AN ABSTRACT OF THE THESIS OF

ANGELINA SIU-HUNG SEN for the MASTER OF SCIENCE
(Name) (Degree)

in MATHEMATICS presented on August 10, 1972
(Major) (Date)

Title: A METHOD OF SOLVING NONLINEAR SIMULTANEOUS EQUATIONS WITHOUT USING JACOBIAN

Abstract approved: Redacted for Privacy
Joel Davis

In this thesis some methods for solving systems of nonlinear equations are described, which do not require calculation of the Jacobian matrix. One of these methods is programmed to solve a parametrized system with possible singularities. The efficiency of this method and a modified Newton's method are compared using experimental results from six test cases.

A METHOD OF SOLVING NONLINEAR SIMULTANEOUS EQUATIONS WITHOUT USING JACOBIAN

by

ANGELINA SIU-HUNG SEN

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

MASTER OF SCIENCE

June 1973

APPROVED:

Redacted for Privacy

Associate Professor of Mathematics

In Charge of Major

Redacted for Privacy

Acting Chairman of Department of Mathematics

Redacted for Privacy

Dean of Graduate School

Date thesis is presented August 10, 1972

Typed by Jolan Erőss for Angelina Siu-Hung Sen

Acknowledgement

I would like to express my thanks and appreciation to my major professor, Dr. Joel Davis, for suggesting the topic and for his time and guidance throughout the preparation of this thesis.

I would also thank Mr. Paul Cullop for sharing his experience with the change of parameter method and suggestions.

TABLE OF CONTENTS

A METHOD OF SOLVING NONLINEAR SIMULTANEOUS EQUATIONS WITHOUT USING JACOBIAN

CHAPTER I

INTRODUCTION

Two of the important drawbacks of using Newton's method to solve the equation

$$\bar{f}(\bar{x}) = \bar{0} \qquad (1.1)$$

where $\bar{x}$ is an n-dimensional vector, and $\bar{f}$ is a column vector of functions f_i, $i = 1,2,\cdots, n$ are

(1) The iterations may not converge if the initial estimate is not sufficiently close to the solution, and a "good" first approximation is not always available.

(2) The Newton's method calls for the evaluation of the Jacobian at each step. For complicated functions it may be difficult to obtain the derivatives analytically, and even when this is possible, the labor involved is usually so huge that made the method undesirable or even impracticable.

In the first part of this thesis we describe in detail some of the methods that do not require the Jacobian matrix. In the second part, one of these methods is described in more detail. To ensure that good initial guesses are available, we insert a parameter into the equation. When the Jacobian of the parameterized system becomes nearly singular, we do an automatic change of parameter to try to get by the singular point. This method is compared with the modified Newton's method for six test cases.

CHAPTER II

Quasi-Newton Methods

The algorithm using Newton's method to solve equation (1.1), $\bar{f}(\bar{x}) = \bar{0}$ is defined by

$$\bar{x}^{(i+1)} = \bar{x}^{(i)} - (J^{(i)})^{-1} \bar{f}^{(i)} \quad \text{if } \exists\, J^{(i)^{-1}}, \qquad (2.1)$$

where $\bar{x}^{(0)}$ is the initial approximation and $\bar{x}^{(i)}$ the $i^{\underline{th}}$ approximation to the solution of (1.1). $J^{(i)}$ being the Jacobian matrix of $\bar{f}$ evaluated at $\bar{x}^{(i)}$ and $\bar{f}^{(i)} = \bar{f}(\bar{x}^{(i)})$. In practice, instead of using (2.1), one may solve the equivalent system, $J^{(i)}(\bar{x}^{(i+1)} - \bar{x}^{(i)}) = -\bar{f}^{(i)}$, for the solution of a system takes fewer operations than inverting a matrix. At each iteration of the Newton's method, the Jacobian must be computed. For complicated functions, the partial derivatives are usually even more complicated than the function itself. In order to reduce the amount of computation, one may modify the Newton's method by either calculating the Jacobian once exactly, inverting it at the first iteration, and use the same matrix throughout all iterates, or calculate the inverted Jacobian every k iterations, where k is a small positive integer greater than one. If the first alternative is used, the convergence rate is not as good as using (2.1), but only one inverse is needed [6]. The second alternative is a compromise between the two.

There are other variations of the Newton's method, in which no exact Jacobian is required. Only some approximation to the Jacobian (or that of the inverse of the Jacobian) matrix is needed, and this matrix is being modified throughout the process. Methods of this type are often referred to as quasi-Newton methods. The advantage of the quasi-Newton methods is that no exact Jacobian is required, and in some particular methods, the modification of the iteration matrix does not require any more evaluations of the function $\bar{f}$ than would have been required if the iteration matrix were held constant. It is then reasonable that these methods will be economical in terms of function evaluations; but since the Jacobian matrix is only approximate, it is also expected that the total number of iterations would be more than that of the Newton's method.

The Secant Method

The secant method for simultaneous non-linear equations is a generalization of the secant method for a single function of one variable. Solving equation (1.1) by this method, n+1 trial guesses $\bar{x}^{(1)}, \bar{x}^{(2)}, \cdots, \bar{x}^{(n+1)}$ must be given and the next guess $\bar{x}^*$ is formed from these previous n+1 points. The idea of this method is outlined by Wolfe [12]. Given the n+1 guesses, we find n+1 scalars, $c_1, c_2, \cdots, c_{n+1}$, so that

$$\text{(a)}\qquad \sum_{j=1}^{n+1} c_j = 1$$

$$\text{(b)}\qquad \sum_{j=1}^{n+1} c_j \bar{f}(\bar{x}^{(j)}) = \bar{0}$$

are satisfied. Then the next guess $\bar{x}^*$ is defined by

$$\bar{x}^* = \sum_{j=1}^{n+1} c_j \bar{x}^{(j)} .$$

This process is continued, replacing $\bar{x}^{(j)}$ by $\bar{x}^*$, where $\|\bar{f}(\bar{x}^{(j)})\|$ is the maximum over the j's, j=1, 2, ⋯, n+1, for some norm, say the Euclidean norm. The process stops when some convergence criterion is reached. This method is based on the following fact.

If (1) $\bar{f}(\bar{x}) = \bar{a} + B\bar{x}$, $\bar{a}$ is an n-dimensional vector, B is an n×n matrix

$$\text{(2)}\qquad \sum_{j=1}^{n+1} c_j = 1$$

$$\text{(3)}\qquad \sum_{j=1}^{n+1} c_j \bar{f}(\bar{x}^{(j)}) = \bar{0}$$

then $\bar{x}^* = \sum_{j=1}^{n+1} c_j\, \bar{x}^{(j)}$ is a root of $\bar{f}(\bar{x})$. So if $\bar{f}(\bar{x})$ is approximately equal to its two term Taylor series expansion around some point close to the root, and the c_j's, j=1, ⋯, n+1 are as described in (a), (b) above, then the $\bar{x}^*$ generated is approximately a root of $\bar{f}(\bar{x})$.

The main computational effort in this method is spent on finding the c's, which are obtained by solving a system of $n+1$ linear equations. A computation scheme developed by Wolfe indicated this set of $n+1$ linear equations are changed only partially from step to step, so after the initial step, the work involved is actually less than solving a general linear system. We will give the algorithm below.

Let $\bar{c} = (c_1, c_2, \cdots, c_n, c_{n+1})^T$

$\bar{y} = (0, 0, \cdots, 0, 1)^T.$

(1) Obtain $n+1$ guesses, $\bar{x}^{(1)}, \bar{x}^{(2)}, \cdots, \bar{x}^{(n+1)}$, and evaluate the function $\bar{f}$ at these points.

(2) Form an $n+1$ by $n+1$ matrix A whose elements are given by

$$A_{ij} = f_i(\bar{x}^{(j)}),\ i=1,2,\cdots,n \text{ and } j=1,2,\cdots, n+1$$

$$A_{ij} = 1 \qquad i=n+1, \text{ and } j=1,2,\cdots, n+1$$

(3) Solve $A\bar{c} = \bar{y}$ for $\bar{c}$, or equivalently,

$$\bar{c} = A^{-1}\bar{y}.$$

(4) $$\bar{x}^* = \sum_{j=1}^{n+1} c_j\, \bar{x}^{(j)}$$

(5) Test for convergence

(6) Find j so that $\|\bar{f}(\bar{x}^{(j)})\|_2 = \max_{k=1,2\cdots,n+1} \|\bar{f}(\bar{x}^{(k)})\|_2.$

(7) Replace $\bar{x}^{(j)}$ by $\bar{x}^*$ and evaluate $\bar{f}(\bar{x}^{(j)})$.

(8) Let $\bar{p} = (f_1(\bar{x}^{(j)}), f_2(\bar{x}^{(j)}), \cdots, f_n(\bar{x}^{(j)}), 1)^T$

and $\bar{q} = A^{-1}\,\bar{p}$

Form the new matrix A^* by replacing the $j^{\underline{th}}$ column of A with $\bar{p}$, then $(A^*)^{-1}$ is given by

$$(A^*)^{-1}_{jk} = (A^{-1})_{jk}/q_j, \quad \text{for } k=1, \cdots, n+1$$

$$(A^*)^{-1}_{\ell k} = (A^{-1})_{\ell k} - (A^{-1})_{jk}(q_\ell/q_j), \quad \ell \neq j \quad \text{and } k=1, \cdots, n+1$$

(9) Replace A by A^*, go to (3).

This method was tested by Wolfe on a number of problems where $n=2$, and his experimental results showed that the convergence is of order $\frac{1}{2}(\sqrt{5}+1)$, which is the same as the secant method when $n=1$, as shown in Jeeves [9]. No further comparisons or tests for n greater than two were made.

Barnes' Generalized Secant Method [1]

Algorithm.

Let $\bar{x}^{(i)}$ be the $i^{\underline{th}}$ approximation to the solution of $\bar{f}(\bar{x}) = \bar{0}$, and $B^{(i)}$ the Jacobian at $\bar{x}^{(i)}$, respectively. As before, $\bar{f}^{(i)} = \bar{f}(\bar{x}^{(i)})$, and $\bar{x}^{(1)}$ is the initial guess to a root of $\bar{f}(\bar{x})$. $\delta\bar{x}^{(i)}$ is defined by

$$B^{(i)}\delta\bar{x}^{(i)} = -\bar{f}^{(i)} \qquad (2.3)$$

and

$$\bar{x}^{(i+1)} = \bar{x}^{(i)} + \delta\bar{x}^{(i)} \qquad (2.4)$$

Then the correction matrix $D^{(i)}$ to be applied to $B^{(i)}$ is chosen so that the corrected Jacobian $B^{(i+1)}=B^{(i)}+D^{(i)}$ satisfies the equation $\bar{f}^{(i+1)}=\bar{f}^{(i)}+J\delta\bar{x}^{(i)}$. $D^{(i)}$ is given by

$$D^{(i)} = \frac{f^{(i+1)}(\bar{z}^{(i)})^T}{(\bar{z}^{(i)})^T\delta\bar{x}^{(i)}} \tag{2.5}$$

where $(\bar{z}^{(i)})^T$ is the transpose of $\bar{z}^{(i)}$. $\bar{z}^{(i)}$ is obtained by the following rule.

(a) $\bar{z}^{(1)}$ is chosen arbitrarily. For simplicity, it is chosen to be $\dfrac{\delta\bar{x}^{(1)}}{\|\delta\bar{x}^{(1)}\|_2}$, as to ensure $(\bar{z}^{(1)})^T\delta\bar{x}^{(1)} \neq 0$

(b) If $i \geq n$, $\bar{z}^{(i)}$ is chosen orthogonal to the previous n-1 steps, $\delta\bar{x}^{(i-n+1)}, \cdots, \delta\bar{x}^{(i-1)}$.

(c) If $i < n$, then $\bar{z}^{(i)}$ is chosen to be orthogonal to the previous (i-1) steps, $\delta\bar{x}^{(1)}, \cdots, \delta\bar{x}^{(i-1)}$ only. In this case, $\bar{z}^{(i)}$ is not uniquely determined. It is suggested that we take $\bar{z}^{(i)}$ to be the linear combination of $\delta\bar{x}^{(1)}, \cdots, \delta\bar{x}^{(i)}$, which is also orthogonal to $\delta\bar{x}^{(1)}, \cdots, \delta\bar{x}^{(i-1)}$. The magnitude of $\bar{z}^{(i)}$ is arbitrarily chosen to be one. Barnes has a detailed discussion in finding the vector $\bar{z}$. This is done essentially by the Gram-Schmidt process.

In this method, no explicit evaluation of the Jacobian is required, and at each step, a linear system is solved to obtain $\delta\bar{x}^{(i)}$, then the Jacobian matrix is corrected. Some work is involved in finding $\bar{z}$ at each step, but no more function evaluation is required in this method than if B were held constant. Barnes proved that this method will converge within n iterations, if $\bar{f}$ is linear.

A Class of Quasi-Newton Methods

Broyden [2] discussed a class of methods not using derivatives in 1965. We will first describe the general set up of the method, and then go into some specific methods, finally summarize Broyden's experimental results on the efficiency of various methods.

The class of methods.

Let $\bar{x}^{(i)}$ be the $i^{\underline{th}}$ approximation to the solution of (2.1). Define $\bar{p}^{(i)}$ by

$$\bar{p}^{(i)} = -(B^{(i)})^{-1}\bar{f}^{(i)} \qquad (2.6)$$

where $B^{(i)}$ is some approximation to the Jacobian at $\bar{x}^{(i)}$. The next approximation to the solution is given by

$$\bar{x}^{(i+1)} = \bar{x}^{(i)} + t^{(i)}\,\bar{p}^{(i)} \qquad (2.7)$$

where $t^{(i)}$ is a scalar chosen to prevent the process from diverging. When $t^{(i)} = 1$ and $B^{(i)} = J^{(i)}$, this is just Newton's iteration. The heart of this class of methods lies in the improvement of the iteration matrix

$B^{(i)}$. Broyden imposes some conditions that the improved matrix $B^{(i+1)}$ must satisfy. Consider the following.

Define $\bar{x} = \bar{x}^{(i)} + t\,\bar{p}^{(i)}$, (2.8)

then for each f_j, $j = 1,2,\cdots, n$, f_j is a function of the single variable t. Taking the derivatives of $\bar{x}$ in (2.8) with respect to t, we get

$$\frac{d\bar{x}}{dt} = \bar{p}^{(i)} \tag{2.9}$$

Since $\frac{\partial f_j}{\partial x_k}$ exists for each $j, k=1, 2,\cdots, n$, we differentiate f_j with respect to t, then

$$\frac{df_j}{dt} = \sum_{k=1}^{n} \frac{\partial f_j}{\partial x_k} \cdot \frac{dx_k}{dt} \qquad j=1,2,\cdots,n \tag{2.10}$$

Combining (2.9) and (2.10), using matrix notations, we get

$$\frac{d\bar{f}}{dt} = J\,\bar{p}^{(i)} \tag{2.11}$$

J being the exact Jacobian at $\bar{x}$.

Broyden contends that any approximation to J should satisfy equation (2.11). Assuming the derivatives are laborious to calculate, we will approximate $\frac{d\bar{f}}{dt}$ by differences. Considering each f_j as a function of t alone, and expanding it in a two-term Taylor series yields

$$\bar{f}(t^{(i)} - s) \cong \bar{f}^{(i+1)} - s\,\frac{d\bar{f}}{dt} \tag{2.12}$$

Combine equation 2.12 with 2.11. We have

$$\bar{f}(t^{(i)}-s) - \bar{f}^{(i+1)} \cong s\ J\ \bar{p}^{(i)} \qquad (2.13)$$

We w~nt to find $B^{(i+1)}$ so that the following equation is satisfied.

$$\bar{f}^{(i+1)}-\bar{f}(t^{(i)}-s^{(i)}) = s^{(i)}B^{(i+1)}\bar{p}^{(i)} \qquad (2.14)$$

The choice of $t^{(i)}$ and $s^{(i)}$ will be discussed later in this chapter. Before we summarize the above general procedure, we observe that if the inverse of an approximate Jacobian is used instead of the approximate Jacobian in equation (2.6), then we can reduce the operation of solving a system of linear equations to a matrix-vector multiplication operation at each step. So we define

$$H^{(i)} = (B^{(i)})^{-1} \qquad (2.15)$$

and use $H^{(i)}$ as the iteration matrix in the following algorithm.

Algorithm.

(a) Let $i = 0$, obtain an initial approximate solution $\bar{x}^{(i)}$ and an approximate Jacobian $B^{(i)}$, invert $B^{(i)}$, set $H^{(i)}= (B^{(i)})^{-1}$; calculate $\bar{f}^{(i)}=\bar{f}(\bar{x}^{(i)})$.

(b) Let $\bar{p}^{(i)} = -H^{(i)}\bar{f}^{(i)}$

(c) $\bar{x}^{(i+1)} = \bar{x}^{(i)}+ t^{(i)}\bar{p}^{(i)}$

(d) Evaluate $\bar{f}^{(i+1)} = \bar{f}(\bar{x}^{(i+1)})$

(e) Test for convergence

(f) set $\bar{y}^{(i)} = \bar{f}^{(i+1)} - \bar{f}^{(i)}$

(g) Find $H^{(i+1)}$ so that it satisfies

$$H^{(i+1)}\bar{y}^{(i)} = s^{(i)}\bar{p}^{(i)}$$

(h) $i = i+1$, go to (b).

In step (g), the matrix $H^{(i+1)}$ is not uniquely determined. With different choices of $H^{(i+1)}$ that satisfy (g) and various $t^{(i)}$ and $s^{(i)}$, we can derive a whole class of methods without derivatives.

Particular methods.

Method 1.

$B^{(i+1)}$ is chosen so that the change in $\bar{f}$ predicted by $B^{(i+1)}$ in a direction $\bar{q}^{(i)}$, orthogonal to $\bar{p}^{(i)}$ is the same as would be predicted by $B^{(i)}$, that is

$$B^{(i+1)}\bar{q}^{(i)} = B^{(i)}\bar{q}^{(i)}, \quad \text{where} \quad (\bar{q}^{(i)})^T\bar{p}^{(i)} = 0 \tag{2.16}$$

It is easy to verify that $B^{(i+1)}$, given by

$$B^{(i+1)} = B^{(i)} + \frac{(\bar{y}^{(i)} - s^{(i)}B^{(i)}\bar{p}^{(i)})\bar{p}^{(i)T}}{s^{(i)}(\bar{p}^{(i)})^T\bar{p}^{(i)}} \tag{2.17}$$

satisfies (2.16) and the equation $\bar{y}^{(i)} = s^{(i)}B^{(i+1)}p^{(i)}$. Since we chose to use H as the iteration matrix, we have to correct H rather than B, and formula 2.17 cannot readily be used. If we apply Householder's formula [8], which says for a non-singular matrix A and vectors $\bar{x},\bar{y}$, all of order n, and if $(A+\overline{xy}^T)$ is non-singular, then

$$(A + \bar{x}\,\bar{y}^T)^{-1} = A^{-1} - \frac{A^{-1}\bar{x}\,\bar{y}^T A^{-1}}{1+\bar{y}^T A^{-1}\bar{x}} \tag{2.18}$$

Now letting $B^{(i)} = A$, $\frac{\bar{y}^{(i)} - s^{(i)}B^{(i)}\bar{p}^{(i)}}{s^{(i)}(\bar{p}^{(i)})^T(\bar{p}^{(i)})}$ to be $\bar{x}$, and $\bar{p}^{(i)}$ to be $\bar{y}$, we can get $H^{(i+1)}$ from $H^{(i)}$ with very little computation.

$$H^{(i+1)} = H^{(i)} - \frac{(H^{(i)}\bar{y}^{(i)} - s^{(i)}\bar{p}^{(i)})\bar{p}^{(i)^T} H^{(i)}}{(\bar{p}^{(i)})^T H^{(i)}\bar{y}^{(i)}} \tag{2.19}$$

Method 2.

Instead of requiring equation 2.16 to be true, here we require

$$H^{(i+1)}\bar{v}^{(i)} = H^{(i)}\bar{v}^{(i)}, \quad \text{with} \quad \bar{v}^{(i)^T}\bar{y}^{(i)} = 0 \tag{2.20}$$

Since $H^{(i+1)}$ must also satisfy $H^{(i+1)}\bar{y}^{(i)} = -s^{(i)}\bar{p}^{(i)}$, we can see that $H^{(i+1)}$ is uniquely defined by

$$H^{(i+1)} = H^{(i)} - \frac{(H^{(i)}\bar{y}^{(i)} - s^{(i)}\bar{p}^{(i)})y^{(i)^T}}{(\bar{y}^{(i)})^T\bar{y}^{(i)}} \tag{2.21}$$

But Broyden [2] found that this method is unsatisfactory in practice, so it will not be further discussed.

The choise of $t^{(i)}$ and $s^{(i)}$.

As mentioned earlier, $t^{(i)}$ is chosen to prevent the process from diverging. Broyden forced the Euclidean norm of $\bar{f}^{(i)}$ to be a non-increasing function of i. By doing so, even though convergence is not guaranteed, divergence, however, can be prevented. One may choose $t^{(i)}$ to minimize the norm of $\bar{f}^{(i+1)}$ or just take any $t^{(i)}$ so that $\|\bar{f}^{(i+1)}\| \leq \|\bar{f}^{(i)}\|$.

Perhaps minimization of the norm of $\bar{f}^{(i+1)}$ will give the greatest immediate improvement to the approximation, but in order to find $t^{(i)}$ which minimizes the norm of $\bar{f}^{(i+1)}$, the vector function $\bar{f}$ needs to be evaluated a number of times. In fact, Broyden found that in order to minimize the norm of $\bar{f}^{(i+1)}$, four to six function evaluations were needed, compared with only one to three evaluations in the norm reduction case. Unless the number of iterations in the norm minimization case is considerably less than the norm reduction case, it would take even more function evaluations than the latter. In later sections, we will see that the norm minimization is indeed a poor strategy compared with the norm reduction method. If one simply chooses the Newtonian value of unity for $t^{(i)}$, it would require the least work in correcting the iteration matrix H, but this does not always give a norm reduction in $\bar{f}^{(i)}$. With a poor initial estimate of the solution, the Newton's method frequently fails to converge. Taking $t^{(i)}$ equal to one may be a good strategy if a good initial estimate of the solution is available. We will leave the discussion of how to get a good initial estimate for Chapter III. Since the precise method of norm reduction or minimization is not of main interest here, we will refer the actual procedure of finding $t^{(i)}$ to section seven and Appendix II in Broyden's paper [2].

The correction on B^i or H^i in the above method depends on the derivative of $\bar{f}$ with respect to t, (equation 2.11), and the value of $\frac{d\bar{f}}{dt}$ is approximated by the finite differences $[\bar{f}(t^{(i)}-s^{(i)})-\bar{f}(t^{(i+1)})]s^{(i)}$, (equation 2.12). We should choose $s^{(i)}$ so that $\frac{d\bar{f}}{dt}$ is being approximated as close as possible. Furthermore, it should be done with no extra function evaluations other than those necessary for the norm reduction. In the case where $t^{(i)}$ is always one, the only possible choice of $s^{(i)}$ is $t^{(i)}$. However, if in the process of norm reduction or minimization more than one function is being evaluated, then there may be several choices of $s^{(i)}$. Suppose the norm of $\bar{f}$ is evaluated successively at $t = t_k$, $k = 1,2,\cdots, \ell$, where $t_1 = 1$, and $t_\ell = t^{(i)}$. Furthermore, $t^{(i)}$ is the value of t that reduces (or minimizes) the norm of $\bar{f}^{(i)}$, then we choose $s^{(i)}=t_\ell-t_{\ell-1}$ (2.22). We will not go into a detailed discussion on how the $s^{(i)}$ is chosen here; this topic is referred to Broyden [2], section six.

In the subsequent comparison of methods, if $s^{(i)}$

is chosen to be equal to $t^{(i)}$, then the term "full step" is used, if $s^{(i)}$ is given by equation (2.22), it is called "incremental".

Some experimental results and comparison.

Broyden's comparison of the various methods are based on what he defined as Mean Convergence Rate, R, defined by

$$R = \frac{1}{m} \ln \frac{N_1}{N_m} , \qquad (2.23)$$

where m is the total number of function evaluations and N_1, N_m are the initial and final Euclidean norms of $\bar{f}$. Method 1, using norm minimization, norm reduction, full step, and incremental are compared with the constant matrix method and the basic method. In the constant matrix method, the Jacobian is calculated first by finite differences and then inverted; this same matrix $H^{(0)}$ is used throughout all iterations. In the basic method, at each iterate, $H^{(i)}$ is recalculated the same way $H^{(0)}$ is calculated.

In Broyden's first four test equations, method 1 with norm reduction, full step variation and the basic method, (both the norm reduction and minimization) did consistently well, but the norm minimization variants of method 1 tended to be slow. The incremental method seemed to be either very efficient or very slow. Broyden suggested that probably a better method of choosing s could make this method more competitive, but no more experiments were done regarding this respect. The constant matrix did reasonably well if a good initial estimate to both the Jacobian and $\bar{x}$ at the solution were available, otherwise, it tended to have a low mean convergence rate. In no case was it better than method 1 with full step reduction. Based on the results from the first four test equations, Broyden tested six more equations, comparing the basic method with method 1, full step reduction only. He concluded that norm reduction is a better strategy than norm minimization. He also tentatively concluded that if a reasonably good initial estimate of the solution is available, method 1 with full step norm reduction is superior to the basic method; otherwise the basic method is better. But he also mentioned in all of his test cases, method 1, with full step reduction has never failed to converge when the basic reduction converges. The former may be a very good alternative to the basic method, especially when a good initial estimate of $H^{(0)}$ is available.

In a later paper [3], Broyden found that norm reduction is not necessary, or even inhibits convergence, if a good initial estimate is available. Throughout the preceeding discussion, it is clear that a good initial estimate is imperative for rapid convergence, and we will discuss how it can be done with the use of a parameter in Chapter III. In Chapter IV, we describe a method using Broyden's method 1 with $s^{(i)}=t^{(i)}$, always equal to unity, and with the use of a parameter to solve a system of nonlinear equations. We conclude this chapter with a function minimization method which may be used in solving a system of equations.

Least Square Minimization Method

Any method which minimizes a sum of squares of nonlinear functions can be applied to solve equation (1.1) in the following way. Let $F(\overline{x}) = \sum_{k=1}^{n} [f_k(\overline{x})]^2 = \|\overline{f}(\overline{x})\|_2^2$, or in matrix notation:

$$\overline{f}(\overline{x}) = \overline{f}(\overline{x})^T\overline{f}(\overline{x}) \qquad (2.24)$$

where T denotes transpose. Suppose a minimum of F is found and at the minimum $F(\overline{x}) = 0$, since F is the sum of n non negative terms, the $\overline{x}$ that minimizes F also solves the system $\overline{f}(\overline{x}) = \overline{0}$.

In this section we first describe the generalized least square method and then introduce Powell's modification [11] which does not require any derivatives. We differentiate (2.24) with respect to $\overline{x}$,

$$F'(\overline{x}) = 2\overline{f}(\overline{x})^T\overline{f}'(\overline{x}) \qquad (2.25)$$

Let $\overline{x}^*$ be $i^{\underline{th}}$ approximation to the position of the minimum $\bar{z}$, and define a vector $\bar{\delta}$ so that $\bar{x}^*+\bar{\delta} = \bar{z}$; then the derivatives in equation (2.25) evaluated at $\bar{z}$ equal to zero.

$$\bar{f}(\bar{x}^*+\bar{\delta})^T\bar{f}'(\bar{x}^*+\bar{\delta}) = \bar{0}^T \qquad (2.26)$$

Expanding each term in the above equation by a two-term Taylor series about $\bar{x}^*$, we obtain

$$(\bar{f}(\bar{x}^*)+\bar{f}'(\bar{x}^*)\bar{\delta})^T(\bar{f}'(\bar{x}^*)+\bar{f}''(\bar{x}^*)\bar{\delta}) \cong \bar{0}^T \qquad (2.27)$$

If we assume the term $\bar{f}''(\bar{x}^*)\bar{\delta}$ is small compared with $\bar{f}'(\bar{x}^*)$, we have

$$\bar{f}(\bar{x}^*)^T\bar{f}'(\bar{x}^*) + \bar{\delta}^T\bar{f}'(\bar{x}^*)^T\bar{f}'(\bar{x}^*) \cong \bar{0}^T \qquad (2.28)$$

Using matrix notation, letting J denote $\bar{f}'(\bar{x}^*)$, rewrite (2.28)

$$\bar{f}(\bar{x}^*)^TJ + \bar{\delta}^TJ^TJ \cong \bar{0}^T$$

The generalized least square method is defined by

$$\bar{\delta}^TJ^TJ = -\bar{f}(\bar{x}^*)^TJ \qquad (2.29)$$

Solving (2.29) for $\bar{\delta}$ and $\bar{x}^{(i+1)}$ is defined to be $\bar{x}^* + \lambda_m\bar{\delta}$, where λ_m is a scalar minimizes the function $F(\bar{x}^*+\lambda\bar{\delta})$. A procedure described by Powell [10] can be used to find the minimum of F along the line $\bar{x}^* + \lambda\bar{\delta}$.

Powell's iterative procedure not using derivatives essentially approximates the derivatives by finite differences, but after the initial iteration, all subsequent calculations of derivatives by differences actually use function values already evaluated during the course of minimization, hence no substantial extra computation is needed. In this iterative scheme, an estimate $\bar{x}^*$ to the position of the minimum $\bar{z}$; n linearly independent vectors $\bar{d}^1, \bar{d}^2, \cdots, \bar{d}^n$ and an estimate of the derivatives are required. We define $\bar{\gamma}^{(i)}$ to be a finite difference approximation to the derivative of $\bar{f}$ in the direction $\bar{d}^{(i)}$, for $i = 1, 2,\cdots, n$, that is

$$\bar{\gamma}^{(i)} \cong J\ \bar{d}^{(i)} \qquad i = 1,2,\cdots,n. \quad (2.30)$$

The magnitude of the $\bar{d}^{(i)}$'s are chosen so that each of the vectors $\bar{\gamma}^{(i)}$ satisfies the following

$$\|\bar{\gamma}^{(i)}\|_2^2 = 1 \qquad i = 1,2,\cdots,n \quad (2.31)$$

At the initial iteration, $\bar{d}^{(1)}, \cdots, \bar{d}^{(n)}$ are chosen to be coordinate directions, $\bar{d}^{(i)} = (0,0,\cdots,S_i,0,\cdots,0)^T$, where the S_i's are chosen so that each $\bar{\gamma}^{(i)}$ satisfies (2.31).

The $\bar{\gamma}$'s are found by the following equation

$$\bar{\gamma}^{(i)} = S_i \left[\frac{\bar{f}(x_1^*,\cdots,x_i^*+\varepsilon_i,\cdots,x_n^*)-\bar{f}(x^*)}{\varepsilon_i}\right]$$

$$i = 1,2,\cdots,n \quad (2.32)$$

Where all ε_i, $i=1,2,\cdots,n$ are scalars, chosen to yield reasonable estimates to the derivatives. Now define a vector $\bar{\delta}$ by

$$\bar{\delta} = \sum_{i=1}^{n} q_i \bar{d}^{(i)} \qquad (2.33)$$

The scalars q_i, $i=1,2,\cdots,n$ are chosen so that $\bar{\delta}$ satisfies equation (2.29). Let R denote the matrix whose column vectors are $\bar{\gamma}^{(i)}$, and the vector $\bar{q}=(q_1,\cdots,q_n)^T$. Combining equations (2.29), (2.33) and assuming the approximation of equation (2.30) is a good one, we solve the following for $\bar{q}$.

$$\bar{q}^T R^T R = -\bar{f}(\bar{x}*)^T R \qquad (2.34)$$

Then at the next iteration, $\bar{x}^*$ is replaced by $\bar{x}* + \lambda_m \bar{\delta}$, where λ_m is a scalar which minimizes $F(\bar{x}* + \lambda\bar{\delta})$. Before commencing the next iteration, one of the directions $\bar{d}^{(j)}$ is replaced by $\bar{\delta}$; and the derivatives of $\bar{f}$ along $\bar{\delta}$ should be calculated. This $\bar{d}^{(j)}$ to be replaced by $\bar{\delta}$ is the one that maximizes the following expression

$$\left| \sum_{k=1}^{n} [\gamma_k^{(i)} \; f_k(\bar{x}*)] q_i \right| \qquad i = 1,2,\cdots,n$$

after j is determined, the vector $\bar{\gamma}^{(j)}$ should be replaced by the derivative of $\bar{f}$ along $\bar{\delta}$. To calculate this new $\bar{\gamma}^{(j)}$, we can make use of the function values already obtained in finding λ_m. If for $k=1,2,\cdots,m$, $F(\bar{x}*+\lambda_k\bar{\delta})$ is calculated, we choose λ_ℓ, λ_j which yield the lowest and the next lowest values of $F(\bar{x}* + \lambda\bar{\delta})$.

Approximate $\frac{\partial \bar{f}}{\partial \lambda}(\bar{x}^*+\lambda\bar{\delta})$ by $\dfrac{\bar{f}(\bar{x}^*+\lambda_\ell\bar{\delta})-\bar{f}(\bar{x}^*+\lambda_j\bar{\delta})}{(\lambda_\ell-\lambda_j)}$

and denote the last quantity by the vector $\bar{u}$. We have

$$\bar{u} = \frac{\bar{f}(\bar{x}^*+\lambda_\ell\bar{\delta}) - \bar{f}(\bar{x}^*+\lambda_j\bar{\delta})}{\lambda_\ell - \lambda_j} \ . \qquad (2.35)$$

Further improvement of $\bar{\gamma}^{(j)}$, which approximates $J\ \bar{d}^{(j)}$, is obtained from the following.

$$\bar{\gamma}^{(j)} = \bar{u} - \left(\frac{\bar{u}\cdot\bar{f}(\bar{x}^*+\lambda_m\bar{\delta})}{\|\bar{f}(\bar{x}^*+\lambda_m\bar{\delta})\|_2^2}\right) \bar{f}(x^*+\lambda_m\bar{\delta}) \qquad (2.36)$$

If the square of the Euclidean norm of $\bar{\gamma}^{(j)}$ as defined in (2.36) is not one, it should be scaled to be so. Using the $\bar{\gamma}^{(j)}$ defined in equation (2.36) and replacing $\bar{x}^*$ by $\bar{x}^*+\lambda_m\bar{\delta}$, $\bar{d}^{(j)}$ by $\bar{\delta}$, the next iteration can now begin.

It is found by Powell that this method which requires no derivatives has the same convergence properties as the generalized least square method, and that it can be very effective even if the individual functions in (2.24) do not tend to zero at the minimum.

CHAPTER III

Construction of Parametrized Equations

The method of using a parameter to generate good initial guesses in solving equation (1.1) is originally due to Davidenko, and investigated by some others [7, 6, 3]. The idea is the following: choose a parameter belonging to a real interval I. (γ can be chosen to belong to a product of intervals, but we will only discuss the real parameter case. For further theory on this topic, see [6].) Define a function $\bar{F}(\gamma,\bar{x})$, so that for some $\gamma_0,\gamma_m \in I$, $\bar{F}(\gamma_0,\bar{x}) = \bar{0}$ is solvable and $\bar{F}(\gamma_m,\bar{x}) \equiv \bar{f}(\bar{x})$. Rather than solving the original equation (1.1), we solve a series of intermediate problems by varying γ from γ_0 to γ_m, solving equations for $\gamma = \gamma_0, \gamma_1, \cdots, \gamma_m$. Now for each γ_i, the solution for the equation $\bar{F}(\gamma_{i-1},\bar{x})$ can serve as a good guess, provided the increments for the γ's are chosen properly. The parameter is inherent in some physical problems, or it may be inserted.

An obvious way to insert a parameter is to set $\bar{F}(\gamma,\bar{x}) = (1-\gamma)\bar{x} + \gamma\bar{f}(\bar{x})$, $0 \leq \gamma \leq 1$. Generally if there is a function $\bar{g}(\bar{x})$, whose solution is obtainable, (say a system of linear equations), we may set

$F(\gamma,\bar{x}) = (1-\gamma)\bar{g}(x) + \gamma\bar{f}(\bar{x})$, $0 \leq \gamma \leq 1$, [6]. Broyden [3] also suggested the following. Given some approximation $\bar{x}^{(0)}$ to the solution of (1.1), we solve the sequence $\bar{f}(\bar{x}) = \bar{f}(\bar{x}^{(0)})\cdot\gamma_i$, $i = 1,2,\cdots,m$. And the γ's are chosen so that $1 > \gamma_1 > \cdots > \gamma_m = 0$. If the equation $\bar{f}(\bar{x}) = \bar{f}^{(0)}\cdot\gamma$ has a solution $\bar{x}$ for all γ, $0 \leq \gamma \leq 1$, and the Jacobian of $\bar{f}$ exists and is non-singular at all these solutions, we may set $\bar{F}(\gamma,\bar{x}) = \bar{f}(\bar{x}) - \bar{f}^{(0)}\gamma$, and (1.1) can be solved. These are just some ways the parameter can be inserted. See Freudenstein and Roth [7] for a different example.

If the initial estimate to the solution is not sufficiently close to the root, then Newton's method, and in fact, most functional iteration methods frequently fail to solve the problem. With the use of a parameter, the same equation can often be solved. However, there is one problem involved with the use of a parameter. Suppose there is one or more γ_j, $1 \leq j \leq m$, such that the Jacobian of the function becomes nearly singular, then one or more of the intermediate problems cannot be solved. The whole process breaks down at this point and the solution cannot be reached. In practice, we detect a possible singular Jacobian from the increment of γ. Suppose $\gamma_i = \gamma_{i-1} + h$, $i = 1, \cdots, m$; we specify an upper limit

ℓ_1 to be the maximum number of iterations allowed for each problem. If the process does not converge within ℓ_1 iterations, we decide the previous solution as an initial guess is not close enough and the step size is too big; so h is halved. Now if the process converges in fewer than a lower limit ℓ_2 set for the number of iterations, we can take a bigger step and go faster, h is then increased. A proper increment for γ is important for it has to be small enough so that the solution at γ_{i-1} may serve as a good initial estimate for the next step, and big enough so that it will take a reasonable number of steps to converge. A threshold is set for the increment; if the increment of γ at any step falls below the threshold, we may encounter a nearly singular point. The change of parameter method originated by Davis is one way to deal with the singularities. I have used this method on several equations and found it to be successful. The experimental results are presented in Chapter V. Before we describe how the change of parameter method works, I would like to mention a method that Freudenstein and Roth use to treat singularities. Since I did not use this method in my programs, it will be mentioned only briefly here. Suppose the function $\overline{f}(\overline{x})$ in equation (1.1) to be solved is in the following form

$$f_i(\bar{x}) = \sum_{j=0}^{m} p_{ij}\ \phi_{ij}(\bar{x}), \quad i = 1, 2, \cdots, n \qquad (3.1)$$

where m is a positive integer and the ϕ's are some functions of $\bar{x}$. Another function $\bar{g}(\bar{x})$, in the form

$$g_i(\bar{x}) = \sum_{j=0}^{m} q_{ij}\ \phi_{ij}(\bar{x}), \quad i = 1, 2, \cdots, n \qquad (3.2)$$

has a known root. One may solve a set of M+1 intermediate equations $\bar{g}^{(k)}(\bar{x})$, where

$$g_i^{(k)}(\bar{x}) = \sum_{j=0}^{m} \{q_{ij}+(p_{ij}-q_{ij})\frac{k}{M}\}\phi_{ij}(\bar{x}), \ i=1,2,\cdots,n \qquad (3.3)$$

As we solve equation (3.3) for $k = 0,1,2,\cdots,M$ the coefficients of $\phi_{ij}(\bar{x})$ are varied, changing from those in equation (3.2) to those in equation (3.3). In this method, singularities are detected by the Jacobian determinant; if it drops below a predetermined value, we have a nearly singular point. At this point the coefficients are only incremented one at a time and the effect on the Jacobian determinant is noted, then the coefficients are selectively incremented so as to increase the value of the Jacobian determinant above the predetermined minimum. After this is done, the regular increment may be re-introduced. Even though there is no guarantee that this routine would work, Freudenstein and Roth found in practice they do eliminate the singularities in a large majority of cases.

We next outline the change of parameter method. For further theory of this method, the readers are referred to Davis [6]. Suppose the Jacobian matrix of the equation

$$\overline{F}(\gamma,\overline{x}) = \overline{0} \qquad (3.4)$$

becomes nearly singular at γ^*, $\gamma_0 < \gamma^* < \gamma_m$. We define a new parameter $\rho \in I^*$, another real interval, and a continuously differentiable function G which maps a neighborhood of $(\rho^*,\overline{x}^*)$ into I, such that $\gamma^* = G(\rho^*,\overline{x}^*)$. Then the function $\overline{F}$ near γ^* is defined by

$$\overline{F}(\gamma,\overline{x}) = \overline{F}(G(\rho,\overline{x}),x) = \overline{F}^*(\rho,\overline{x}) \qquad (3.5)$$

While the Jacobian matrix $[\frac{\partial F_j}{\partial x_k}]_{j,k}$ is singular at γ^*, the matrix $[\frac{\partial F_j^*}{\partial x_k}]_{j,k}$ may not be. Differentiating (3.5) with respect to x_k, we get

$$\frac{\partial F_j}{\partial x_k}(\rho,\overline{x}) = \frac{\partial F_j}{\partial x_k}(\gamma,\overline{x}) + \frac{\partial F_j}{\partial j}(\gamma,\overline{x})\cdot\frac{\partial G}{\partial x_k}(\rho,\overline{x}) \qquad (3.6)$$

If J denotes the Jacobian of $\overline{F}$ at $\overline{x}$, J^* the Jacobian of $\overline{F}^*$, $\overline{F}_\gamma$ the derivative of $\overline{F}$ with respect to gamma and $\overline{G}_x$, a row vector, the derivative of G with respect to $\overline{x}$, equation (3.6) can be rewritten in vector notation

$$J^* = J + \overline{F}_\gamma \cdot \overline{G}_x \qquad (3.7)$$

In practice, the function G is chosen to be $G(\rho,\overline{x}) = \rho+\overline{v}\cdot\overline{x}$, $\overline{v}$ being a n-dimensional row vector $c(1,1,\cdots,1)$, or $c(-1,1,\cdots,1)$, where c is a scalar factor determined so that the change of parameter method is most effective, if it works at all. Now the change of parameter equation is

$$\gamma = \rho + \overline{v} \cdot \overline{x} \qquad (3.8)$$

with this particular G, we rewrite (3.7) as

$$J^* = J + \overline{F}_\gamma \cdot \overline{v} \qquad (3.9)$$

Cullop [4] suggested a proper choice of c may be one so that $\|J\| = \|\overline{F}_\gamma \cdot \overline{v}\|$. If we choose to use the maximum norm, this means $\|J\|_\infty = c \cdot n\|\overline{F}_\gamma\|_\infty$, so $c = \dfrac{\|J\|_\infty}{n\cdot\|\overline{F}_\gamma\|_\infty}$, where both J and $\overline{F}_\gamma$ are evaluated at the last convergent solution $\overline{x}^{(j)}$ at the $j^{\underline{th}}$ step. After making some progress with the new parameter ρ, we switch back to the original parameter and proceed until the last equation is solved.

The change of parameter method may or may not work. If the dimension of the null space of J evaluated at $\gamma^*,\overline{x}^*$ has dimension more than the dimension of the parameter space, (in this case, one) or $\overline{F}_\gamma$ is in the range of the Jacobian, this method will not work. In the first case, a parameter with dimension greater than one may be used, even though it is more complicated. In the latter case, a different parametrized equation has to be set up and a new set of equations solved.

From the experience with several equations, the change of parameter method seemed to work very well. In the next chapter, programming details about the change of parameter are explained.

CHAPTER IV

Details of a Method Not Using Jacobian

In this chapter, we give the programming details of a method which solves a system of non-linear equations with a scalar parameter. In this algorithm, all the intermediate equations are solved using Broyden's method 1 as described in Chapter II. A subroutine RHO is written to take care of the change of parameter, if any singular points should arise during the process. First we give the information about how this program can be used, the various subroutines used in the program, the main program and finally some flow charts.

Subroutines To be Provided by the User

There are three FORTRAN subroutines the user must provide.

(1) For initial values.

```
SUBROUTINE GIVEN (N, G, GL, SH, HL, X, CF)
COMMON (If needed)
DIMENSION  X(10)
```

In this subroutine, all the variables in the parameter list are to be assigned a value in the subroutine. It is called only once at the beginning of the main program. Any other initialization for the user subroutines can also be done here. Below is a list of the names of the variables and their corresponding inputs.

Parameter	Input
N	Dimension of the system, not to exceed 10.
G	Initial value of gamma. (The value of gamma for which the solution of $\bar{F}(\gamma,\bar{x})=\bar{0}$ is known or solvable.)
GL	The last value of gamma. (The gamma for which $\bar{F}(\gamma,\bar{x})=\bar{0}$ is to be solved.)
SH	Initial increment for gamma.
HL	The threshold for gamma, in absolute value.
X	An n-dimensional vector, the initial guess to solve $\bar{F}(\gamma,\bar{x})=\bar{0}$.
CF	Convergence factor, if $\|\bar{x}^{(i+1)}-\bar{x}^{(i)}\|_2 \leq CF\cdot\|\bar{x}^{(i+1)}\|_2$ the iteration will stop.

(2) For function values

```
SUBROUTINE   PG (G, X, F)
COMMON       (If needed)
DIMENSION    X(10), F(10)
```

The function value of $\bar{F}(G,X)$ is calculated and stored in F.

(3) For the partial derivatives of $\bar{F}$ with respect to gamma.

```
SUBROUTINE PGAM (N, X, PL)
COMMON (If needed)
DIMENSION  X(10),PL(10)
```

The partial derivative of $\bar{F}$ with respect to G is evaluated analytically and assigned to PL.

The main program and all its necessary subroutines are stored in *ASEN. *ASEN and the three user provided object decks must be loaded together. If the user subroutines are stored in a saved file, say USER, then after the job card, we only need

```
7
8 LOAD, *ASEN, USER
  RUN
7
8 LOGOFF
```

Subroutines in *ASEN

Below we give the names of each subroutines used in the main program along with a brief description about what is accomplished in each. In all of the cases below, N stands for the number of variables in the problems to be solved. In some cases further details are given later in this chapter.

(1) SUBROUTINE AX(A, X, N, Y)

DIMENSION A(10, 10), X(10), Y(10)

Multiply the matrix A with a vector X to get a column vector Y. $Y_i = \sum_{j=1}^{N} A_{ij} X_j$

(2) SUBROUTINE XA(X, A, N, Y)

DIMENSION A(10, 10), X(10), Y(10)

Similar to (1), but X now is a row vector, the order of multiplication is XA = Y, $Y_i = \sum_{j=1}^{N} X_j A_{ji}$.

(3) SUBROUTINE PROD (N, A, B, P)

DIMENSION A (10), B(10)

P is a scalar product of vectors A and B,

$$P = \sum_{i=1}^{N} A_i B_i$$

(4) SUBROUTINE PRODM (N, A, B, C)

DIMENSION A(10), B(10), C(10, 10)

Multiply a column vector A with a row vector

B to get a matrix C, $C_{ij} = A_i B_j$.

(5) SUBROUTINE ANORMM (C, N, UBM)

DIMENSION A(10, 10)

UBM is the maximum norm of the matrix A.

$$UBM = \underset{i}{Max} \sum_{j=1}^{N} |A_{ij}| \qquad i=1,2,\cdots,N$$

(6) SUBROUTINE ANORMINF (N,X,P)

DIMENSION X(10)

The value P is defined to be the maximum norm of the vector X.

$$P = \underset{i}{Max} |X_i| \qquad i=1,2,\cdots,N$$

(7) SUBROUTINE TNORM (N,V,TN)

DIMENSION V(10)

TN is the Euclidean norm of the vector V.

$$TN = \sqrt{\sum_{i=1}^{N} V_i^2}$$

(8) SUBROUTINE INVERSE (N, D, B)

DIMENSION X(10), B(10, 10), D(10, 10), A(10, 20)

Given any matrix D as input, the inverse D is obtained by Gauss elimination with partial pivoting. (Row interchanges, if needed) output is the matrix B.

(9) SUBROUTINE GJACOB (N, H, G, X, PX, P)

DIMENSION X(10), PX(10, 10), Y(10), P(10), PP(10)

The Jacobian of the function $\overline{F}(G,X)$ is approximated using finite differences. The function value $P = \overline{F}(G,X)$ has to be calculated before calling GJACOB. H is a small scalar. The Jacobian is stored in PX.

$$PX_{ij} = \frac{P_i(X_1,X_2,\cdots,X_{j-1},X_j+H,X_{j+1},\cdots,X_N)-P_i(X)}{H}.$$

(10) SUBROUTINE JINVERSE (N, G, X, F, H, SH, DN)

DIMENSION X(10), F(10), H(10,10), D(10,10), B(10,20)

Given current values of the parameter G, X, and $F = \overline{F}(G,X)$ and a scalar SH, the approximate Jacobian D is first calculated and then inverted. Outputs are a scalar DN, the maximum norm of D; and a matrix H, the inverse of D.

(11) SUBROUTINE PREDICT (N, X, PI, PL, SH)

DIMENSION X(10), PI(10,10), PL(10), XP(10)

Upon entry to this subroutine, X solves $\overline{F}(G,X)=\overline{0}$. By calling PREDICT. X is "corrected" using Taylor series expansion so that it may be a better initial guess to solve $\overline{F}(G+SH,X)$. The partial derivatives with respect to gamma, PL; the approximate inverse of the Jacobian, PI; all evaluated at X, should be available before calling this subroutine.

(12) SUBROUTINE RINC (SH, SR, H, V, PL, N)

DIMENSION H(10,10), V(10), PL(10)

When we change the parameter using the equation $\gamma = \rho + v \cdot x$, the increment for the ρ parameter SR is calculated in RINC.

(13) SUBROUTINE GINC (SH, SR, H, V, PL, N)

DIMENSION H(10,10), V(10), PL(10).

Similar to RINC, but this time the increment of γ, SH is calculated.

(14) SUBROUTINE BROY (N, N2, I, H, X, F, ID, V, G, R, CF)

DIMENSION X(10), F(10), P(10), FM(10), Z(10), T(10),
B(10,10), H(10,10), V(10), Y(10)

This subroutine solves the equation $\overline{F}(G,X) = \overline{0}$ using Broyden's method (or $\overline{F}(R+V \cdot X, X) = \overline{0}$, if it is in the R parameter). The value ID equals to zero indicates it is in the G parameter, otherwise R. I is the total number of iterations it took to satisfy the stopping criterion, if convergent. CF is the same as described in SUBROUTINE GIVEN. N2 is the maximum number of iterations allowed each time BROY is called. If the stopping criterion is still not met on the N2th iteration, the process is considered non-convergent. H is an approximate inverse of the Jacobian $\overline{F}$ or $\overline{F}^*$.

(15) SUBROUTINE RHO (N, N1, N2, G, HL, IC, SH, SR,IT,
IFF, X, XM, XM1, V, PL, H, HB, CF)

```
DIMENSION X(10),XM(10), XM1(10), PL(10), V(10),
      H(10,10), HB(10,10), Q(10), P(10), F(10),
      Z(10), B(10,10), U(10).
```

When a possible singular point is detected, the subroutine RHO is called to effect a change of parameter. If the change of parameter works, after nine successful steps in RHO, the routine would go back to the main program. XM, XM1 are storage locations for values of X from the two previous successful steps; HB is the approximate inverse of the Jacobian of $\overline{F}*(R,X)$.

General Strategy

In this section, we will describe how the main program works. Given an equation $\overline{F}(\gamma,\overline{x})$; two particular values of γ, γ_0, and γ_m; and a good estimate to a solution of $\overline{F}(\gamma_0,\overline{x}) = \overline{0}$, this program attempts to solve $\overline{F}(\gamma_m,\overline{x}) = \overline{0}$, by varying γ from γ_0 to γ_m and solving the equation $\overline{F}(\gamma,\overline{x}) = \overline{0}$ for some intermediate gammas. Each of these equations are solved by Broyden's method. If no singular points are encountered as we solve $\overline{F}(\gamma_i,\overline{x}) =\overline{0}$, $i = 0,1,2,\cdots,m$, then the main subroutine we use is BROY. BROY will be flow charted in the next section. In solving $\overline{F}(\gamma_i,\overline{x}) = \overline{0}$, we allow a maximum of N2 iterations (N2 is taken to be eight, the choice of this number is discussed later in this chapter.) If the two norm of the differences of $\bar{x}^{(j+1)}$ and $\bar{x}^{(j)}$ is less than or equal to the convergence factor CF times the two norm of $\bar{x}^{(j+1)}$, the $j+1^{th}$ iterate for the solution to $\bar{F}(\gamma_i,\bar{x}) = \bar{0}$, $1 \leq j \leq 7$, then the method is considered successful, otherwise not successful.

In deciding an approximate increment for the parameter, we choose to do the following. If the process converges in exactly N1 iterations (we will take N1 to be six; this choice will be discussed below), we will leave the present increment SH alone; if more than six, the step size is halved, regardless of convergence or nonconvergence. Every time the increment is halved, its absolute value is checked against the threshold HL set by the user, if $|SH|$ is less than HL, a change of parameter will take place. In the case of failure of convergence, the previous convergent value of $\bar{x}$, the approximate inverse of the Jacobian, H; and gamma G from the last step are restored. In the case of two or fewer iterations, we decide that the step size is too small, and is replaced by three times SH. For the case of three to five iterations, the increment is doubled. The only exception to the above rule of increasing the step size is at the step immediately after SH was halved; in this case, the step size is not to be increased. The above discussion on increasing or decreasing the step SH of the G parameter works the same way for the SR of the R parameter. The numbers six, eight, etc., used here in deciding the increase or decrease of the increment are chosen quite arbitrarily, but they seem to work fairly well. As for the maximum number of iterations allowed to solve each equation, it is decided based on Cullop's [4] experiments. He used six as an upper limit in the modified Newton iteration, since the methods not using Jacobians supposedly take more iterations than Newton's, we allow two more iterations in the upper limit. It is not known yet whether some other numbers will be more efficient. If the iteration is successful, the values of X, H, G are stored, and then printed in the following order. The value of gamma, G; the number of iterations it took to converge, I; the cummulative number of iterations, IT; the number of times the Jacobian is calculated by finite differences, IFF; the increment of gamma, SH; (in the RHO subroutine, the increment of R, SR is printed instead of SH, and then R is also printed). On the next line, print all components of the solution X. The value of G is compared with the last value of gamma, GL; (GL = γ_m), if they are equal, the final answer of the root X and the function value at X are printed. If G is not equal to GL, the parameter is incremented and begin to solve the next equation. In the gamma parameter before we add SH to G, we make sure the resulting value of G does not pass by the final gamma GL. This is done by making sure $|SH| \leq |GL-G|$, if SH has the same sign as GL-G.

We have mentioned in Chapter III, the solution $\bar{x}$ to $\bar{F}(\gamma_i,\bar{x}) = \bar{0}$ may serve as an initial guess for $\bar{F}(\gamma_{i+1},\bar{x}) = \bar{0}$. However, we can improve this guess with little extra work. This is done in subroutine PREDICT. We consider $\bar{x}$ as a function of γ, where $\bar{F}(\gamma,\bar{x}) = \bar{0}$, let $\bar{x}^{(i)} = \bar{x}(\gamma_i)$, and $\bar{F}(\gamma_i,\bar{x}^{(i)}) = \bar{0}$, upon differentiation with respect to γ, we get

$$\bar{F}_x(\gamma_i,\bar{x}^{(i)})\,\frac{d\bar{x}^{(i)}}{d\gamma} + \bar{F}_\gamma(\gamma_i,\bar{x}^{(i)}) = \bar{0}$$

or

$$\bar{x}'(\gamma_i) = \frac{d\bar{x}^{(i)}}{d\gamma} = -\bar{F}_x(\gamma_i,\bar{x}^{(i)})^{-1}\,\bar{F}_\gamma(\gamma_i,\bar{x}^{(i)}) \qquad (4.1)$$

Now write a two term Taylor series expansion for $\bar{x}(\gamma)$ about γ_i, we get

$$\bar{x}^{(i+1)} \cong \bar{x}^{(i)} + \bar{x}'(\gamma_i)SH \qquad (4.2)$$

Equation (4.2) will provide a good approximation if SH is sufficiently small; combining this with equation (4.1), we have

$$\bar{x}^{(i+1)} \cong \bar{x}^{(i)} - \bar{F}_x(\gamma_i,\bar{x}^{(i)})^{-1}\,\bar{F}_\gamma(\gamma_i,\bar{x}^{(i)})SH \qquad (4.3)$$

Hopefully, the right hand side of equation (4.3) will give a better approximation to $\bar{x}^{(i+1)}$ than $\bar{x}^{(i)}$ alone.

From now on, H will denote an approximate inverse of the Jacobian, and $\overline{PL}$ denotes the partial derivative of $\overline{F}$ with respect to γ. Rewriting (4.3) in this new notation, we get

$$\overline{x}^{(i+1)} \cong \overline{x}^{(i)} - H \cdot \overline{PL} \cdot SH \qquad (4.4)$$

In the RHO parameter, the above theory works the same way. Suppose the change of parameter is given by $\gamma = \rho+\overline{v}\cdot\overline{x}$, $\overline{v}$ is a constant row vector and

$$\overline{F}(\gamma,\overline{x}) = \overline{F}(\rho+v\cdot\overline{x},\overline{x}) = \overline{F}^*(\rho,\overline{x}) \qquad (4.5)$$

Let HB denote an approximate inverse of the Jacobian to the function on the right hand side of equation (4.5). Since the partial derivative of $\overline{F}$ with respect to γ is the same as that of $\overline{F}^*$ with respect to ρ, to get a better prediction in RHO, we simply change H to HB and SH to SR in equation (4.4).

When $|SH| < HL$, the subroutine RHO is called, and the parameter ρ is tried, where

$$\gamma = \rho+\overline{v}\cdot\overline{x}, \quad \overline{v} = c(1,1,\cdots,1). \qquad (4.6)$$

c is a scalar factor as described in Chapter III. Then HB and SR are calculated. The derivation of HB is covered in Chapter II and III, (equations 2.19, 3.3, 3.4). We will just restate the result here with the notation of this chapter.

$$HB = H - \frac{H\cdot\overline{PL}\cdot\overline{V}\cdot H}{1+\overline{V}\cdot H\cdot\overline{PL}} \qquad (4.7)$$

The R increment SR is calculated in RINC. If we differentiate equation (4.6) with respect to γ, we get

$$1 = \frac{d\rho}{d\gamma} + \overline{v} \cdot \frac{d\overline{x}}{d\gamma} \qquad (4.8)$$

Combining this with equation (4.1), using notation of this chapter we get

$$\frac{d\rho}{d\gamma} = 1 + \overline{V}\cdot H\cdot\overline{PL} \qquad (4.9)$$

Now $d\rho$, $d\gamma$ are fair approximations to SR and SH. SUBROUTINES RINC and GINC calculate SR from SH and vice versa based on equation (4.9), replacing $d\rho$, $d\gamma$ by SR and SH.

At the first iteration where we switch from one parameter to another, we recalculate the solution at the same point (the last convergent point before changing parameter) with the new parameter, this usually takes only one, at most two iterations. The reason for recalculating the same point is to ensure that the new Jacobian is non-singular. After eight more successful steps in the new parameter RHO, the program returns to the G parameter. Solving $\overline{F}^*(\rho,\overline{x}) = \overline{0}$ involves slightly more work than solving $\overline{F}(\gamma,\overline{x}) = \overline{0}$, for the equation provided by the user is in the G parameter, so whenever the function value is required, the value of G has to be available before $\overline{F}$

can be evaluated. In RHO, since G is a function of both R and X, when either R or X is changed, G must be recalculated. Also in RHO, there is no way to check whether G is equal to GL. Aside from these difficulties, solving $\overline{F}^*(\rho,\overline{x}) = \overline{0}$ in RHO is on the whole the same as solving $\overline{F}(\gamma,\overline{x}) = \overline{0}$. We use the same criterion for convergence, and increasing or decreasing step size in both. In the RHO parameter, however, if $|SR|$ falls below HL, there are several alternative approaches. If there had been six or more successive iterations in RHO, we go back to the original parameter assuming the singular point is passed. In the case of fewer than six iterations, another vector $\overline{v} = c(-1,1,\cdots,1)$ is tried, giving a new parameter R'. Switch back to the G parameter if we can get at least six successful iterations. Suppose the second change of parameter is tried and could only bring from three to six successful steps before $|SR|$ falls below HL again, the G parameter is tried anyway, hoping these steps in R' have brought it past the singular point. However, if R' only yields two or fewer successful steps, we consider it a failure and set IC = 0 to indicate this fact, then return to the gamma parameter. In this case, if only one (or no) successful step in the gamma parameter is obtained, a message will be printed, and the user should find a different way to parametrize the equation.

It is found in all test cases, nine successful steps in R suffices to bring the equation past the singular point. In case that nine steps are not sufficient (this fact is indicated by the failure of convergence if switch back to the G parameter), the program will go back to the R parameter and continue from where it had left off without having to calculate HB and SR again.

At the beginning of the program, and whenever a change of parameter takes place (from G to R or vice versa), the approximate Jacobian is calculated and then inverted. The matrix H (or HB) is being corrected as we go along. If there is no change of parameter needed, only one Jacobian is evaluated. Even in the change of parameter case, it is not clear, without further experiments, whether the finite difference evaluation of the inverse of the Jacobian may be skipped without leading to slower convergence or even non-convergence.

One last point we ought to mention here before we turn to the flow charts is that whenever we change from one parameter to the other, it is possible that we may proceed in a direction where the previous X are calculated again. If we have no way to detect this situation, the value GL may never be reached, and the equation is not solved. To avoid retracing the curve, after the second successful step in a different parameter, we make sure at least one component of the solution X or the gamma parameter does not retrace itself. We can illustrate this as follows. Suppose we change from G to R, we label the three consecutive points of X(G), say P_1, P_2, P_3; the points P_1, P_2 are calculated by solving $\overline{F}(G,X) = \overline{0}$, P_2 (again) and P_3 are found by solving $\overline{F}^*(R,X) = \overline{0}$. Calculate the corresponding G in each of these points, say G_1, G_2, G_3. If at least one of the following equations is true, we keep the calculated step SR. The equations are

$$(G_3-G_2)(G_2-G_1) > 0$$

and

$$(X_i^{(3)} - X_i^{(2)})(X_i^{(2)} - X_i^{(1)}) > 0 \qquad 1 \leq i \leq N$$

If none of the above equations are true, change the sign of SR and recalculate the point P_3. This idea of checking for retracing curve is due to Cullop [4].

<u>Flow Charts</u>

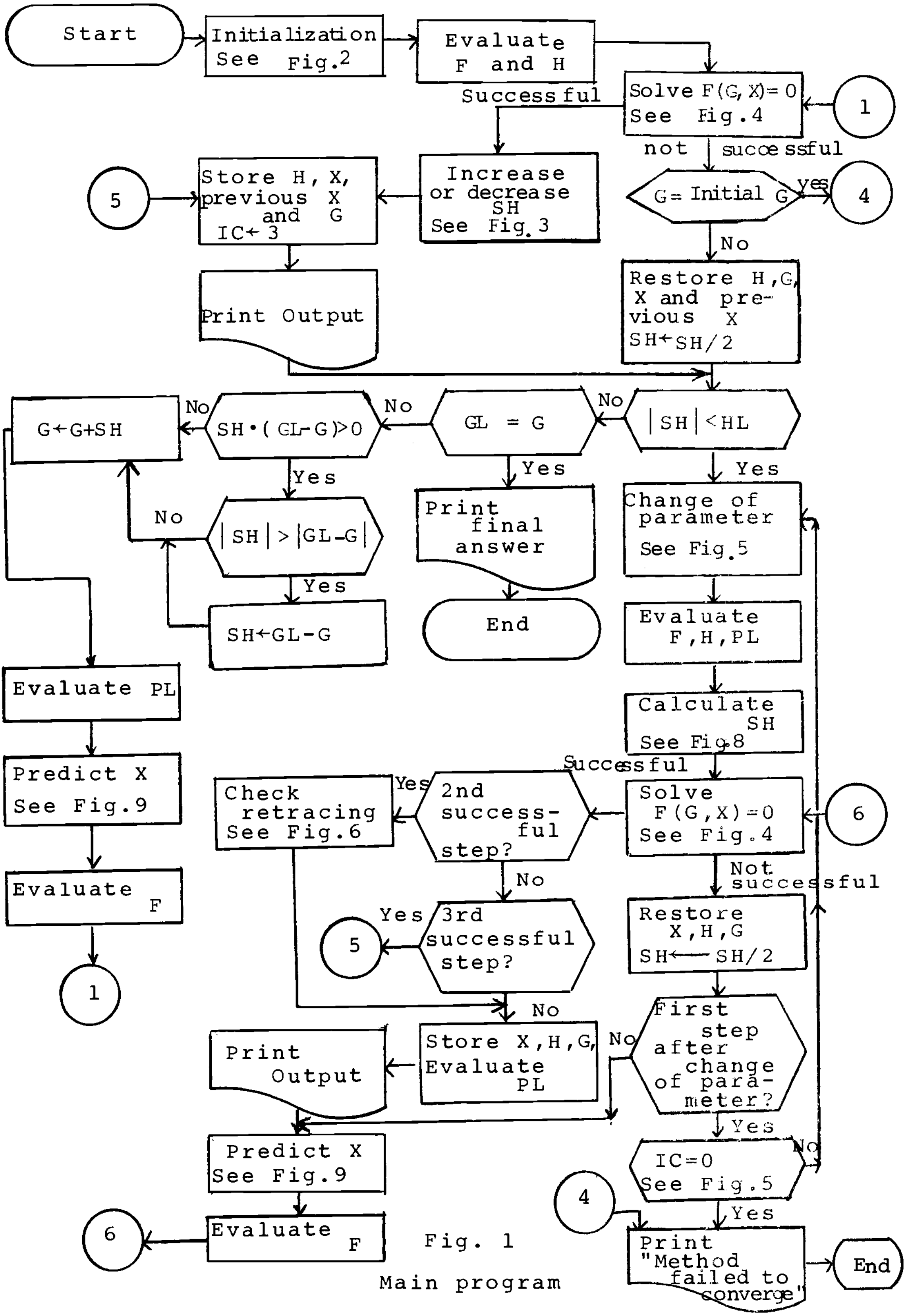


Fig. 1

Main program

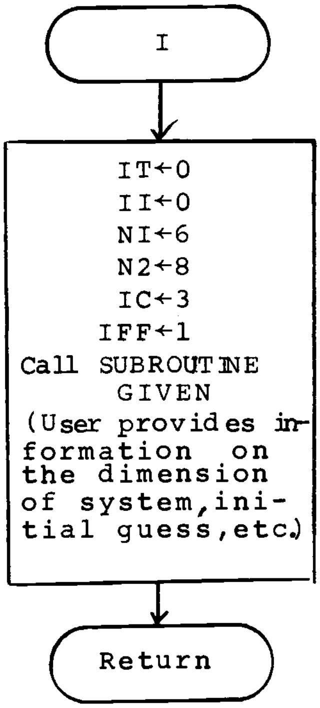

I
IT←0
II←0
NI←6
N2←8
IC←3
IFF←1
Call SUBROUTINE GIVEN
(User provides information on the dimension of system, initial guess, etc.)
Return

Fig.2

Initialization

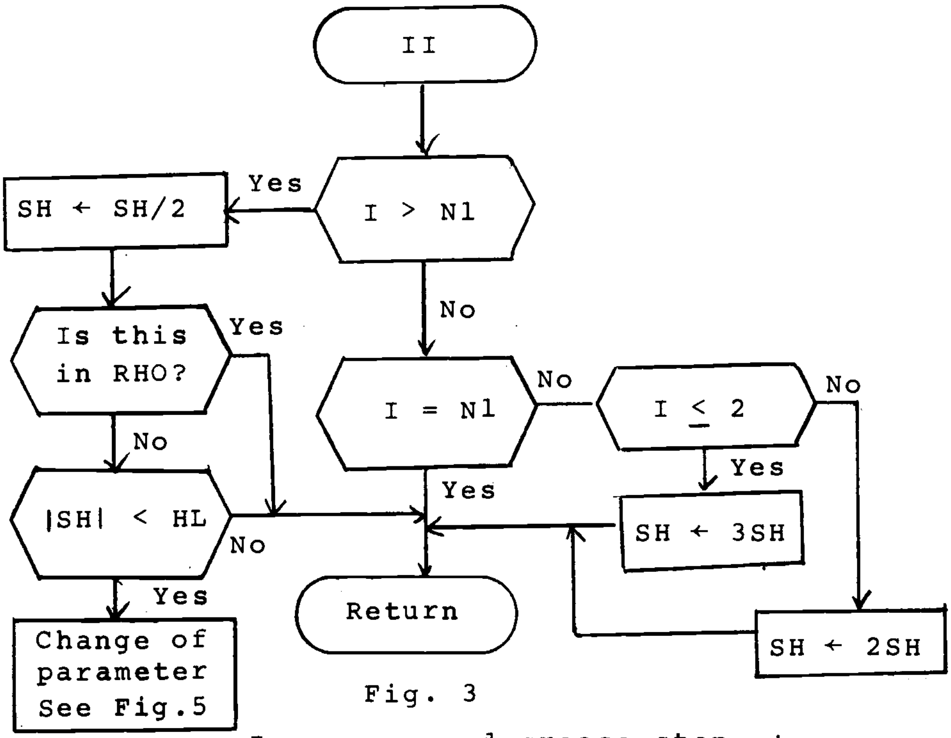


Fig. 3

Increase or decrease step size of the parameter

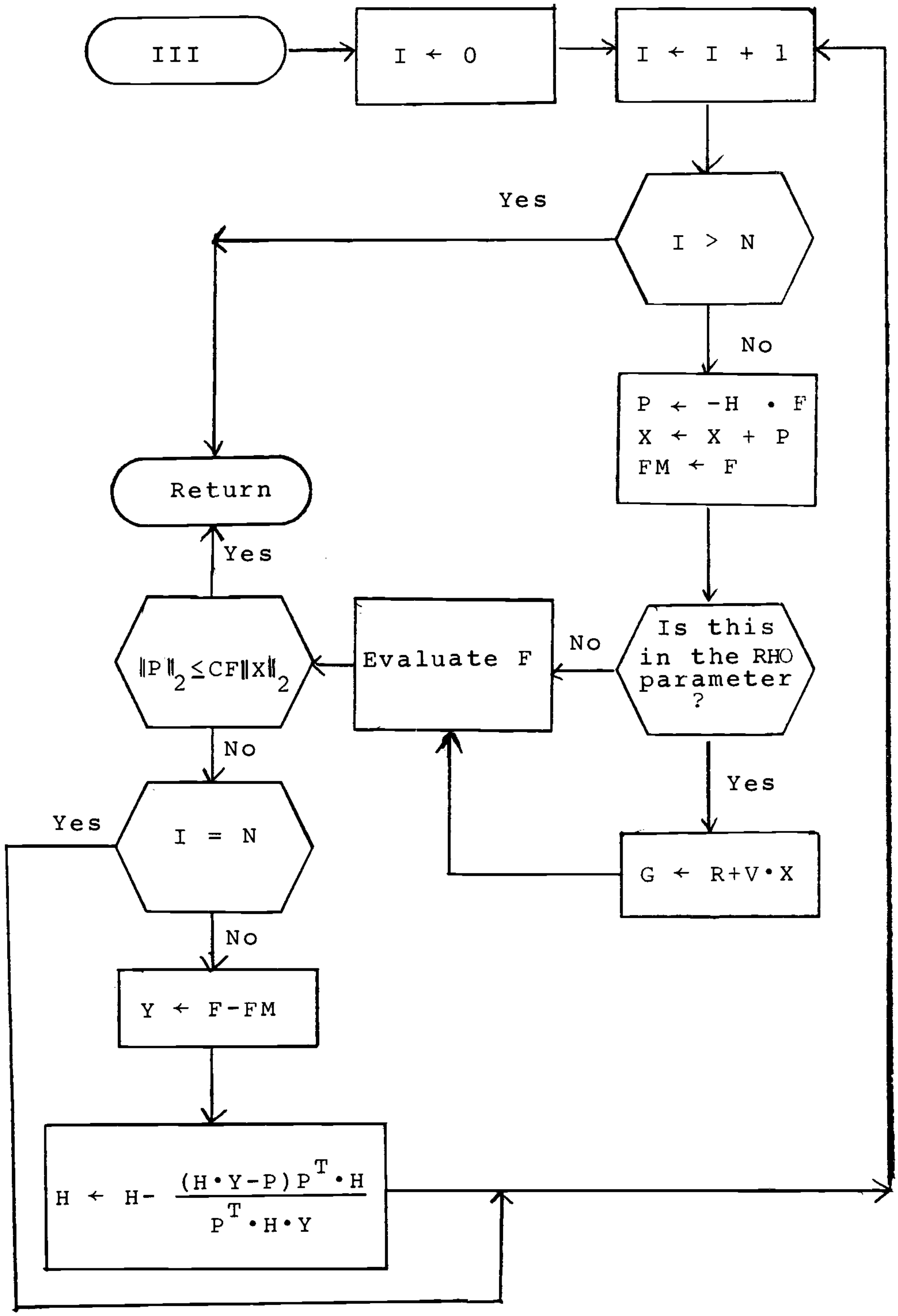


Fig. 4

Solve F(G,X)=0 or F(R+V•X,X)=0 by Broyden's Method

(SUBROUTINE BROY)

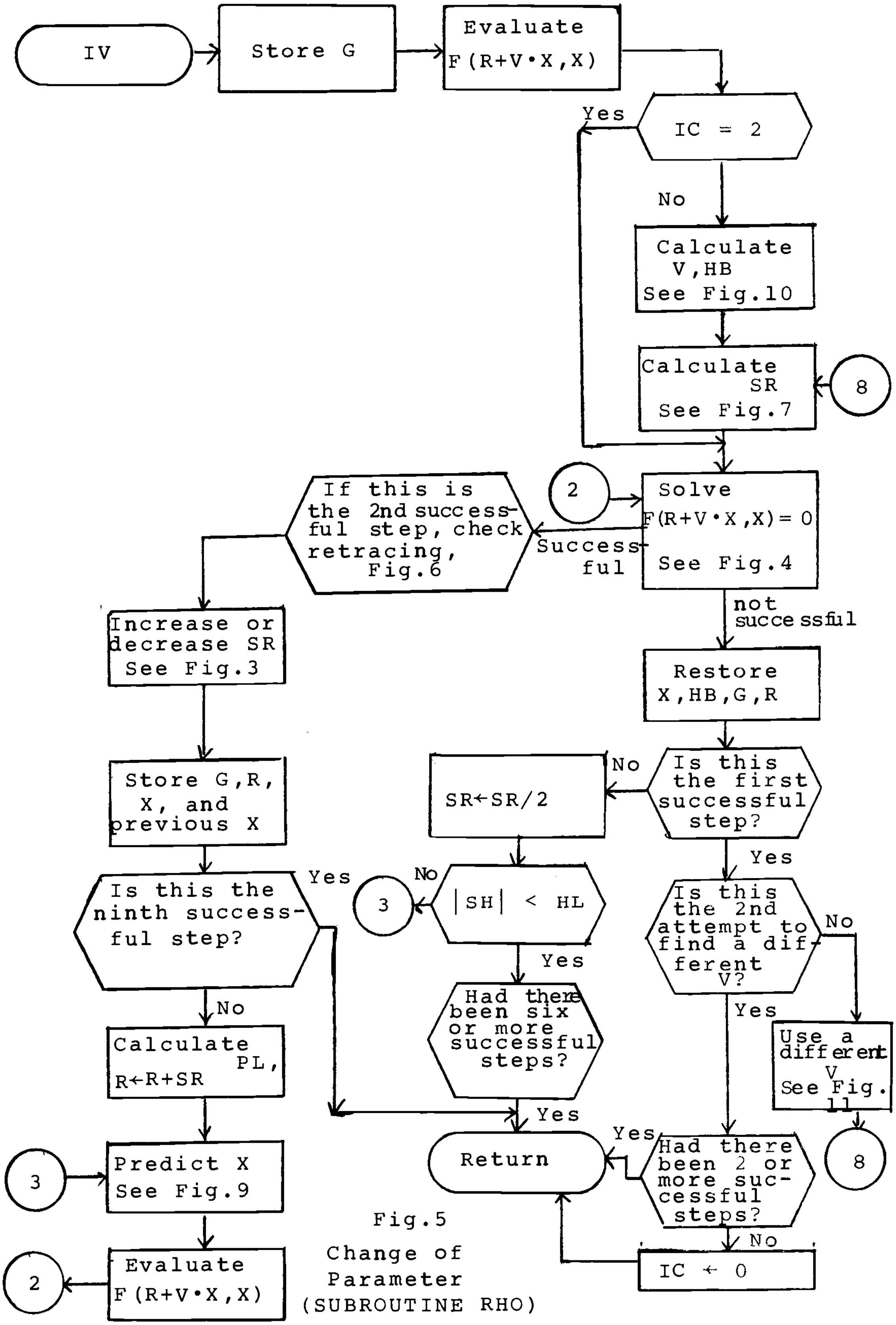

IV
Store G
Evaluate F(R+V•X,X)
IC = 2
Yes
No
Calculate V,HB See Fig.10
Calculate SR See Fig.7
8
2
Solve F(R+V•X,X)= 0 See Fig.4
Success-ful
If this is the 2nd success-ful step, check retracing, Fig.6
Increase or decrease SR See Fig.3
Store G,R, X, and previous X
Is this the ninth success-ful step?
Yes
No
Calculate PL, R←R+SR
3
Predict X See Fig.9
2
Evaluate F(R+V•X,X)
not successful
Restore X,HB,G,R
Is this the first successful step?
No
SR←SR/2
No
3
|SH| < HL
Yes
Had there been six or more successful steps?
Yes
Return
Yes
Is this the 2nd attempt to find a dif-ferent V?
No
Use a different V See Fig. 11
8
Yes
Had there been 2 or more suc-cessful steps?
Yes
No
IC ← 0

Fig.5

Change of Parameter

(SUBROUTINE RHO)

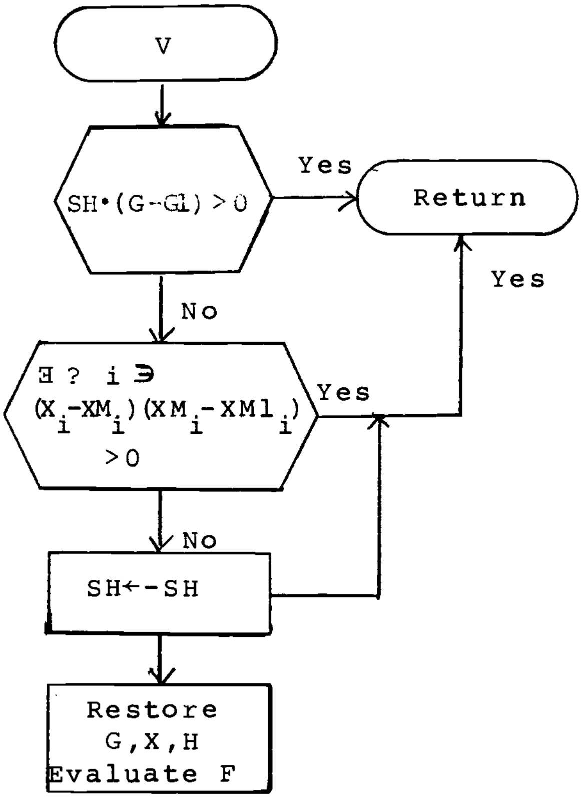


Fig.6 Check for retracing.

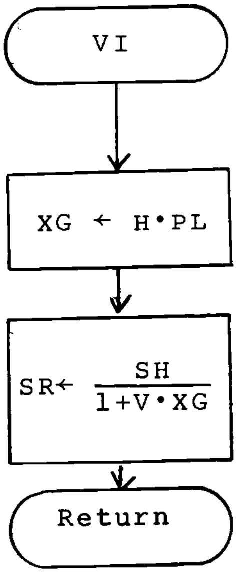


Fig.7

Calculate Rho increment (SUBROUTINE RINC)

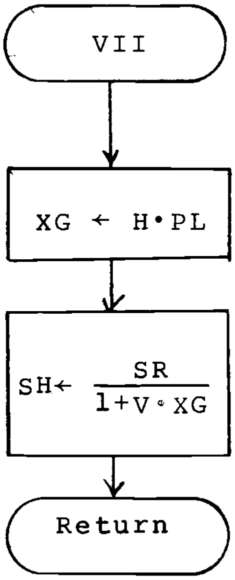


Fig.8

Calculate Gamma increment (SUBROUTINE GINC)

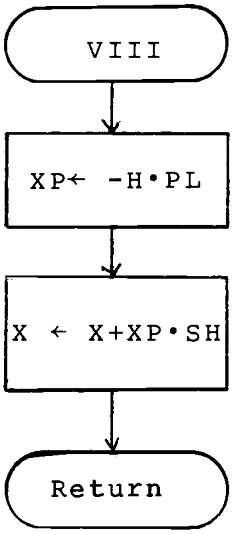


Fig. 9

Predicts X as initial guess for the next step

(SUBROUTINE PREDICT)

(In the RHO parameter, replace H by HB, SH by SR)

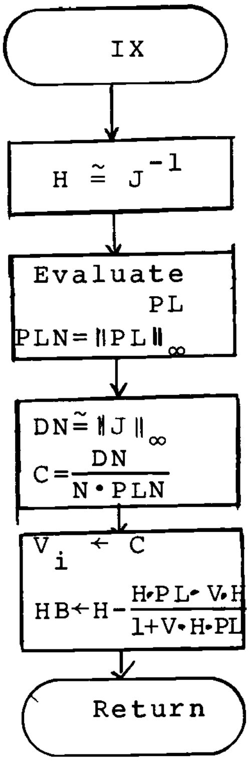


Fig.10

Calculate V and HB

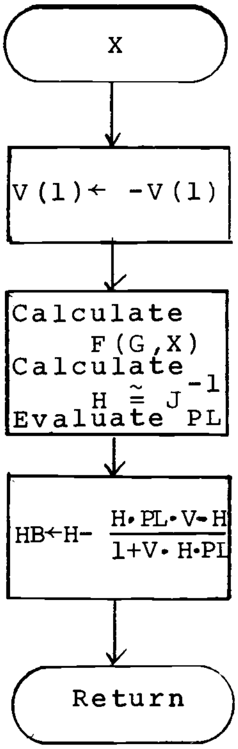


Fig.11

Find a different V

CHAPTER V

Comparison of Some Methods with Experimental Results

Basis for Comparison and Test Equations

The method described in Chapter IV is compared with the modified Newton's method programmed by Cullop [4]. Both of the methods use a parameter to ensure a good initial guess to be available. Also each program has a subroutine to take care of the change of parameter, if needed. Cullop's modified Newton's method works as follows. In applying the iterative scheme defined in equation (2.1), the Jacobian matrix is calculated on the first and fifth iteration. At each iteration, the system $J(\bar{x}^{(i+1)}-\bar{x}^{(i)}) = -\bar{f}^{(i)}$ is solved. The basis of comparison is the actual computer time used. We also list the total number of iterations for reference in each case. Both methods are programmed in FORTRAN, and run on CDC 33000. We will first list the six test equations to be solved, then the ones with parameter, and the initial conditions.

Case 1. This is a test equation used by Freudenstein and Roth [7]

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

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

$$F_1 = -71+x_1+((-x_2-13)x_2-50)x_2+\gamma(58+(18x_2+48)x_2)$$

$$F_2 = 129+x_1+((x_2-19)x_2+106)x_2+\gamma(-158+(-18x_2+120)x_2)$$

Initial values: $x_1 = 15$, $x_2 = -2$, $0 \leq \gamma \leq 1$

Case 2. This test equation differs slightly from the above case, but no change of parameter is needed.

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

$$f_2 = -29 + x_1 + ((x_2-14)x_2+262)x_2$$

$$F_1 = -71+x_1+((-x_2-13)x_2-50)x_2+\gamma(58+(18x_2+52)x_2)$$

$$F_2 = 129+x_1+((x_2+19)x_2+106)x_2+\gamma(-158-(33x_2-156)x_2)$$

Initial values $x_1 = 15$, $x_2 = -2$, $0 \leq \gamma \leq 1$

Case 3. Polynomial equations

$$F_1 = \gamma(x_1x_2x_3+4x_3^3)+x_1^2+x_2-x_1x_3-8$$

$$F_2 = \gamma(x_3x_2^2+x_1x_3)-2x_1+x_2/2+x_3^2+4$$

$$F_3 = \gamma(x_1^2+x_2^2+x_3^2)x_3+x_1x_3-x_2^2+x_2x_3-1$$

Initial values are $x_1 = 3$, $x_2 = 2$, $x_3 = 1$;

they solve the system when $\gamma=0$. Different equations are obtained when we set different final values for gamma.

If the last γ is chosen to be one, no change of parameter is necessary, however, if the last γ is set to be a negative number, say -1, there are several times a change of parameter is needed.

Cases 4-6.

These three examples are taken from elasticity described in Davis' Technical Report [6]. They describe the deformation of a shallow clamped symmetric spherical cap under uniform pressure. These equations are very complicated and hence not given here. Two of the parameters involved in these equations are λ and μ^2. λ is inversely proportional to the pressure applied to the cap and μ^2 is related to the thickness of the cap. In each of these cases, μ^2 is held constant, and the λ parameter is incremented from zero to one. It is found that when μ^2 is small, (less than seven), no change of parameter is necessary, but if μ^2 is big, then there are several singular points and the bigger μ^2 gets, more singular points arise.

Case 4.	$\mu^2 = 5$	$0 \leq \lambda \leq 1$
Case 5.	$\mu^2 = 9.8$	$0 \leq \lambda \leq 1$
Case 6.	$\mu^2 = 15$	$0 \leq \lambda \leq 1$

Each of the test cases involves six equations and six unknowns. The initial values of $\bar{x}$, when $\lambda=0$, are all set to be (1,1,1,1,1,1).

The above test equations were solved by both the modified Newton's method and the Broyden's method with the same initial guess to $\bar{x}$, convergence criterion and the threshold for the step size of the parameter.

Experimental Results and Case Discussions.

Case 1. Using the modified Newton's method, the solution (5,4) is found without changing the parameter. With Broyden' method, a change of parameter is necessary at $\lambda = 0.92362$ and finally the program returns to the original parameter and the solution at $\lambda = 1$ is reached.

	Modified Newton	Broyden
CPU time (sec.)	7.3	6.9
No. of iterations	157	425

If we try to solve this problem with Newton's method, using the same initial guess (15,-2), but without the use of a parameter, the iteration will not converge to any solution, see [7]. Broyden [3] has also worked on this equation but used a different parameter θ, and the method failed. The failure of the method may due to the singularity of the Jacobian as θ decreases from 0.418 to 0.368. Since Broyden's method described in his paper did not take care of singularities, the process broke down and the solution is never reached. Frudenstein and Roth [7] also solved this problem with their parameter perturbation procedure.

Case 2. In this test equation, both the modified Newton's method and Broyden's method converge to the solution (-8.43481, -1.91165) when $\gamma=1$ without a change of parameter.

	Modified Newton	Broyden
CPU time (sec.)	6.4	4.1
No. of iterations	9	17

Case 3. In this test case, the parameter is set to be equal to zero initially; then we let γ increase when $\gamma = 1$, we solve the following set of equations without any change of parameter:

$$f_1 = x_1x_2x_3 + 4x_2^3 + x_1^2 + x_2 - x_1x_3 - 8$$

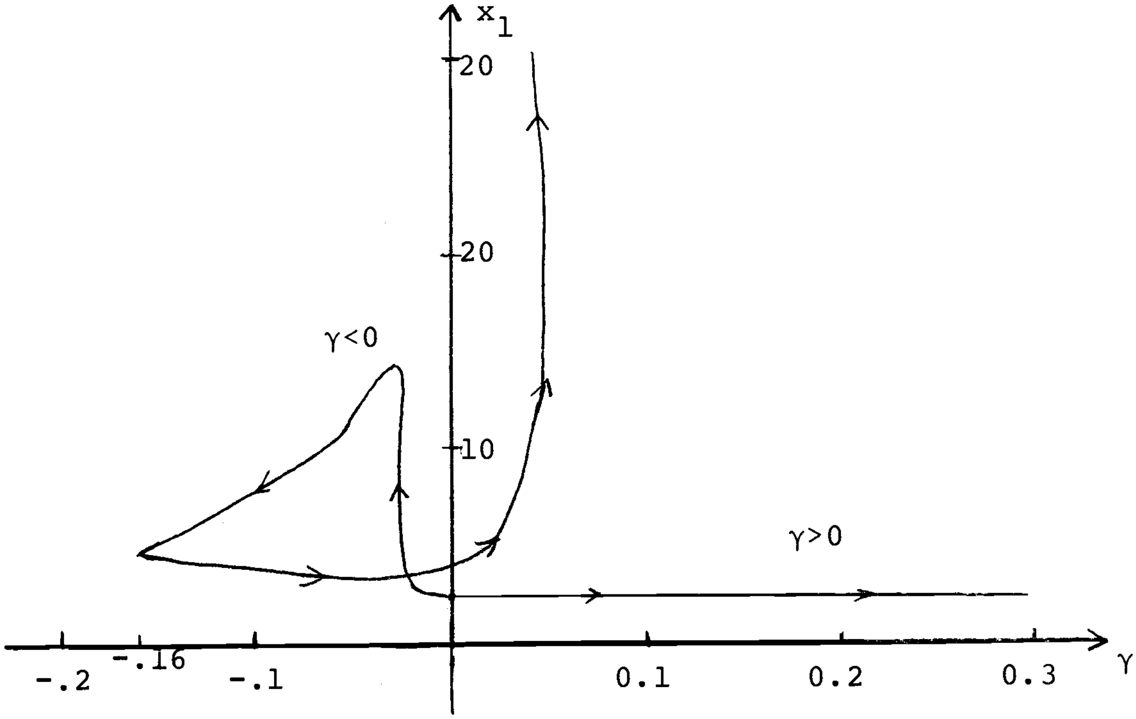
$$f_2 = (x_3x_2^2 + x_1x_3) - 2x_1 + x_2/2 + x_3^2 + 4$$

$$f_3 = (x_1^2 + x_2^2 + x_3^2)x_3 + x_1x_3 - x_2^2 + x_2x_3 - 1$$

	Modified Newton	Broyden
CPU time (sec.)	7.83	4.0
No. of iterations	36	34

The solution is (2.42649, .72091, .15863).

Now if we let γ go into the negative direction, there are several singular points. In this case we did not look for a final solution for any specific gamma, only the behavior of the components of the solution $\bar{x}(\gamma)$ are noted. But if we compare the time each of the two methods took to reach a value of gamma, say 0.041, the modified Newton took about 14 seconds compared with 11.5 seconds using Broyden's method. In both methods, there were four times the parameter ρ had to be used. γ would first decrease and then increase until it became positive and the curve is not traced further. The behavior of the solution may be best illustrated with a graph of one of the components of $\bar{x}$, we plot $\bar{x}_1(\gamma)$ below



Case 4. With $\mu^2 = 5$, no change of parameter is necessary.

	Modified Newton	Broyden
CPU time (sec.)	6.7	5.0
No. of iterations	11	18

Case 5. With $\mu^2 = 9.8$ in both methods, the parameter was changed twice.

	Modified Newton	Broyden
CPU time (sec.)	27.9	16.9
No. of iterations	352	388

Case 6. $\mu^2 = 15$.

There were also two places the parameter was needed to be changed for both methods to reach the solution at $\lambda = 1$.

	Modified Newton	Broyden
CPU time (sec.)	27.1	27.6
No. of iterations	364	597

This is the first test case that Broyden's method is inferior to the modified Newton's in terms of computing time, but the difference of .5 second in 27 seconds is not significant.

Conclusion

In view of the above experimental results, the method that does not require exact Jacobians being described in Chapter IV is more efficient than the modified Newton's method in terms of getting to the solution in less computing time. Another advantage is that it saves the user some time to punch cards for the Jacobian. The number of iterations needed to solve each intermediate problems vary from equations to equations. For the modified Newton's method, it appears three and four are the typical numbers of iterations; for Broyden's method it is five and six. It is then expected that the function $\overline{F}(\gamma,\overline{x})$ is evaluated more often in the latter case, and experimental results confirm this. (The only exception is case 3.) If it is impractical to calculate the Jacobian, then the Newton's iteration cannot be applied at all.

There are, of course, some other methods that do not use derivatives. (See Chapter II.) It is not clear which one of them is most efficient. Due to the limited amount of time available to complete this thesis, I regret that I can do no further comparison among these quasi-Newton methods.

Finally, some comments on the partial derivatives of $\overline{F}(\gamma,\overline{x})$ with respect to gamma. It is used in getting a prediction to the solution of $\overline{F}(\gamma_{i+1},\overline{x}) = \overline{0}$ from the solution of $\overline{F}(\gamma_i,\overline{x}) = \overline{0}$. It may be better, in terms of computational effort, not to use the prediction at all; or this derivative may be calculated by finite differences also. The latter case seems to be very plausible, for the Jacobian we used in predicting is never exact anyway. So getting an approximate partial derivative of $\overline{F}$ with respect to gamma would not affect the overall computing time appreciably. In either case it will save the user the trouble of getting the derivative with respect to gamma and we could then have a method which used no partial derivatives at all.

BIBLIOGRAPHY

1. Barnes, J. G. P. An algorithm for solving nonlinear equations based on the secant method. Computer Journal 8:66-72. 1964.

2. Broyden, C. G. A class of methods for solving nonlinear simultaneous equations. Mathematics of Computation 19:577-593. 1965.

3. _________. A new method of solving nonlinear simultaneous equations. Computer Journal 12:94-99. 1969.

4. Cullop, Paul Wayne. Automated rigorous solutions to nonlinear equations. Master's thesis. Corvallis, Oregon State University, 1973. 52 numb. leaves.

5. Davidenko, D. F. On a new method of numerical solution of system of nonlinear equations. Akademiia nauk SSSR Doklady (N.S.) 88:601-602. 1953.

6. Davis, Joel. The solution of nonlinear operator equations with critical points. Technical Report No. 25. Corvallis, Oregon State University, 1966. 69 numb. leaves.

7. Freudenstein, F. and Roth, B. Numerical solution of systems of nonlinear equations. Association for Computing Machinery Journal 10:550-556. 1963.

8. Householder, A. S. Principles of numerical analysis. New York, McGraw-Hill, 1953. 470 p.

9. Jeeves, T. A. Secant modification of Newton's method. Communication of the Association for Computing Machinery 8:9-10. 1958.

10. Powell, M. J. P. An efficient method for finding the minimum of a function of several variables without calculating derivatives. Computer Journal 7:155-162. 1964a.

11. _________. A method for minimizing a sum of squares of nonlinear functions without calculating derivatives. Computer Journal 7:303-307. 1964b.

12. Wolfe, Philip. The secant method for simultaneous nonlinear equations. Communications of the Association for Computing Machinery 2:12-13. 1959.