. J., Vol. 12 (1969), 171-178.

(49) Powell, M. J. D. "An Iterative Method for Finding Stationary Values of a Function of Several Variables." Comp. J., Vol. 5 (1962), 147-151.

(50) _____. "A FORTRAN Subroutine for Solving Systems of Nonlinear Algebraic Equations." Numerical Method for Nonlinear Algebraic Equations. Ed. P. Rabinowitz. London: Gordon and Breach Science Publishers, 1970, 115-162.

(51) _____. "On the Convergence of the Variable Metric Algorithm." J. Inst. Math. Appl., Vol. 7 (1971), 21-36.

(52) _____. "Quadratic Termination Properties of Davidon's New Variable Metric Algorithm." Math. Prog., Vol. 12 (1977), 141-147.

(53) Rheinboldt, W. C. "Update Methods: Methods for Solving Systems of Nonlinear Equations." SIAM, Vol. 14 (1974), 53-67.

(54) Rosenbrock, H. H. "An Automatic Method for Finding the Greatest or Least Value of a Function." Comp. J., Vol. 3 (1960), 175-184.

(55) Shanno, D. F. "Conditioning of Quasi-Newton Methods for Function Minimization." Math. Comp., Vol. 24 (1970), 647-656.

(56) Stewart, G. W. Introduction to Matrix Computations. New York: Academic Press, 1973.

(57) Taylor, R. J. "The Davidon-Fletcher-Powell Method and Families of Variable Metric Methods for Unconstrained Minimization." (Unpublished Ed.D. dissertation, Oklahoma State University, 1976).

(58) Wolfe, P. "The Second Method for Simultaneous Nonlinear Equations." Comm. of the ACM, Vol. 2 (1959), 12-13.

APPENDIX A

LIBRARY WRITE-UP FOR THE BROGAY PACKAGE

Name of Routine: BROGAY

Language: A.N.S.I. Standard FORTRAN

Author: Guang-Nay Wang
Department of Mathematics
Oklahoma State University
Stillwater, Oklahoma 74074

Date: May, 1978

I. Purpose

BROGAY finds a solution of systems of N nonlinear equations to N unknowns, i.e., given

$$F : R^n \to R^n,$$

which is differentiable, BROGAY finds a point $X^* \in R^n$ such that $F(X^*) = 0$.

II. Method Used

The method used is based on the Gay-Schnabel proposal which is a modification of the Broyden method. A new step is determined in each iteration by the orthogonal projection of the current step onto the orthogonal complement of the previous step. This method is a member of the quasi-Newton method.

III. Use

A. Provide Main Program

The user must provide a main program to perform initialization, call BROGAY, etc.

B. FORTRAN Call

```
CALL BROGAY (F,N,X,PAR,FDSTEP,EPS,EPS2,C2,EPS3,NSEV,ITMAX,
             ITS,STEPMX,FX,FX1,DX,DF,H,IRC,METHOD,K)
```

where

F — External subroutine supplied by the user to evaluate the the function F, a zero of which is sought. The statement CALL F(X, FX, PAR) results in FX ← F(X). The parameter array PAR is passed unchanged.

N — $F : R^n \to R^n$.

X — A vector which contains the initial guess of the unknowns for input; contains the value of unknowns in the last iterate for output.

FDSTEP — If FDSTEP > 0, then H is initialized by computing a finite difference Jacobian with step size FDSTEP and inverting the result to obtain the initial H.

EPS — First stopping criterion: stop if $||F|| < eps$.

EPS2 — Second stopping criterion: stop if

$$||DX|| < eps2 * (C2 + ||X||) \text{ twice.}$$

C2 — Used in second stopping test for defending the case that X becomes a zero vector.

EPS3 — Stopping tolerance for nonlinear search technique: a new point X1 will be accepted if

$$||F(X1)|| < EPS3x||F(X)||.$$

NSEV — Maximum number of function evaluations during nonlinear search for new iterate X1; if exceeded, BROGAY returns with IRC = 6.

ITMAX — Maximum allowed number of iterations.

ITS — Number of iterations actually performed.

STEPMX — Number allowed step size = maxnorm (DX); if exceeded, the DX is replaced by (STEPMX/MAXNORM(DX))*DX.

FX — Current F(X).

FX1 — Next F(X).

DX — Change in X.

DF — Change in F.

H — Inverse Jacobian approximation.

IRC Return code:

= 1 first stopping test met

= 2 second stopping test met

= 3 both 1 and 2

= 4 maximum number of iteration performed without the above

= 5 singular Jacobian at start

= 6 nonlinear search technique cannot reduce $||F(X)||$ sufficiently

= 7 invalid value of n or method.

METHOD Method selection code:

= 0 Broyden's first method

= 1 Gay-Schnabel's extension to modify Broyden's first method

= 2 Gay-Schnabel's extension to modify Broyden's second method.

K Extension selection code:

= 1 Gay-Schnabel's first extension

= 2 Second extension with t = 2

= 3 Second extension with t = 3

= 4 Second extension with t = 4

calling program.

C. Error Information and Treatment

An error return can occur on any one of the following six conditions:

(1) Failure to converge--the nonlinear search technique cannot reduce $||F|| <$ EPS.

(2) $N \leq 0$ or $N \geq 10$.

(3) $k \leq 0$ or $K \geq 5$.

(4) METHOD = 0, 1, 2.

(5) Singular Jacobian at start.

(6) Maximum number of iterations performed.

D. Structure of the Program

(1) Calling program (MAIN).

(2) Subroutine F: contains the functions to be solved.

(3) Subroutine NLSRCH: Broyden's line search for nonlinear equations (from Gay-Schnabel's proposal). It can be supplied by user.

(4) Subroutine MATNV: matrix inversion to inverse the Jacobian matrix (from J. P. Chandler). It is available from a numerical source, such as IMSL.

(5) BLOCK DATA: the given X-vector elements for the solution are entered here.

(6) Subroutine REPROT: output subroutine.

E. Example Problem

$$f_1(X) = 10(X_2 - X_1)$$

$$f_2(X) = 1 - X_1$$

$$X_0 = (-1.2,\ 1)^T,\ X^* = (1,\ 1)^T.$$

The MAIN program and subroutines supplied for this test program are presented below, followed by the printout from BROGAY for the above problem.

```
//WANG     JOB TYPRUN=COPY                                                JOB  527
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL FSUBA
      REAL*8 X(10),PAR(10),FX(10),FX1(10),DX(10),DF(10),H(10,10)
      N=2
      DO 10 I=1,N
   10 PAR(I)=0.0D0
      X(1)=-1.2D0
      X(2)=1.0D0
      FDSTEP=0.000100
      EPS=0.1D-8
      EPS2=0.1D-7
      C2=0.1D-3
      EPS3=1.0D0
      NSEV=100
      ITMAX=200
      STEPMX=10.00
      METHOD=1
      K=1
      CALL BROGAY (FSUBA,N,X,PAR,FDSTEP,EPS,EPS2,C2,EPS3,NSEV,ITMAX,
     1             ITS,STEPMX,FX,FX1,DX,DF,H,IRC,METHOD,K)
      WRITE (6,111) (X(I),I=1,N)
  111 FORMAT (1H0,3X,'THE SOLUTION OF THE SYSTEM IS : ',2D24.16)
      STOP
      END
      SUBROUTINE FSUBA (X,FX,PAR)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 FX(10),X(10),PAR(10)
      FX(1)=10.0D0*(X(2)-X(1)*X(1))
      FX(2)=1.0D0-X(1)
      RETURN
      END
//

------ JES2 JOB STATISTICS ------

        33 CARDS READ

         0 SYSOUT PRINT RECORDS

         0 SYSOUT PUNCH RECORDS

       0.30 MINUTES EXECUTION TIME
```

I	NF	IRC	F	D(FNORM)	DF	DX	X-X*
0	3	0	0.4919D 01	0.0000D 00	0.0000D 00	0.0000D 00	0.2200D 01
1	6	0	0.4780D 01	0.1390D 00	0.1589D 00	0.3149D 00	0.2089D 01
2	9	0	0.4749D 01	0.3119D-01	0.6787D-01	0.1558D 00	0.2047D 01
3	12	0	0.4337D 01	0.4124D 00	0.6114D 00	0.5273D 00	0.2199D 01
4	14	0	0.3867D 01	0.4701D 00	0.5786D 00	0.1082D 01	0.1899D 01
5	17	0	0.3840D 01	0.2648D-01	0.6633D-01	0.7809D-01	0.1908D 01
6	20	0	0.3097D 01	0.7435D 00	0.1100D 01	0.6411D 00	0.2012D 01
7	22	0	0.1479D 01	0.1618D 01	0.2499D 01	0.7374D 00	0.1663D 01
8	25	0	0.1454D 01	0.2447D-01	0.1813D 00	0.4956D-01	0.1660D 01
9	29	0	0.1435D 01	0.1955D-01	0.1461D 00	0.5891D-01	0.1651D 01
10	31	0	0.1420D 01	0.1451D-01	0.6746D 00	0.3541D 00	0.1557D 01
11	33	0	0.1092D 01	0.3380D 00	0.3437D 00	0.3047D 00	0.1330D 01
12	36	0	0.1092D 01	0.2192D-03	0.1900D-02	0.1063D-02	0.1331D 01
13	39	0	0.1059D 01	0.2307D-01	0.1053D 00	0.4382D-01	0.1361D 01
14	41	0	0.7821D 00	0.2766D 00	0.2837D 00	0.2980D 00	0.1098D 01
15	44	0	0.7820D 00	0.1557D-03	0.1899D-02	0.1239D-02	0.1089D 01
16	47	0	0.7718D 00	0.1020D-01	0.6428D-01	0.3299D-01	0.1119D 01
17	49	0	0.5395D 00	0.2323D 00	0.2386D 00	0.3080D 00	0.8178D 00
18	52	0	0.5394D 00	0.8316D-04	0.1594D-01	0.1468D-01	0.8035D 00
19	55	0	0.5298D 00	0.9598D-02	0.6500D-01	0.4254D-01	0.8446D 00
20	57	0	0.3921D 00	0.1377D 00	0.1703D 00	0.2775D 00	0.5694D 00
21	60	0	0.3766D 00	0.1556D-01	0.4307D-01	0.7156D-01	0.4983D 00
22	64	0	0.3573D 00	0.1922D-01	0.4832D 00	0.9206D-01	0.5837D 00
23	66	0	0.1725D 00	0.1849D 00	0.1955D 00	0.2437D 00	0.3408D 00
24	68	0	0.1373D 00	0.3515D-01	0.1547D 00	0.1139D 00	0.2275D 00
25	72	0	0.1255D 00	0.1186D-01	0.1057D 00	0.3756D-01	0.2645D 00
26	74	0	0.8891D-01	0.3656D-01	0.8847D-01	0.1106D 00	0.1541D 00
27	75	0	0.7153D-01	0.1738D-01	0.7013D-01	0.1477D 00	0.7153D-02
28	77	0	0.1084D-01	0.6069D-01	0.8238D-01	0.8238D-02	0.1084D-02
29	78	0	0.1388D-15	0.1084D-01	0.1084D-01	0.1084D-02	0.1388D-16
29	78	1	0.1388D-15	0.0000D 00	0.1084D-01	0.1084D-02	0.1388D-16

THE SOLUTION OF THE SYSTEM IS : 0.1000000000000000D 01 0.1000000000000000D 01

IV. For Large Program

If N > 10, it will be necessary to make dimension change in all routines.

V. Reference

Gay, D. M., and R. B. Schnabel. "Solving Systems of Nonlinear Equations by Broyden's Method With Projected Updates." Department of Computer Science, Cornell University, 1977.

APPENDIX B

PROGRAM LISTING

```
      $JOB TIME=10,NOSUBCHK
 1          IMPLICIT REAL*8 (A-H,O-Z)
 2          EXTERNAL FSUBA
 3          REAL*8 X(10),PAR(10),FX(10),FX1(10),DX(10),DF(10),H(10,10)
 4          N=2
 5          DO 10 I=1,N
 6       10 PAR(I)=0.0D0
 7          X(1)=-1.2D0
 8          X(2)=1.0D0
 9          FDSTEP=0.000100
10          EPS=1.D-8
11          EPS2=0.1D-7
12          C2=0.1D-3
13          EPS3=1.0D0
14          NSEV=100
15          ITMAX=200
16          STEPMX=10.D0
17          METHOD=1
18          K=1
19          CALL BROGAY (FSUBA,N,X,PAR,FDSTEP,EPS,EPS2,C2,EPS3,NSEV,ITMAX,
           1             ITS,STEPMX,FX,FX1,DX,DF,H,IRC,METHOD,K)
20          WRITE (6,111) (X(I),I=1,N)
21      111 FORMAT (1H0,3X,'THE SOLUTION OF THE SYSTEM IS : ',2D24.16)
22          STOP
23          END

24          SUBROUTINE FSUBA (X,FX,PAR)
25          IMPLICIT REAL*8 (A-H,O-Z)
26          REAL*8 FX(10),X(10),PAR(10)
27          FX(1)=10.0D0*(X(2)-X(1)*X(1))
28          FX(2)=1.0D0-X(1)
29          RETURN
30          END

31          SUBROUTINE BROGAY  (F,N,X,PAR,FDSTEP,EPS,EPS2,C2,EPS3,NSEV,ITMAX,
           1             ITS,STEPMX,FX,FX1,DX,DF,H,IRC,METHOD,K)
      C*****************************************************************
      C*                                                               *
      C*       THIS PROGRAM IS DESIGNED TO TEST THE ALGORITHMS DEVELOPED BY  *
      C*    D. M. GAY AN R. B. SCHNABEL, WHICH IS A MODIFICATION OF THE BROY-*
      C*    DEN METHOD FOR SOLVING SYSTEMS OF NONLINEAR EQUATIONS IN N UNKNOW*
      C*    IN THIS METHOD, A NEW STEP IS DETERMINED AT EACH ITERATION BY THE*
      C*    ORTHOGONAL PROJECTION OF THE CURRENT STEP ONTO THE ORTHOGONAL     *
      C*    COMPLEMENT OF THE PREVIOUS STEPS. THIS METHOD IS A MEMBER OF THE *
      C*    QUASI-NEWTON METHOD. DIFFERENT VERITIONS ARE OBTAINED BY TAKING  *
      C*    DIFFERENT NUMBER OF PREVIOUS STEPS.                              *
      C*                                                                     *
      C*    ORIGINAL SOURCE : GAY, D. M. AND R. B. SCHNABEL,                 *
      C*       'SOLVING SYSTEMS OF NONLINEAR EQUATIONS BY BROYDEN'S METHOD   *
      C*       WITH PROJECTED UPDATES.' (1977)                               *
      C*                                                                     *
      C*    PROGRAMMER : GUANG-NAY WANG                                      *
      C*                 DEPARTMENT OF MATHEMATICS                           *
      C*                 OKLAHOMA STATE UNIVERSITY                           *
      C*                                                                     *
      C*    VARIABLES :                                                      *
      C*       F       EXTERNAL SUBROUTINE SUPPLIED BY THE USER TO EVALUATE*
      C*                THE FUNCTION F, A ZERO OF WHICH IS SOUGHT. THE       *
      C*                STATEMENT CALL F(X,FX,PAR) RESULTS IN FX <-- F(X).   *
      C*                THE PARAMETER ARRAY PAR IS PASSED UNCHANGED.         *
```

```
C*       N          NUMBER OF UNKNOWS AND NUMBER OF EQUATIONS                *
C*       X          A VECTOR WHICH CONTAINS THE INITIAL GUESS OF THE         *
C*                  UNKNOWS FOR INPUT; CONTAINS THE VALUE OF UNKNOWS IN      *
C*                  LAST ITERATE FOR OUTPUT.                                 *
C*       FDSTEP     IF FDSTEP>0 THEN H IS INITIALIZED BY COMPUTING A         *
C*                  FINITE DIFFERENCE JACOBIAN WITH STEP SIZE FDSTEP AND     *
C*                  INVERTING THE RESULT TO OBTAIN THE INITIAL H.            *
C*       EPS        FIRST STOPPING CRITERION : STOP IF ||F|| < EPS           *
C*       EPS2       SECOND STOPPING CRITERION : STOP IF                      *
C*                  ||DX|| < EPS2*(C2+||X||) TWICE                           *
C*       C2         USED IN SECOND STOPPING TEST FOR DEFENDING THE CASE      *
C*                  X BEING A ZERO VECTOR,                                   *
C*       EPS3       STOPPING TOLERENCE FOR NONLINEAR SEARCH TECHNIQUE:       *
C*                  A NEW POINT X1 WILL BE ACCEPTED IF                       *
C*                  ||F(X1)|| < EPS3*||F(X)||.                               *
C*       NSEV       MAXIMUM NUMBER OF FUNCTION EVALUATIONS DURING            *
C*                  NONLINEAR SEARCH FOR NEW ITERATE X1; IF EXCEEDED,        *
C*                  BROGAY RETURNS WITH IRC=6                                *
C*       ITMAX      MAXIMUM ALLOWED NUMBER OF ITERATIONS                     *
C*       ITS        NUMBER OF ITERATIONS ACTUALLY PERFORMED.                 *
C*       STEPMX     LARGEST ALLOWED STEPSIZE=MAXNORM(DX); IF EXCEEDED,       *
C*                  THE DX IS REPLACED BY (STEPMX/MAXNORM(DX))*DX            *
C*       FX         CURRENT F(X)                                             *
C*       FX1        NEXT F(X)                                                *
C*       DX         CHANGE IN X                                              *
C*       DF         CHANGE IN F                                              *
C*       H          INVERSE JACOBIAN APPROXIMATION.                          *
C*       IRC        RETURN CODE :                                            *
C*                  IRC=1    FIRST STOPPING TEST MET                         *
C*                  IRC=2    SECOND STOPPING TEST MET                        *
C*                  IRC=3    BOTH 1 & 2                                      *
C*                  IRC=4    MAXIMUM NUMBER OF ITERATION PERFORMED           *
C*                           WITHOUT THE ABOVE                               *
C*                  IRC=5    SINGULAR JACOBIAN AT START                      *
C*                  IRC=6    NONLINEAR SEARCH TECHNIQUE CANNOT REDUCE        *
C*                           ||F(X)|| SUFFICIENTLY.                          *
C*                  IRC=7    INVALID VALUE OF N AND METHOD.                  *
C*       METHOD     METHOD SELECTION CODE                                    *
C*                  =0       BROYDEN'S FIRST METHOD.                         *
C*                  =1       MODIFICATION OF BROYDEN'S FIRST METHOD          *
C*                  =2       MODIFICATION OF BROYDEN'S SECOND METHOD         *
C*       K          GAY-SCHNABEL'S EXTENSION SELLECTION CODE                 *
C*                  K=1      GAY-SCHNABEL'S FIRST EXTENSION                  *
C*                  K>1      GAY-SCHNABEL'S SECOND EXTENSION                 *
C*                           =2, CASE T=2                                    *
C*                           =3, CASE T=3                                    *
C*                           =4, CASE T=4                                    *
C*                                                                           *
C*    SUBROUTINES :                                                          *
C*       GENP1      GENERATE ORTHOGONAL PROJECTION OF CURRENT STEP ONTO       *
C*                  THE ORTHOGONAL COMPLEMENT OF PREVIOUS ONE STEP           *
C*       GENP2      GENERATE ORTHOGONAL PROJECTION OF CURRENT STEP ONTO       *
C*                  THE ORTHOGONAL COMPLEMENT OF PREVIOUS TWO STEPS           *
C*       GENP3      GENERATE ORTHOGONAL PROJECTION OF CURRENT STEP ONTO       *
C*                  THE ORTHOGONAL COMPLEMENT OF PREVIOUS THREE STEPS         *
C*       GENP4      GENERATE ORTHOGONAL PROJECTION OF CURRENT STEP ONTO       *
C*                  THE ORTHOGONAL COMPLEMENT OF PREVIOUS FOUR STEPS          *
C*       XDOTP      DO INNER PRODUCT OF TWO VECTORS                          *
C*       MAXNRM     FIND THE MAXIMUM NORM OF A GIVEN VECTOR                  *
C*       TWONRM     FIND THE L2 NORM OF A GIVEN VECTOR                       *
```

```
C*        DIFF        FIND THE DIFFERENCE OF TWO VECTORS                  *
C*        ADDMUL      SUMMATION OF ONE VECTOR WITH A SCALOR MULTIPLE OF   *
C*                    ANOTHER VECTOR                                      *
C*        VCMUL       A SCALOR MULTIPLE OF A GIVEN VECTOR                 *
C*        F           EXTERNAL SUBROUTINE CONTAINS THE FUNCTIONS TO SOLVE *
C*        REPORT      OUTPUT SUBROUTINE                                   *
C*        N_SRCH      BROYDEN'S LINE SEARCH FOR NONLINEAR EQUATIONS       *
C*                    FROM GAY-SCHNABEL'S PROPOSAL. IT CAN BE SUPPLIED    *
C*                    BY THE USER.                                        *
C*        MATNV       MATRIX INVERSION SUBROUTINE BY DR. J.P.CHANDER      *
C*        BLOCK DATA      THE GIVEN SOLUTION ENTERED HERE                 *
C*                                                                        *
C************************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL F
      REAL*8 X(10),PAR(10),FX(10),FX1(10),DX(10),DF(10),H(10,10),A(10),
     1       V(10),Z(10),SBAR(10),PREV1(10),PREV2(10),PREV3(10),
     2       PREV4(10)
      IRC=7
      IF (N .LE. 0 .OR. METHOD .LT. 0 .OR. METHOD .GT. 2) RETURN
C*******
C
C         PARAMETER INITIALIZATION
C
C*******
      NSAME=0
      DXNORM=0.D0
      DFNORM=0.D0
      K0=0
      IRC=0
      ITS=0
      LOOP=0
      CALL F(X,FX,PAR)
      CALL TWONRM (N,FNORM,FX)
      NF=1
C
C                                   INITIALIZE H IF NECESSARY
C
      IF (FDSTEP) 40,40,10
   10 DELTAX=1.D0/FDSTEP
      DO 30 J=1,N
         DX(J)=X(J)
         X(J)=X(J)+FDSTEP
         CALL F (X,FX1,PAR)
         DO 20 I=1,N
            DF(I)=FX1(I)-FX(I)
            H(I,J)=DF(I)*DELTAX
   20       CONTINUE
         X(J)=DX(J)
   30    CONTINUE
      NF=NF+N
      M=0
      MA=10
C*******
C
C         JACOBIAN INVERSION AND SINGULARITY CHECKING
C
C*******
      CALL MATINV (H,N,M,P,DET,MA)
      IF (M) 40,40,650
```

```
C*******
C
C         MAJOR LOOP STARTS
C
C*******
   40  CONTINUE
       CALL REPORT (ITS,KO,NF,N,DXNORM,X,FNORM,DFNORM,IRC)
C*******
C
C         FIRST STOPPING TEST  :
C            FIRST STOPPING MEETS IF ||F|| .LE. EPS OR THE NUMBER
C            OF ITERATIONS EXCEEDS LIMIT
C
C*******
       IF (FNORM-EPS) 50,50,60
   50  IRC=IRC+1
   60  IF (IRC-1) 70,670,670
   70  IF (ITS-ITMAX) 80,640,640
   80  ITS=ITS+1
C*******
C
C         DX <-- H*FX
C
C*******
       DO 100 I=1,N
          DO 90 J=1,N
   90        A(J)=H(I,J)
  100     CALL XDUOTP (A,FX,N,DX(I))
C*******
C
C         ENSURE MAXNORM(DX) .LE. STEPMX AND X <-- X-DX
C
C*******
       CALL MAXNRM (N,DXNORM,DX)
       IF (DXNORM-STEPMX) 130,130,110
  110  DO 120 I=1,N
  120     DX(I)=STEPMX/DXNORM*DX(I)
       DXNORM=STEPMX
C*******
C
C         SECOND STOPPING TEST
C
C*******
  130  CALL NLSRCH (F,N,FX1,X,DX,EPS3,FNORM,PAR,NSEV,NF,STEPMX/DXNORM)
       IF (FNORM) 660,140,140
  140  CALL TWONRM (N,DXNORM,DX)
       CALL TWONRM (N,XNORM,X)
C*******
C
C         SECOND STOPPING MEETS IF ||DX|| .LE. EPS2*||X||
C         IN TWO CONSECUTIVE TIMES
C
C*******
       EPSDX=EPS2*(XNORM+C2)
       IF (DXNORM-EPSDX) 160,160,150
  150  NSAME=0
       GO TO 180
  160  NSAME=NSAME+1
       IF (NSAME-2) 180,170,170
  170  IRC=2
```

```
 91          GO TO 40
     C*******
     C
     C          UPDATE H
     C          DF <-- FX1-FX
     C
     C*******
 92      180 DO 190 I=1,N
 93      190     DF(I)=FX1(I)-FX(I)
 94          CALL TWONRM  (N,DFNORM,DF)
 95          IF (DFNORM) 40,40,200
     C*******
     C
     C          Z <-- DX-H*DF
     C
     C*******
 96      200 LOP=LOP+1
 97          DO 220 I=1,N
 98          DO 210 J=1,N
 99      210 A(J)=H(I,J)
100          CALL XDOTP (A,DF,N,V(I))
101      220 Z(I)=DX(I)-V(I)
     C*******
     C
     C          USE FX AS A TEMPORARY ARRAY FOR DX OR DF
     C
     C*******
102          IF (METHOD-1) 370,230,250
     C*******
     C
     C          METHOD=0 : BROYDEN'S FIRST METHOD
     C          METHOD=1 : MODIFICATION OF BROYDEN' FIRST METHOD
     C          M2 : D=1 : MODIFICATION OF BROYDEN' SECOND METHOD
     C
     C*******
103      230 DO 240 I=1,N
104      240 FX(I)=DX(I)
105          GO TO 270
106      250 DO 260 I=1,N
107      260 FX(I)=DF(I)
     C*******
     C
     C          GENERATE SBAR IN DIFFERENT CASES
     C             K=1 : GAY-SCHNABEL'S FIRST EXTENSION
     C             K>1 : GAY-SCHNABEL'S SECOND EXTENSION
     C                   K=2 : CASE T=2
     C                   K=3 : CASE T=3
     C                   K=4 : CASE T=4
     C
     C*******
108      270 IF (LOP-K) 280,290,340
109      280 GO TO (290,310,320,330), LOP
110      290 DO 300 I=1,N
111      300 SBAR(I)=FX(I)
112          GO TO 360
113      310 CALL GENP1( N,PREV1,FX,SBAR)
114          GO TO 360
115      320 CALL GENP2 (N,PREV1,PREV2,FX,SBAR)
116          GO TO 360
117      330 CALL GENP3 (N,PREV1,PREV2,PREV3,FX,SBAR)
```

```
118           GO TO 360
119       340 GO TO (310,320,330,350), K
120       350 CALL GENP4 (N,PREV1,PREV2,PREV3,PREV4,FX,SBAR)
121       360 IF (METHOD-1) 370,390,420
122       370 DO 380 I=1,N
123       380 SBAR(I)=DX(I)
      C*******
      C
      C           V <-- (H**T)*SBAR
      C
      C*******
124       390 DO 410 J=1,N
125           DO 400 I=1,N
126       400 A(I)=H(I,J)
127       410 CALL XDOTP (A,SBAR,N,V(J))
128           GO TO 440
129       420 DO 430 I=1,N
130       430 V(I)=SBAR(I)
131       440 CALL XDOTP(V,DF,N,VDOTDF)
      C*******
      C
      C           IF SBAR=0 THEN H(K+1)=H(K)
      C                                 IF SBAR=0 THEN H(K+1)=H(K)
      C
      C*******
132           IF (VDOTDF) 445,465,445
133       445 VDOTDF=1.0D0/VDOTDF
134           DO 450 I=1,N
135       450 V(I)=V(I)*VDOTDF
      C*******
      C
      C           H <-- H+Z*(V**T)
      C
      C*******
136           DO 460 J=1,N
137           DO 460 I=1,N
138       460 H(I,J)=H(I,J)+Z(I)*V(J)
      C*******
      C
      C           STORE THE LATEST K S'S
      C
      C*******
139       465 IF (METHOD-1) 490,470,470
140       470 IF (_D7-K) 490,480,570
141       480 GO TO (490,510,530,550),LOP
142       490 DO 500 I=1,N
143           PREV1(I)=FX(I)
144       500 FX(I)=FX1(I)
145           GO TO 40
146       510 DO 520 I=1,N
147           PREV2(I)=FX(I)
148       520 FX(I)=FX1(I)
149           GO TO 40
150       530 DO 540 I=1,N
151           PREV3(I)=FX(I)
152       540 FX(I)=FX1(I)
153           GO TO 40
154       550 DO 560 I=1,N
155           PREV4(I)=FX(I)
156       560 FX(I)=FX1(I)
```

```
157         GO TO 40
158     570 GO TO (490,580,600,620),K
159     580 DO 590 I=1,N
160         PREV1(I)=PREV2(I)
161         PREV2(I)=FX(I)
162     590 FX(I)=FX1(I)
163         GO TO 40
164     600 DO 610 I=1,N
165         PREV1(I)=PREV2(I)
166         PREV2(I)=PREV3(I)
167         PREV3(I)=FX(I)
168     610 FX(I)=FX1(I)
169         GO TO 40
170     620 DO 630 I=1,N
171         PREV1(I)=PREV2(I)
172         PREV2(I)=PREV3(I)
173         PREV3(I)=PREV4(I)
174         PREV4(I)=FX(I)
175     630 FX(I)=FX1(I)
176         GO TO 40
      C*******
      C
      C         MAXIMUM NUMBER OF ITERATION PERFORMED
      C
      C*******
177     640 IRC=4
178         GO TO 670
      C*******
      C
      C         SINGULAR JACOBIAN AT START
      C
      C*******
179     650 IRC=5
180         GO TO 670
      C*******
      C
      C         NLSRCH CANNOT REDUCE YYF(X)|| SUFFICIENTLY
      C
      C*******
181     660 IRC=6
182         CALL MAXNRM (N,DXNORM,DX)
183         CALL MAXNRM (N,DFNORM,DF)
184     670 CALL FDPRDT (ITS,KF,NF,N,DXNORM,X,FNORM,DFNORM,IRC)
185         RETURN
186         END

187         SUBROUTINE GENP1 (N,U,V,W)
      C*******
      C
      C       W IS THE ORTHOGONAL PROJECTION OF V ONTO THE ORTHOGONAL
      C       COMPLEMENT OF U
      C
      C*******
188         IMPLICIT REAL*8 (A-H,O-Z)
189         REAL*8 U(10),V(10),W(10)
190         CALL XDOTP (U,V,N,DUMER)
191         CALL XDOTP (U,U,N,DENOM)
192         T=DUMER/DENOM
193         DO 100 I=1,N
194     100 W(I)=V(I)-U(I)*T
```

```
      RETURN
      END

      SUBROUTINE GENP2 (N,U1,U2,V,W)
C*******
C
C     W IS THE ORTHOGONAL PROJECTION OF V ONTO THE ORTHOGONAL COMPLEMENT
C     U1 AND U2
C
C*******
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 U1(10),U2(10),V(10),W(10),B2(10),B3(10)
      CALL GENP1 (N,U1,U2,B2)
      CALL GENP1 (N,U1,V,B3)
      CALL XDOOTP (V,B2,N,DUMER)
      CALL XDOOTP (B2,B2,N,DENOM)
      T=DUMER/DENOM
      DO 200 I=1,N
  200 W(I)=B3(I)-B2(I)*T
      RETURN
      END

      SUBROUTINE GENP3 (N,U1,U2,U3,V,W)
C*******
C
C     W IS THE ORTHOGONAL PROJECTION OF V ONTO THE ORTHOGONAL COMPLEMENT
C     U1,U2,U3
C
C*******
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 U1(10),U2(10),U3(10),V(10),W(10),B3(10),B4(10)
      CALL GENP2 (N,U1,U2,U3,B3)
      CALL GENP2 (N,U1,U2,V,B4)
      CALL XDOOTP (V,B3,N,DUMER)
      CALL XDOOTP (B3,B3,N,DENOM)
      T=DUMER/DENOM
      DO 300 I=1,N
  300 W(I)=B4(I)-B3(I)*T
      RETURN
      END

      SUBROUTINE GENP4 (N,U1,U2,U3,U4,V,W)
C*******
C
C     W IS ORTHOGONAL PROJECTION OF V ONTO THE ORTHOGONAL COMPLEMENT OF
C     U1,U2,U3,U4
C
C*******
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 U1(10),U2(10),U3(10),U4(10),V(10),W(10),B4(10),B5(10)
      CALL GENP3 (N,U1,U2,U3,U4,B4)
      CALL GENP3 (N,U1,U2,U3,V,B5)
      CALL XDOOTP (V,B4,N,DUMER)
      CALL XDOOTP (B4,B4,N,DENOM)
      T=DUMER/DENOM
      DO 400 I=1,N
  400 W(I)=B5(I)-B4(I)*T
      RETURN
      END
```

```
233         SUBROUTINE XDOTP (X,Y,N,PROD)
      C********
      C
      C       PROD <-- INNER PRODUCT OF X AND Y
      C
      C********
234         IMPLICIT REAL*8 (A-H,O-Z)
235         REAL*8 X(10),Y(10)
236         PROD=0.0D0
237         DO 10 I=1,N
238      10    PROD=PROD+X(I)*Y(I)
239         RETURN
240         END

241         SUBROUTINE MAXNRM (N,U,V)
      C********
      C
      C       U <-- MAXNORM(V)
      C
      C********
242         IMPLICIT REAL*8 (A-H,O-Z)
243         REAL*8 V(10)
244         U=0.0D0
245         DO 20 I=1,N
246            S=DABS(V(I))
247            IF (U-S) 10,20,20
248      10    U=S
249      20    CONTINUE
250         RETURN
251         END

252         SUBROUTINE TWONRM (N,U,V)
      C********
      C
      C       U <-- TWONORM(V)
      C
      C********
253         IMPLICIT REAL*8 (A-H,O-Z)
254         REAL*8 V(10)
255         T1=0.0D0
256         DO 10 I=1,N
257      10    T1=DMAX1(T1,DABS(V(I)))
258         S=0.0D0
259         IF (T1) 40,40,20
260      20 S1=1.0D0/T1
261         DO 30 I=1,N
262            T=V(I)*S1
263      30    S=S+T*T
264      40 U=DSQRT(S)*T1
265         RETURN
266         END

267         SUBROUTINE DIFF (N,U,V,W)
      C********
      C
      C       U <-- V-W
      C
      C********
268         IMPLICIT REAL*8 (A-H,O-Z)
269         REAL*8 V(10),U(10),W(10)
```

```
      DO 201 I=1,N
  201 U(I)=V(I)-W(I)
      RETURN
      END

      SUBROUTINE ADDMUL (N,U,V,W,X)
C*******
C
C     U <-- V+W*X
C
C*******
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 U(10),V(10),X(10)
      DO 301 I=1,N
  301 U(I)=V(I)+W*X(I)
      RETURN
      END

      SUBROUTINE VCMUL (N,U,V,W)
C*******
C
C     U <-- V*W
C
C*******
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 U(10),W(10)
      DO 1301 I=1,N
 1301 U(I)=V*W(I)
      RETURN
      END

      SUBROUTINE REPROT (I,K,NF,N,DXNORM,X,FNORM,DFNORM,IRC)
      IMPLICIT REAL*8(A-H,O-Z)
C******
C
C        PRINTER SUBROUTINE FOR BROGAY
C
C*******
      COMMON/LSTPAR/XSTAR(10)
      REAL*8 X(10),DXSTAR
      IF (I.EQ.0) F0=0.0D0
      DF=F0-FNORM
      F0=FNORM
      IF (I.GE. 1) GO TO 10
      WRITE (6,1001)
 1001 FORMAT ('1    I   NF IRC          F          D(FNORM)          DF
     1 DX          X-X*'/
     2'  ---  --  --  ----------   ----------   ----------   ----------
     3   ----------')
      DF=0.0D0
   10 DXSTAR=0.0D0
      DO 20 J=1,N
   20 DXSTAR=DXSTAR+(X(J)-XSTAR(J))**2
      DXSTAR=DSQRT(DXSTAR)
      WRITE (6,1002) I,NF,IRC,FNORM,DF,DFNORM,DXNORM,DXSTAR
 1002 FORMAT(1X,3I4,5E13.4)
      RETURN
      END
C*
```

```
      SUBROUTINE NLSRCH(CALCF, N, F, X, P, EPS, NORM, PAR, ITLIM,
     1                  NFCALL, TMAX)
      IMPLICIT REAL*8(A-H,O-Z)

***   BROYDEN'S "SEARCH" ROUTINE FOR NONLINEAR EQUATION SOLVING  ***

***   PROGRAMMER:  DAVID M GAY; LATEST CHANGE: 21 SEPT. 1976  ***

------------------------------  SUMMARY  ------------------------------

        FIND  T  SUCH THAT  ||F(X - T*P)|| < EPS*||F(X)|| BY FIRST TRY-
     ING T = 1, THEN T DETERMINED BY BROYDEN'S CUBIC ERROR TERM IDEA,
     THEN (FROM THE  THIRD EVALUATION ON) BY MINIMIZING THE QUADRATIC
     INTERPOLATING POLYNOMIAL ABOUT THE THREE NEAREST RECENT POINTS.
     BEFORE RETURNING, SET  P <-- -T*P = (THE STEP TAKEN).

***   REFERENCE:  BROYDEN, C.G.(1965), "A CLASS OF METHODS FOR SOLVING
***    NONLINEAR SIMULTANEOUS EQUATIONS", MATH. COMP. 19, PP. 577-593.

------------------------------  PARAMETERS  ------------------------------

***...PARAMETER USAGE CODES:  --> = INPUT; <-> = I/O; <-- = OUTPUT.

  CALCF --> SUBROUTINE FOR EVALUATING FUNCTION WHOSE ZERO IS SOUGHT:
              CALLING SEQUENCE:  CALL CALCF(X, F, PAR):
                 X = ARGUMENT, F = RESULT, PAR = PRIVATE PARAMETERS.
      N --> F MAPS  N-SPACE  TO  N-SPACE.
      F <-- F(X) AT RETURN.
      X --> INPUT:  START OF THE SEARCH; OUTPUT:LAST POINT AT WHICH
                 F  WAS EVALUATED.
      P <-> INPUT: -(SEARCH DIRECTION); OUTPUT: STEP TAKEN.
    EPS --> STOPPING TOLERANCE: STOP WHEN ||F(X-T*P)|| < EPS*||F(X)||.
   NORM <-> INPUT: ||F(X)|| (TWONORM); OUTPUT:  ||F(X)|| (NEW X). PRO-
              VIDED STOPPING TEST MET; OTHERWISE -||F(X)|| (IF ITLIM
              EXCEEDED).
    PAR --> PRIVATE PARAMETER ARRAY PASSED UNCHANGED TO  CALCF.
  ITLIM --> MAXIMUM FUNCTION EVALUATIONS DURING THIS CALL ON  NRSRCH:
              IF EXCEEDED, RETURN WITH  NORM = -||F(X)||.
 NFCALL <-> TOTAL NUMBER OF FUNCTION EVALUATIONS:  INCREMENTED AT EACH
              CALL OF  CALCF.
   TMAX --> ONLY CONSIDER POINTS BETWEEN  X + TMAX*P  AND  X - TMAX*P.

********************************************************************

***  PARAMETER DECLARATIONS  ***

      SUBROUTINE CALCF
      INTEGER ITLIM, N, NFCALL
      REAL*8 EPS,F(10),NORM,P(10),PAR(10),TMAX,X(10)

***  LOCAL VARIABLES  ***

      SUBROUTINE LINALG
      INTEGER I, L
      REAL*8 A1, A2, D1, D2, D3, NORM0, NRMLIM, PHI(3), T, TMIN, T0,
     1       VT(3)

***  OPERATION CODES FOR LINEAR ALGEBRA ROUTINE  LINALG  ***
```

```
C------------------------------- BODY --------------------------------NLS00670
C                                                                     NLS00680
      NORMLIM = EPS*NORM                                              NLS00690
      TMIN = -TMAX                                                    NLS00700
      NORMO = NORM                                                    NLS00710
      CALL DIFF (N,Y,X,P)                                             NLS00720
      CALL CALCF(X, F, PAR)                                           NLS00730
      NFCALL = NFCALL + 1                                             NLS00740
      CALL TWONRM (N,NORM,F)                                          NLS00750
      T = 1.D0                                                        NLS00760
      IF (NORM .LT. NRMLIM) GO TO 8000                                NLS00770
      TO = T                                                          NLS00780
      L = 2                                                           NLS00790
      VT(1) = 0.D0                                                    NLS00800
      PHI(1) = NORMO                                                  NLS00810
      VT(3) = T                                                       NLS00820
      PHI(3) = NORM                                                   NLS00830
      T = NORM/NORMO                                                  NLS00840
      T = (DSQRT(1.D0 + 6.D0*T) - 1.D0)/(3.D0*T)                      NLS00850
      DO 1000 I = 2, ITLIM                                            NLS00860
      CALL ADDVUL (N,X,Y,TO-T,P)                                      NLS00870
          TO = T                                                      NLS00880
          CALL CALCF(X, F, PAR)                                       NLS00890
          NFCALL = NFCALL + 1                                         NLS00900
      CALL TWONRM (N,NORM,F)                                          NLS00910
          IF (NORM .LT. NRMLIM) GO TO 8000                            NLS00920
          IF (I .GE. ITLIM) GO TO 1000                                NLS00930
          VT(L) = T                                                   NLS00940
          PHI(L) = NORM                                               NLS00950
          D1 = VT(2) - VT(3)                                          NLS00960
          D2 = VT(3) - VT(1)                                          NLS00970
          D3 = VT(1) - VT(2)                                          NLS00980
          A2 = -(PHI(1)*D1 + PHI(2)*D2 + PHI(3)*D3)/(D1*D2*D3)        NLS00990
          IF (A2 .LE. 0.D0) GO TO 100                                 NLS01000
          A1 = (PHI(1) - PHI(2))/D3 - A2*(VT(1) + VT(2))              NLS01010
          T = -A1/(2.*A2)                                             NLS01020
          GO TO 300                                                   NLS01030
  100     IF (PHI(3) .LE. PHI(1)) GO TO 200                           NLS01040
          T = 3.D0*VT(1) - 2.D0*VT(2)                                 NLS01050
          GO TO 300                                                   NLS01060
  200     T = 3.D0*VT(3) - 2.D0*VT(2)                                 NLS01070
  300     IF (T .LE. VT(3)) GO TO 400                                 NLS01080
          IF (VT(3) .GE. TMAX) GO TO 2000                             NLS01090
          T = DMIN1(T, TMAX)                                          NLS01100
          VT(1) = VT(2)                                               NLS01110
          PHI(1) = PHI(2)                                             NLS01120
          VT(2) = VT(3)                                               NLS01130
          PHI(2) = PHI(3)                                             NLS01140
          L = 3                                                       NLS01150
          GO TO 1000                                                  NLS01160
  400     IF (T .GE. VT(1)) GO TO 500                                 NLS01170
          IF (VT(1) .LE. TMIN) GO TO 2200                             NLS01180
          T = DMAX1(T, TMIN)                                          NLS01190
          VT(3) = VT(2)                                               NLS01200
          PHI(3) = PHI(2)                                             NLS01210
          VT(2) = VT(1)                                               NLS01220
          PHI(2) = PHI(1)                                             NLS01230
          L = 1                                                       NLS01240
          GO TO 1000                                                  NLS01250
  500     IF (T .LE. VT(2)) GO TO 600                                 NLS01260
```

```
      IF (T .GE. VT(3)) GO TO 2000                                      NLS01230
      L = 3                                                             NLS01240
      GO TO 1000                                                        NLS01250
800   IF (T .LE. VT(1) .OR. T .GE. VT(2)) GO TO 2000                    NLS01260
      L = 1                                                             NLS01270
1000  CONTINUE                                                          NLS01280
      NORM = -NORM                                                      NLS01290
      GO TO 8000                                                        NLS01300
2000 T = TO                                                             NLS01310
      NORM = -NORM                                                      NLS01320
8000 CALL VCMUL (N,P,-T,P)
      RETURN                                                            NLS01340
      END                                                               NLS01350

      SUBROUTINE MATINV (A,N,B,M,DET,MA)
      IMPLICIT REAL*8 (A-H,O-Z)
C                                                                       MATNV  2
C MATNV 4.0        A.N.S.I. STANDARD FORTRAN        NOVEMBER 1974       MATNV  3
C                                                                       MATNV  4
C CO-OP 61 NBSB MATINV - MATRIX INVERSION WITH SOLUTION OF LINEAR EQNS. MATNV  5
C                                                                       MATNV  6
C INVERTS -A-, THE N BY N MATRIX OF COEFFICIENTS, SOLVES M SYSTEMS OF   MATNV  7
C N LINEAR EQUATIONS, AND EVALUATES THE DETERMINANT OF -A-.             MATNV  8
C ON ENTRY A CONTAINS THE MATRIX OF COEFFICIENTS AND B CONTAINS THE M   MATNV  9
C VECTORS OF CONSTANTS.  ON EXIT A CONTAINS THE INVERSE OF THE MATRIX   MATNV 10
C OF COEFFICIENTS AND B CONTAINS THE M SOLUTION VECTORS.                MATNV 11
C ON EXIT M CONTAINS (-M MINUS THE RANK OF -A-).                        MATNV 12
C MA IS THE FIRST DIMENSION OF THE ARRAYS IN WHICH THE MATRICES -A-     MATNV 13
C AND -B- ARE STORED.                                                   MATNV 14
C                                                                       MATNV 15
C USES THE GAUSS-JORDAN METHOD WITH COMPLETE (DOUBLE) PIVOTING.         MATNV 16
C                                                                       MATNV 17
C ORIGINALLY A SHARE PROGRAM BY B. GARBOW (ARGONNE NATIONAL LAB).       MATNV 18
C THIS VERSION TREATS THE LARGEST NONSINGULAR SUBMATRIX OF -A-...       MATNV 19
C IF ANY PIVOT IS .LE. PVBAD (.GE.ZERO), THE REMAINDER OF -A- AND       MATNV 20
C B IS ZEROED OUT, -A- IS UNSCRAMBLED, AND MATINV RETURNS.              MATNV 21
C NOTE....  IF PVBAD IS .GT. ZERO, THE MATRIX -A- MUST BE SCALED....    MATNV 22
C                                                                       MATNV 23
      DIMENSION A(MA,N),B(MA,1),INDEX(50,2)                             MATNV 24
      DIMENSION A(15,15),B(15,1),INDEX(15,2)                            MATNV 25
C                                                                       MATNV 26
C INITIALIZE.                                                           MATNV 27
C                                                                       MATNV 28
C                              NINDX IS THE FIRST DIMENSION OF -INDEX-. MATNV 29
      NINDX=15                                                          MATNV 31
      NINDY=50                                                          MATNV 32
      PVBAD=0.                                                          MATNV 33
      ZERO=0.                                                           MATNV 34
      UNITY=1.                                                          MATNV 35
      NVAR=N                                                            MATNV 36
      IF(NVAR-NINDX)3010,3010,3000                                      MATNV 37
3000 IRANK=0                                                            MATNV 38
      DETRM=ZERO                                                        MATNV 39
      GO TO 3310                                                        MATNV 40
3010 NM=M                                                               MATNV 41
      DETRM=UNITY                                                       MATNV 42
      DO 3020 J=1,NVAR                                                  MATNV 43
        DO 3020 JL=1,2                                                  MATNV 44
3020      INDEX(J,JL)=0                                                 MATNV 45
                                                                        MATNV 46
```

```
      C  SEARCH FOR THE NEXT PIVOT ELEMENT.                                  MATNV 47
      C                                                                      MATNV 48
402           I=0                                                            MATNV 49
403           IRANK=0                                                        MATNV 50
404    3030 AMAX=ZERO                                                        MATNV 51
405           DO 3100 J=1,NVAR                                               MATNV 52
406             IF(INDEX(J,1))3100,3040,3100                                 MATNV 53
407    3040     DO 3090 K=1,NVAR                                             MATNV 54
408               IF(INDEX(K,1))3090,3050,3090                               MATNV 55
409    3050       T=A(J,K)                                                   MATNV 56
410               IF(T)3060,3070,3070                                        MATNV 57
411    3060       T=-T                                                       MATNV 58
412    3070       IF(T-AMAX)3090,3090,3080                                   MATNV 59
413    3080       IROW=J                                                     MATNV 60
414               KOLUM=K                                                    MATNV 61
415               AMAX=T                                                     MATNV 62
416    3090       CONTINUE                                                   MATNV 63
417    3100     CONTINUE                                                     MATNV 64
418           IF(AMAX-PIVRAD)3320,3320,3110                                  MATNV 65
419    3110 INDEX(KOLUM,1)=IROW                                              MATNV 66
      C                                                                      MATNV 67
      C  INTERCHANGE ROWS TO PUT THE PIVOT ELEMENT ON THE DIAGONAL           MATNV 68
      C                                                                      MATNV 69
420           IF(IROW-KOLUM)3120,3160,3120                                   MATNV 70
421    3120 DETERM=-DETERM                                                   MATNV 71
422           DO 3130 L=1,NVAR                                               MATNV 72
423             SWAP=A(IROW,L)                                               MATNV 73
424             A(IROW,L)=A(KOLUM,L)                                         MATNV 74
425    3130     A(KOLUM,L)=SWAP                                              MATNV 75
426           IF(NB)3160,3160,3140                                           MATNV 76
427    3140 DO 3150 L=1,NB                                                   MATNV 77
428             SWAP=B(IROW,L)                                               MATNV 78
429             B(IROW,L)=B(KOLUM,L)                                         MATNV 79
430    3150     B(KOLUM,L)=SWAP                                              MATNV 80
431    3160   I=I+1                                                         MATNV 81
432           INDEX(I,2)=KOLUM                                               MATNV 82
433           PIVOT=A(KOLUM,KOLUM)                                           MATNV 83
434           DETERM=PIVOT*DETRM                                             MATNV 84
435           IRANK=IRANK+1                                                  MATNV 85
      C                                                                      MATNV 86
      C  DIVIDE THE PIVOT ROW BY THE PIVOT ELEMENT.                          MATNV 87
      C                                                                      MATNV 88
436           A(KOLUM,KOLUM)=UNITY                                           MATNV 89
437           DO 3170 L=1,NVAR                                               MATNV 90
438    3170     A(KOLUM,L)=A(KOLUM,L)/PIVOT                                  MATNV 91
439           IF(NB)3200,3200,3180                                           MATNV 92
440    3180 DO 3190 L=1,NB                                                   MATNV 93
441    3190     B(KOLUM,L)=B(KOLUM,L)/PIVOT                                  MATNV 94
      C                                                                      MATNV 95
      C  REDUCE THE NON-PIVOT ROWS.                                          MATNV 96
      C                                                                      MATNV 97
442    3200 DO 3250 L1=1,NVAR                                                MATNV 98
443           IF(L1-KOLUM)3210,3250,3210                                     MATNV 99
444    3210     T=A(L1,KOLUM)                                                MATNV100
445             A(L1,KOLUM)=ZERO                                             MATNV101
446             DO 3220 L=1,NVAR                                             MATNV102
447    3220       A(L1,L)=A(L1,L)-A(KOLUM,L)*T                               MATNV103
448             IF(NB)3250,3250,3230                                         MATNV104
449    3230     DO 3240 L=1,NB                                               MATNV105
450    3240       B(L1,L)=B(L1,L)-B(KOLUM,L)*T                               MATNV106
```

```
451     3250    CONTINUE                                                MATNV107
452             IF(I-NVAR) 3030,3260,3260                               MATNV108
        C                                                               MATNV109
        C   INTERCHANGE COLUMNS.                                        MATNV110
        C                                                               MATNV111
453     3260 KOLUM=INDEX(I,2)                                           MATNV112
454             IROW=INDEX(KOLUM,1)                                     MATNV113
455             IF(IROW-KOLUM) 3270,3290,3270                           MATNV114
456     3270 DO 3280 K=1,NVAR                                           MATNV115
457             SWAP=A(K,IROW)                                          MATNV116
458             A(K,IROW)=A(K,KOLUM)                                    MATNV117
459     3280    A(K,KOLUM)=SWAP                                         MATNV118
460     3290 I=I-1                                                      MATNV119
461     3300 IF(I)3310,3310,3240                                        MATNV120
462     3310 M=NVAR-IPANK                                               MATNV121
463             DET=DETRM                                               MATNV122
464             RETURN                                                  MATNV123
        C                                                               MATNV124
        C   ZERO OUT THE REST OF A AND B                               MATNV125
        C                                                               MATNV126
465     3320 DO 3370 J=1,NVAR                                           MATNV127
466             DO 3330 K=1,NVAR                                        MATNV128
467               IF(INDEX(K,2)-J)3330,3370,3330                        MATNV129
468     3330      CONTINUE                                              MATNV130
469             DO 3340 JL=1,NVAR                                       MATNV131
470               A(JL,J)=ZERO                                          MATNV132
471     3340      A(J,JL)=ZERO                                          MATNV133
472             IF(M)3370,3370,3350                                     MATNV134
473     3350    DO 3360 JL=1,M                                          MATNV135
474     3360      B(J,JL)=ZERO                                          MATNV136
475     3370    CONTINUE                                                MATNV137
476             GO TO 3330                                              MATNV138
        C                                                               MATNV139
        C   END MATNV.                                                  MATNV140
477             END                                                     MATNV141

478             BLOCK DATA
479             IMPLICIT REAL*8 (A-H,O-Z)
480             COMMON/LSTPAR/XSTAR(10)
481             DATA XSTAR/1.0D0,1.0D0,8*0.0D0/
482             END

        $ENTRY
```

VITA 2

Guang-Nay Wang

Candidate for the Degree of

Doctor of Education

Thesis: COMPUTATIONAL EXPERIMENTS WITH SYSTEMS OF NONLINEAR EQUATIONS

Major Field: Higher Education

Biographical:

Personal Data: Born in China, September 19, 1945, the son of Peng Wang and Ching-Ru Yu; married to Yu-Shien Chu, on November 11, 1972.

Education: Graduated from Taiwan Provincial Keelung High School, Keelung, Taiwan, Republic of China, in July, 1962; received the Bachelor of Science degree with a major in Mathematics from National Taiwan Normal University, Taipei, Taiwan, Republic of China, in 1968; received the Master of Arts degree in Mathematics from Northeast Missouri State University, Kirksville, Missouri, in May, 1974; completed requirements for the Doctor of Education degree at Oklahoma State University, Stillwater, Oklahoma, in December, 1978.

Professional Experience: Mathematics teacher at Taiwan Provincial Keelung High School, August, 1967, to July, 1973; graduate assistant in the Mathematics Department at Northeast Missouri State University, August, 1973, to May, 1974; graduate assistant in the Mathematics Department at Oklahoma State University, August, 1974, to May, 1978.

Professional Organizations: The Mathematical Association of America.