The University of Michigan
College of Engineering

Technical Report

Operator Approximation and Its Application to Root Finding and Minimization Problems

by

G. Scott Dixon, Jr.

Project Director
Elmer G. Gilbert
ORA Project 026450

Supported by
United States Air Force, Air Force Office of Scientific Research
Grant No. AFOSR-69-1767
Arlington, Virginia

June 1973

ACKNOWLEDGMENTS

The work represented by this dissertation has been made possible by the cooperation and guidance of a number of individuals. The author wishes to express special appreciation to Professor Elmer Gilbert who, as Chairman of the Doctoral Committee, provided expert advice and critcal review throughout the period of research. The author is also greatful to the other members of the Doctoral Committee for their interest in the work and for their helpful comments. The computer programs used in the numerical experiments were written by Alan Dohner. His expert help is appreciated. The manuscript itself was prepared by Chris Angell. It would be hard to imagine a more cooperative and efficient typist.

This research was supported by an NDEA Title IV Fellowship from the University of Michigan, by the United States Air Force under Grant AFOSR-69-1767 and by the Reliance Electric Company. Recognition is due to Dr. Edward Gilbert of the Reliance Electric Company for his patience and cooperation.

Finally, a special measure of gratitude is due to the author's wife, who not only offered encouragement and understanding, but also aided in the preparation of the manuscript.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF ILLUSTRATIONS

TABLE OF ILLUSTRATIONS (continued)

TABLE OF ILLUSTRATIONS (continued)

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

The research presented in this dissertation is concerned with two closely related problems which are encountered in computational mathematics. They are the determination of the unconstrained minimum of a function $f(x) : D \subset R^n \to R^1$ and the determination of a root of a function $F(x) : D \subset R^n \to R^n$; that is, an $x \in D$ such that $F(x) = 0$. As is usual R^n denotes the space of n-tuples of real numbers.

Although both of these problems have been extensively investigated over the years they deserve further research for at least two reasons. First, because many practical computations reduce to these problems, even a modest improvement in solution techniques can result in a large reduction in computational cost. Second, there exist similarities in certain current algorithms which suggests that a unified theory can be developed for these algorithms. Such a theory, besides clarifying existing algorithms, can be expected to yield insight into the development of new algorithms.

The functions treated will be assumed to have a Taylor series expansion. For instance, in the root finding problem

$$F(x) = F(x_0) + J(x_0)(x - x_0) + \text{h.o.t.} \qquad (1.1.1)$$

where $J(x_0)$ is the derivative of F at x_0 and h.o.t. represents the higher order terms. A similar expansion for the gradient of $f(x)$ pertains in the minimization problem. In all the algorithms to be considered it is implicitly assumed that, when the higher order terms are neglected, the Taylor series expansion gives rise to an estimate of the root of $F(x)$ given by

$$x = x_0 - J^{-1}(x_0)F(x_0). \qquad (1.1.2)$$

Rather than directly evaluating and inverting the matrix $J(x_0)$ as in the Newton method these algorithms estimate $J^{-1}(x_0)$ by determining a linear transformation which, under appropriate conditions, transforms a set of vectors $u_i \equiv F(x_i) - F(y_i)$ into the set of vectors $v_i \equiv x_i - y_i$, $i = 1,\ldots,m$, with $m \leq n$. From equation 1.1.1 we see that if the higher order terms are negligible, if $m = n$, and if $J(x_0)$ is nonsingular then this linear transformation is $J^{-1}(x_0)$. Notice that the existence and uniqueness of such a transformation is not guaranteed. This leads to convergence difficulties in many algorithms. In certain minimization algorithms it is necessary for the objective function to be quadratic, that is for the higher order terms to be identically zero, in order for the algorithm to determine such a linear transformation.

The set of vectors u_i and v_i, $i = 1,\ldots,m$, can be viewed as forming a data set containing information about the local character of the function $F(x)$. The algorithms which have been developed to date consider data sets which are formed either by refreshing the entire data set at each stage or by generating one data pair (u_i,v_i) at each stage and using data generated in the previous $m-1$ stages of the algorithm. Those algorithms which refresh the data set at each stage, such as certain secant methods for root finding, demonstrate nice theoretical terminal rate of convergence properties (ORT 70, section 11.3) but may not converge globally since at some stage a matrix may not exist which transforms the vectors u_i into the vectors v_i. Also these algorithms require a large amount of computation at each stage since the function must be evaluated $m = n$ times.

The sequential methods which generate only a single pair u_i and

v_i at each stage require far less computation. However, in many cases these methods cannot be shown to converge either globally or locally (ORT 70, theorem 11.3.5). Davidon's minimization algorithm is one sequential algorithm which can be shown to converge superlinearly if the objective function is twice differentiable and if the eigenvalues of $H(x)$, the second derivative function or Hessian of $f(x)$, are greater than some positive constant (POW 71). However, Davidon's algorithm computes a matrix which transforms the vectors u_i into the vectors v_i only when the objective function is quadratic. Several other sequential minimization algorithms have been proposed (MUR 70, FLE 70, PEA 69, BRO 70, 70a) which, when the objective function is quadratic, compute a matrix which transforms the u_i into the v_i. No extensive theoretical results have been presented for these algorithms when the objective function is not quadratic. Further, many of them exhibit a tendency to have convergence difficulties in practical problems.

The fundamental objectives of this work are: 1) the theoretical unification of certain algorithms which utilize sequences of data sets to estimate the linear transformation $J^{-1}(x_0)$ or $H^{-1}(x_0)$ and 2) the investigation of new algorithms for generating the sequence of data sets. Some of the material concerning arbitrary data sets may be of value in other problems which involve the estimation of a linear transformation.

1.2 Organization

In Chapter 2 data sets whose elements are pairs of vectors (u_i,v_i) are considered. These data sets are assumed to approximately characterize an unknown linear transformation. Although the data sets may arise in the root finding or minimization problem this is not necessary to the development. For a particular data set the family of approximations to

this unknown linear transformation is characterized in a useful way. When two data sets have common elements, results are obtained which characterize the family of approximations based on one data set in terms of the family of approximations based on the other data set. This leads to a result which allows the Davidon algorithm and other minimization algorithms to be viewed in a general framework.

In Chapter 3 the general results of Chapter 2 are applied to existing algorithms for root finding and minimization. In particular, the relationship between the secant algorithms for root finding and the material of Chapter 2 is considered. Also, using the results of Chapter 2 the properties of a class of minimization algorithms on a quadratic surface are investigated. This class contains new algorithms in addition to many existing algorithms.

In Chapter 4 an entirely new class of minimization algorithms is proposed. This class is constructed to have desirable convergence and rate of convergence properties while allowing considerable latitude in the choice of a specific algorithm within the general class. Theoretical results are presented concerning the behavior of these algorithms on nonquadratic functions.

In order to verify the theoretical results of Chapter 4 and to further establish the properties of the new class of algorithms, two particular algorithms were tested on a variety of objective functions. A version of Davidon's algorithm is used as a standard of comparison. The results of these computer runs are presented in Chapter 5.

Chapter 6 summarizes the significant results of the dissertation and relates the work to results obtained by other researchers.

CHAPTER 2

EMULATION AND APPROXIMATION

2.1 Notation and Definitions

In this chapter a general structure is presented which unifies many recent algorithms for function minimization and root finding. Although the unification of these algorithms has been the primary motivation for this work some of the material may be of value in the solution of other problems.

We will be concerned with a sequence of sets $\{S_i\}$, $i \in I_p$, where the index set I_p is some subset of I, the non-negative integers. The elements of the sets S_i are ordered, and consist of data obtained from some process such as the evaluation of a function or the gradient of a function. More specifically each set S_i contains a finite number $k(i) \geq 0$ of elements from the product space $R^m \times R^n$, $n,m > 0$. Each set S_i will be viewed as defining a relation on some elements of R^m and R^n. That is $u \in R^m$ is related to $v \in R^n$ under the relation S_i if and only if $(u,v) \in S_i$. When not empty the set S_i will also be viewed as a pair of matrices (U_i,V_i) whose j th columns $u_j \in R^m$ and $v_j \in R^n$ respectively make up the j th element (u_j,v_j) of S_i. Clearly, if $k(i) > 0$, U_i is $m \times k(i)$ and V_i is $n \times k(i)$. In most cases the ordering of the set S_i is not important. If the ordering of S_i is significant it will be specified. If S_i^x is any subset of a set S_i then the elements of S_i^x will retain the relative ordering assigned in S_i. This subset will also be viewed as a relation on $R^n \times R^m$ or as a pair of matrices (U_i^x,V_i^x) formed as just described for S_i.

If S_i^x is the empty set then the relation on $R^n \times R^m$ as well as the matrices (U_i^x,V_i^x) will be undefined.

The linear space of all linear transformations from R^m into R^n will be denoted by T. U^k and V^k will denote the linear spaces of all linear transformations from R^k into R^m and from R^k into R^n respectively. Of course, if elements of R^m, R^n and R^k are viewed as column vectors, elements of T, U^k and V^k have as their concrete representation matrices of the appropriate size and conversely any matrix of the appropriate size represents a linear transformation in one of these spaces. In some instances a matrix will be interpreted as an ordered set of elements in some vector space where the i th column of the matrix is the i th element of the set.

By the notation $R(A)$ we mean the range of the linear transformation A. By the notation $R^{\perp}(A)$ we mean the orthogonal complement of $R(A)$; that is, the set of $x \in R^n$ such that $\langle x,y \rangle = \sum_{i=1}^{n} x^i y^i = 0$ for any $y \in R(A)$ where x^i and y^i are the i th components of x and y respectively. The notation $N(A)$ will denote the null space of A and the notation $N^{\perp}(A)$ will denote the orthogonal complement of $N(A)$. By A' or x' we will mean the transpose of the matrix A or the vector x.

In the discussion of Chapter 1, section 1.1, it was seen that if the data set $S_i = (U_i,V_i)$ was derived from a root finding or minimization problem then a linear transformation that takes the columns of U_i into the columns of V_i is an estimate of the inverse Hessian or Jacobian matrix. The following definitions formalize this notion in a slightly more general context. It is assumed that $\{S_i\}$, $i \in I_p$, is a sequence of data sets with $k(i)$ elements in $R^m \times R^n$, that $\mathsf{V}^{k(i)}$ is equipped with some norm denoted by $\| \quad \|_{k(i)}$ and that A is some subset of T.

Definition 2.1.1 $A_i \in \mathsf{A}$ approximates the data set or relation $S_i = (U_i,V_i)$ over A iff

$$\|A_iU_i - V_i\|_{k(i)} = \min_{\bar{A} \in A} \|\bar{A}U_i - V_i\|_{k(i)}. \qquad (2.1.1)$$

If S_i is empty then any $A_i \in A$ approximates S_i over A.

Definition 2.1.2 $A_i \in A$ emulates the data set or relation $S_i = (U_i, V_i)$ over A iff

$$\|A_iU_i - V_i\|_{k(i)} = 0. \qquad (2.1.2)$$

If S_i is empty then any $A_i \in A$ emulates S_i over A.

Definition 2.1.3 The sequence $\{A_i\} \in A$ approximates (emulates) the sequence of data sets $\{S_i\}$ over A iff A_i approximates (emulates) S_i over A for all $i \in I_p$.

Definitions 2.1.1 and 2.1.2 can be given a geometric interpretation. The product space $R^m \times R^n$ is itself a linear space of dimension $m + n$ and a linear transformation $A_i \in T$ defines a subspace of $R^m \times R^n$ with dimension m whose elements are all the pairs $(u,v) \in R^m \times R^n$ where $v \in R^n$ is the image under A_i of $u \in R^m$ and where u is arbitrary. If $u_1,\ldots,u_m$ is any basis for R^m then the pairs $(u_1,A_iu_1),\ldots,(u_m,A_iu_m)$ are a basis for this subspace. A transformation A_i approximates S_i over A if the "vertical distance" between the elements of S_i and the subspace defined by A_i is minimized over all elements of A. If all elements of S_i lie in the subspace defined by A_i then A_i emulates S_i. By vertical distance we mean the set distance induced by the norm $\|\ \|_{k(i)}$ which, of course, weights all elements of S_i. If the matrix norm $\|\ \|_{k(i)}$ is invariant with respect to the ordering of the columns of the matrix then the ordering of the elements of S_i is of no consequence. However, if the norm is dependent on column ordering then this vertical distance, and the family of approximation to S_i over it will vary with the ordering of S_i. In all cases treated herein the norm $\|\ \|_{k(i)}$ is invarient with respect to column

ordering.

It may be that the data sets S_i are better approximated by an affine transformation than by a linear transformation. That is, there may exist a pair $(A_i,b_i) \in \mathsf{T} \times R^m$ such that

$$\|A_iU_i + b_ie - V_i\|_{k(i)} \leq \|AU_i - V_i\|_{k(i)} \qquad (2.1.3)$$

for all $A \in \mathsf{T}$ where e is the row vector $(1,\ldots,1)$. However, this approximation is only superficially different from the approximation being considered since

$$A_iU_i + b_ie = [A_i,b_i]\begin{bmatrix} U_i \\ e \end{bmatrix}. \qquad (2.1.4)$$

Here and in the rest of the text the square brackets denote a partitioning of a matrix. Thus the problem of determining an affine transformation such that $R^m \to R^n$ can be viewed as the problem of determining a linear transformation such that $R^{m+1} \to R^n$ where the matrix U_i is augmented by the row vector e and A_i is augmented by the column vector b_i.

In the particular case of the minimization and root finding problems if the elements of S_i are obtained by differencing then it is known, a priori, that -s is an element of S_i whenever s is an element of S_i. In this situation a linear transformation is the best approximation to S_i over T.

The existence of an A_i which approximates S_i over A is not guaranteed by the definition. However, the sets A considered herein will always contain an approximating A_i. Since the A_i which approximates S_i over A may not be unique, reference will be made to the set of A_i which approximate S_i over A. An A_i which emulates S_i over A may

not exist. If one does exist it may not be unique, in which case reference will be made to the set of A_i which emulate S_i over $\mathbb{A}$. Clearly, since an A_i which approximates (emulates) S_i over $\mathbb{A}$ may not be unique then neither is a sequence $\{A_i\}$ which approximates (emulates) the sequence $\{S_i\}$ over $\mathbb{A}$ necessarily unique. Notice that if the norm $\| \quad \|_{k(i)}$ is chosen to be independent of column ordering then the set of A_i which approximate S_i over $\mathbb{T}$ is invarient over all orderings of the set S_i. The set of A_i which emulate S_i over $\mathbb{T}$ is invariant over all orderings of S_i for any norm $\| \quad \|_{k(i)}$.

Extensive use will be made of generalized inverses of a matrix. At this point algebraic definitions of a generalized inverse and the pseudoinverse are given.

Definition 2.1.4 Let M be an arbitrary $n \times m$ matrix. A generalized inverse (g-inverse) of M (denoted by M^-) is any $m \times n$ matrix such that $MM^-M = M$.

Definition 2.1.5 Let M be an arbitrary $n \times m$ matrix. The pseudo-inverse (p-inverse) of M (denoted by M^+) is the $m \times n$ matrix such that $MM^+M = M$, $M^+MM^+ = M^+$, $(M^+M)' = M^+M$ and $(MM^+)' = MM^+$.

Some elementary properties of g-inverses are worthy of note.

1. If M^- is any g-inverse of M then for all $x \in R(M)$, $y = M^-x$ satisfies $My = x$.
2. If M^- is any g-inverse of M the operator MM^- projects vectors onto $R(M)$ and the operator $(I-MM^-)$ projects vectors onto $R^{\perp}(M)$. Therefore MM^- is the identity operator when restricted to $R(M)$ and $(I-MM^-)$ is the identity operator when restricted to $R^{\perp}(M)$.

3. If M^- is a g-inverse of M such that $(M^-M)' = M^-M$ then M^-M is the identity operator when restricted to $N^{\perp}(M)$.

4. If M^- is a g-inverse of M such that $(MM^-)' = MM^-$ then MM^- projects vectors orthogonally onto $R(M)$ and $(I-MM^-)$ projects vectors orthogonally onto $R^{\perp}(M)$.

5. If M^- is a g-inverse of M such that $M^-MM^- = M^-$ then $M^-x = 0$ for all $x \in R^{\perp}(M)$.

6. The p-inverse M^+ of M is unique, $y = M^+x$ minimizes $(x - My)'(x - My)$ and among all vectors which minimize $(x - My)'(x - My)$, $y = M^+x$ is the unique vector which minimizes $y'y$.

7. The following identities hold for the p-inverse:

$$M'MM^+ = M'$$

$$M'^+ = M^{+'}$$

$$M^{++} = M$$

$$M'M^{+'}M^+ = M^+$$

$$(M'M)^+ = M^+M^{+'}$$

A more complete treatment of generalized inverses is given in (PEN 55), (PEN 54), (RAO 71), (DES 63) and (ALB 72).

2.2 Approximations

All of the concrete results which follow involve a particular norm on $V^{k(i)}$ and on T which will be called a P-trace norm. If M is an arbitrary element of $V^{k(i)}$ or T then the P-trace norm of M is

given by

$$\|M\|_P = (\text{trace } (M'PM))^{1/2} \qquad (2.2.1)$$

where P is square, positive definite and symmetric. It is easily verified that this norm is independent of column ordering and therefore that the results are invariant over all orderings of the elements of S_i. The theorems are stated only for a particular element of the sequence $\{S_i\}$. The results can easily be restated for the entire sequence since by definition a result holds for the sequence $\{S_i\}$ iff it holds for each element of the sequence.

Theorem 2.2.1 If the norm on $V^{k(i)}$ is a P-trace norm and the data set $S_i = (U_i, V_i)$ is not empty then the set of approximations to S_i over T is the set of A_i which satisfy the matrix equation

$$A_i U_i U_i' = V_i U_i'. \qquad (2.2.2)$$

Proof: Under this norm

$$\|AU_i - V_i\|_P = (\text{trace } (AU_i - V_i)' P(AU_i - V_i))^{1/2} \qquad (2.2.3)$$

Since this function is non-negative, its minima are unchanged if we deal with $\|AU_i - V_i\|_P^2$. Clearly $\|AU_i - V_i\|_P^2$ is quadratic in the elements of A_i and differentiable with respect to the elements of A_i. Therefore $A_i \in T$ is an approximation to S_i over T iff

$$\frac{d}{dA}\|AU_i - V_i\|_P^2 \Big|_{A_i} = \frac{d}{dA} \text{trace } (AU_i - V_i)' P(AU_i - V_i) \Big|_{A_i} = 0 \qquad (2.2.4)$$

where $\frac{d}{dA} f(A)$ denotes the matrix whose ij th component is the partial derivative of f with respect to the ij th element of the matrix A. From the rules for differentiation of the trace function in Appendix I we have

$$\frac{d}{dA} \left(\| AU_i - V_i \|_P^2 \right) = 2PAU_iU_i' - 2PV_iU_i'. \qquad (2.2.5)$$

Since P is nonsingular $A_i \in T$ approximates S_i over T iff

$$A_iU_iU_i' - V_iU_i' = 0. \qquad (2.2.6)$$

Q.E.D.

Although it seems most reasonable to seek an approximation to S_i over the whole space T, the structure of the problem may lead one to restrict the class of admissible transformations. In the minimization problem the Hessian matrix is symmetric everywhere and is positive definite and symmetric at any isolated minimum. The following theorem deals with the approximation problem on the subspace T_s of symmetric matrices and on the subspace T_{ss} of skew-symmetric matrices.

Theorem 2.2.2 If the norm on $V^{k(i)}$ is a P-trace norm and the data set $S_i = (U_i, V_i)$ is not empty then the set of approximations to S_i over T_s is the set of symmetric A_i which satisfy the matrix equation

$$PA_iU_iU_i' + U_iU_i'A_iP = PV_iU_i' + U_iV_i'P. \qquad (2.2.7)$$

and the set of approximations to S_i over T_{ss} is the set of skew-symmetric A_i which satisfy the matrix equation

$$PA_iU_iU_i' - U_iU_i'A_iP = PV_iU_i - U_iV_i'P \qquad (2.2.8)$$

Proof: Again we may deal with the function $\| AU_i - V_i \|_P^2$ without affecting the location of any minima. Since $\| AU_i - V_i \|_P^2$ is quadratic in the components of A and differentiable with respect to the elements of A, and since T_s and T_{ss} are subspaces of T at any

minimum A_i of $\|AU_i - V_i\|_P^2$ restricted to Γ_s (or Γ_{ss}) it must be that

$$\text{trace}\left(\frac{d}{dA}\|AU_i - V_i\|^2 \Big|_{A_i}\right)' A = 0 \qquad (2.2.9)$$

for all A in Γ_s (or Γ_{ss}). This follows since at a minimum the gradient must be orthogonal to the subspace Γ_s (or Γ_{ss}). Again using the rules for differentiation of trace functions equation 2.2.9 is equivalent to requiring that

$$\text{trace}\,(A_i U_i U_i' - V_i U_i')' P\, A = 0 \qquad (2.2.10)$$

for all $A \in \Gamma_s$ (or Γ_{ss}). Let $\{A_{kj}\}$ be a basis for Γ_s (or Γ_{ss}). Because the trace is a linear operation, equation 2.2.10 becomes

$$\text{trace}\,(A_i U_i U_i' - V_i U_i')' P\, A_{kj} = 0 \qquad (2.2.11)$$

for all $A_{kj} \in \{A_{kj}\}$. Let $\{\Delta_{kj}\}$ be the set of matrices whose kj th element equals one and all other elements equal zero. Then a basis for Γ_s is $\{\Delta_{kj} + \Delta_{jk}\}$ and a basis for Γ_{ss} is $\{\Delta_{kj} - \Delta_{jk}\}$. Using the basis for Γ_s we have that at a minimum A_i of $\|AU_i - V_i\|_P^2$

$$\text{trace}\,(A_i U_i U_i' - V_i U_i')' P\, (\Delta_{kj} + \Delta_{jk}) = 0 \qquad (2.2.12)$$

or

$$\text{trace}\,(A_i U_i U_i' - V_i U_i')' P\, \Delta_{kj} + \text{trace}\,(A_i U_i U_i' - V_i U_i')' P\, \Delta_{jk} = 0. \qquad (2.2.13)$$

This is equivalent to requiring that $P(A_i U_i U_i' - V_i U_i')$ be skew-symmetric or that

$$P(A_i U_i U_i' - V_i U_i') + (A_i U_i U_i' - V_i U_i')P = 0. \qquad (2.2.14)$$

A similar argument for Γ_{ss} completes the proof. Q.E.D.

It is not easy to solve equations 2.2.7 or 2.2.8 either explicitly or computationally. In minimization algorithms perhaps an even more

useful set over which to seek an approximation is the set of positive definite or positive definite and symmetric matrices. If the matrices are constrained to be positive definite then it is guaranteed that the search direction will be downhill. A result parallel to theorems 2.2.1 and 2.2.2 has not been obtained for these classes of transformations. The solution of the approximation problem on other subsets of T is a topic which is worthy of further research. In this dissertation we pursue in depth only the approximations to S_i over T.

In order to explicitly characterize the family of approximations to S_i over T it is convenient to apply a result from the theory of g-inverses.

Theorem 2.2.3 The matrix equation $BXC = D$ has a solution for X iff

$$BB^{-}DC^{-}C = D \tag{2.2.15}$$

for any g-inverses B^{-} and C^{-} of B and C respectively. If a solution exists then the general solution is given by

$$X = B^{-}DC^{-} + Y - B^{-}BYCC^{-} \tag{2.2.16}$$

where Y is arbitrary and B^{-} and C^{-} are any g-inverses.

A proof of this result may be found in (RAO 72).

The following theorem is a consequence of theorem 2.2.3.

Theorem 2.2.4 If the norm on $V_{k(i)}$ is a P-trace norm and if the data set $S_i = (U_i, V_i)$ is not empty then the set of approximations to S_i over T is a non-empty linear manifold in T of dimension $n \times (m - r)$ where r is the rank of U_i. This manifold is given by the formula

$$A_i = V_iU_i^{+} + Y_i(I - U_iU_i^{+}) \tag{2.2.17}$$

where Y_i is n x m and arbitrary. Further

$$\|V_iU_i^{+}\|_{P_1} \leq \|V_iU_i^{+} + Y_i(I - U_iU_i^{+})\|_{P_1} \tag{2.2.18}$$

with equality only when $Y_i(I - U_iU_i^{+}) = 0$ for any P_1-trace norm or T.

Proof: Since U_i^+ is uniquely defined for any matrix, equation 2.2.17 yields a set of A_i which is not empty. The matrix $(I - U_iU_i^+)$ operating on row vectors in R^m projects elements of R^m orthogonally onto $R^1(U_i)$ which is a subspace of R^m with dimension $(m - r)$. Viewing Y_i as n arbitrary row elements of R^m the dimension of the subspace of T formed by $Y_i(I - U_iU_i^+)$, Y_i arbitrary, is $n \times (m - r)$ which establishes that equation 2.2.17 defines a manifold of dimension $n \times (m - r)$. From theorem 2.2.1, A_i approximates S_i over T iff

$$A_iU_iU_i' = V_iU_i'. \tag{2.2.19}$$

Applying theorem 2.2.3, with the p-inverse as the specific g-inverse, there exists a solution of equation 2.2.19 for A_i iff

$$V_iU_i'(U_iU_i')^+(U_iU_i') = V_iU_i'. \tag{2.2.20}$$

However, by property 7 of a p-inverse

$$V_iU_i'\,(U_iU_i')^+(U_iU_i') = V_iU_i'\,(U_i^{+\prime}U_i^+)\,U_iU_i' \tag{2.2.21}$$

$$= V_iU_i^+U_iU_i'$$

$$= V_iU_i'.$$

Therefore there always exists an approximation to S_i over T. Also by theorem 2.2.3 the set of approximations to S_i over T is given by

$$A_i = V_iU_i'\,(U_iU_i')^+ + Y_i(I - U_iU_i'\,(U_iU_i')^+) \tag{2.2.22}$$

$$= V_iU_i'U_i^{+\prime}U_i^+ + Y_i(I - U_iU_i'U_i^{+\prime}U_i^+)$$

$$= V_iU_i^+ + Y_i(I - U_iU_i^+)$$

where Y_i is arbitrary. It remains to be shown that $V_iU_i^+$ is the approximation to S_i over T of minimum norm under any P_1-trace norm on T.

We consider the square of the norm, noting that the minima are unchanged by the squaring operation. Substituting

$$\|V_i U_i^+ + Y_i(I - U_i U_i^+)\|_{P_1}^2 = \text{trace } (V_i U_i^+ + Y_i(I - U_i U_i^+))' P_1^{1/2} P_1^{1/2}$$

$$(V_i U_i^+ + Y_i(I - U_i U_i^+)) \qquad (2.2.23)$$

$$= \text{trace } P_1^{1/2}(V_i U_i^+ + Y_i(I - U_i U_i^+))(V_i U_i^+$$

$$+ Y_i(I - U_i U_i^+))' P_1^{1/2}$$

$$= \text{trace } P_1^{1/2}(V_i U_i^+)(V_i U_i^+)' P_1^{1/2} +$$

$$\text{trace } P_1^{1/2}(Y_i(I - U_i U_i^+))(Y_i(I - U_i U_i^+))' P_1^{1/2}$$

$$= \|V_i U_i^+\|_{P_1}^2 + \|Y_i(I - U_i U_i^+)\|_{P_1}^2$$

which proves the final assertion. Q.E.D.

Notice that direct application of theorem 2.2.3 yields the following representation for the family of approximations to S_i over $\top$.

$$A_i = V_i U_i (U_i U_i')^- + Y_i(I - U_i U_i'(U_i U_i')^-) \qquad (2.2.24)$$

where Y_i is arbitrary and $(U_i U_i')^-$ is any g-inverse of $U_i U_i'$. The simpler form of theorem 2.2.4 only holds if the g-inverse chosen is the p-inverse.

The following two theorems characterize the set of approximations to S_i over $\top$ in terms of a specific approximation $\bar{A}_i$.

Theorem 2.2.5 If $S_i = (U_i, V_i)$ is not empty and if $\bar{A}_i$ approximates S_i over $\top$ then the set of all approximations to S_i over $\top$ is given by

$$A_i = \bar{A}_i - W_i(I - U_iU_i^+) \tag{2.2.25}$$

where W_i is $n \times m$ and arbitrary.

Proof: If the solutions to equation 2.2.17 are viewed as any particular solution plus the set of solutions to the homogeneous equation the result follows. We have from theorem 2.2.4

$$\bar{A}_i = V_iU_i^+ + Y_i(I - U_iU_i^+) \tag{2.2.26}$$

for some matrix Y_i. Substituting this into the proposed solution for an arbitrary A_i gives

$$A_i = V_iU_i^+ + Y_i(I - U_iU_i^+) + W_i(I - U_iU_i^+) \tag{2.2.27}$$

$$= V_iU_i^+ + (Y_i + W_i)(I - U_iU_i^+)$$

where W_i is arbitrary. This is the set of all approximation to S_i over $\top$. Q.E.D.

Theorem 2.2.6 If $S_i = (U_i, V_i)$ is not empty and if $\bar{A}_i$ approximates S_i over $\top$ then the set of all Y_i in equation 2.2.17 which yield $\bar{A}_i$ is a linear manifold in $\top$ of dimension $n \times r$ where r is the rank of U_i. This manifold is given by

$$Y_i = \bar{A}_i + X_iU_iU_i^+ \tag{2.2.28}$$

where X_i is $n \times m$ and arbitrary.

Proof: We seek the set of Y_i which are solutions to the equation

$$Y_i(I - U_iU_i^+) = \bar{A}_i - V_iU_i^+. \tag{2.2.29}$$

By theorem 2.2.3 a solution for Y_i exists iff

$$(\bar{A}_i - V_iU_i^+)(I - U_iU_i^+)(I - U_iU_i^+)^+ = (\bar{A}_i - V_iU_i^+). \tag{2.2.30}$$

Because $(I - UU^+)$ is a projection, $(I - UU^+)^+ = (I - UU^+)$, see (PEN 54), and therefore this is equivalent to requiring that

$$\bar{A}_i U_i U_i^+ = V_i U_i^+. \tag{2.2.31}$$

Since $\bar{A}_i$ is an approximation to S_i over $\top$

$$\bar{A}_i U_i U_i' = V_i U_i'. \tag{2.2.32}$$

Postmultiplying by $U_i'^+ U_i^+$ and using property 7 of a p-inverse we have that

$$\bar{A}_i U_i U_i^+ = V_i U_i^+ \tag{2.2.33}$$

and therefore a solution for Y_i must exist. This simply verifies the result of theorem 2.2.4. Using theorem 2.2.3 the family of solutions for Y_i is given by

$$Y_i = (\bar{A}_i - V_i U_i^+)(I - U_i U_i^+)^+ + \bar{X}_i(I - (I - U_i U_i^+)(I - U_i U_i^+)^+) \tag{2.2.34}$$

where $\bar{X}_i$ is arbitrary. Because $(I - U_i U_i^+)$ is a projection operator

$$Y_i = \bar{A}_i + X_i U_i U_i^+ \tag{2.2.35}$$

where X_i is arbitrary.

The dimension of this manifold is clearly n x r since $U_i U_i^+$ projects the n rows of X_i onto $R(U_i)$. Q.E.D.

2.3 Emulations

If an A_i exists which emulates S_i over $\top$ then any emulating matrix is an approximating matrix and any approximating matrix is an emulating matrix. Although any emulation is also an approximation, the family of emulations can be analyzed further. The set of A_i which emulate S_i over $\top$ is the set of solutions to the matrix equation

$$AU_i = V_i. \tag{2.3.1}$$

These solutions are characterized by the following theorem.

Theorem 2.3.1 If $S_i = (U_i, V_i)$ is not empty the set of A_i which emulate S_i over $\top$ is not empty iff the following equivalent conditions are satisfied.

$$V_i U_i^- U_i = V_i \tag{2.3.2}$$

$$\text{Rank} \begin{bmatrix} U_i \\ V_i \end{bmatrix} = \text{Rank } V_i \qquad (2.3.3)$$

Further, if not empty the set of A_i which emulate S_i over $\top$ is given by

$$A_i = V_i U_i^- + Y_i(I - U_i U_i^-) \qquad (2.3.4)$$

where U_i^- is any g-inverse of U_i and Y_i is $n \times m$ and arbitrary.

Proof: Direct application of theorem 2.2.3 yields the equation 2.3.2 as well as the formula which characterizes the set of A_i which emulate S_i over $\top$. Equation 2.3.3 is simply a statement that a solution exists for A_i iff a solution exists for each row of A_i. This rank condition for vector equations is well known. Q.E.D.

Notice that since any emulation is also an approximation to S_i over $\top$ all of the results of section 2.2 remain valid for emulations. A particular condition for the existence of an A_i which emulates S_i over $\top$ is that U_i have independent columns.

The conditions of theorems 2.2.5 and 2.2.6 can be relaxed if one is dealing with a set S_i for which an emulation exists over $\top$.

Theorem 2.3.2 If $S_i = (U_i, V_i)$ is not empty and if $\bar{A}_i$ emulates S_i over $\top$ then the set of all A_i which emulate S_i over $\top$ is given by

$$A_i = \bar{A}_i - W_i(I - U_i U_i^-) \qquad (2.3.5)$$

where W_i is $n \times m$ and arbitrary and where U_i^- is any g-inverse of U_i.

Theorem 2.3.3 If $S_i = (U_i, V_i)$ is not empty and if $\bar{A}_i$ emulates S_i over $\top$ then the set of all Y_i in equation 2.3.4 which yield $\bar{A}_i$ is a linear manifold in $\top$ of dimension $n \times r$ where r is the rank of U_i. This manifold is given by

$$Y_i = \bar{A}_i + X_i U_i U_i^- \qquad (2.3.6)$$

where X_i is $n \times m$ and arbitrary and U_i^- is any g-inverse of U_i.

The proofs of both these results are entirely parallel to the proofs of theorems 2.2.5 and 2.2.6.

2.4 Sequential Characterization of Approximation and Emulation

We have thus far been concerned with the determination of an approximation to a particular set S_i, a member of a sequence of data sets. If two data sets S_i and S_j have no common elements, then the problems of determining an emulation or approximation to S_i and S_j must be treated independently. However, if S_i and S_j are not disjoint considerable simplification may be possible. It is assumed here that S_i and S_j are any two data sets. In the applications of Chapter 3, S_i will be the successor to S_j in some sequence of data sets. We will denote the elements of S_i also contained in S_j by S^p_{ij} and the elements of S_i not contained in S_j by S^q_{ij}. Elements of the set S_i will be assumed to be ordered so that all elements of S_i contained in S^p_{ij} precede all elements of S_i contained in S^q_{ij}. The elements of the sets S^p_{ij} and S^q_{ij} will retain the relative ordering assigned them in S_i. The ordering of S_j is of no consequence. Associated with the sets S^p_{ij} and S^q_{ij} are the pairs of matrices (U^p_{ij},V^p_{ij}) and (U^q_{ij},V^q_{ij}) which will be viewed as their concrete representation.

Before proceeding, two theorems from the theory of generalized inverses are presented which are useful in the computation of an emulation or approximation to S_i over $\top$.

Theorem 2.4.1 Let the $n \times m$ matrix A and the vector $a \in R^n$ be given. If $a \notin R(A)$ a g-inverse of $[A,a]$ is given by

$$[A,a]^- = \begin{bmatrix} A^-(I - ab') \\ b' \end{bmatrix} \qquad (2.4.1)$$

where A^- is any g-inverse of A and where $b' = a'(I - AA^-)'(I - AA^-)(a'(I - AA^-)'(I - AA^-)a)^{-1}$. If $a \in R(A)$, a g-inverse of $[A,a]$ is

given by

$$[A,a]^- = \begin{bmatrix} A^-(I - ab') \\ b' \end{bmatrix} \qquad (2.4.2)$$

where A^- is any g-inverse of A and where b' is arbitrary.

Further, if $a \notin R(A)$, the p-inverse of $[A,a]$ is given by

$$[A,a]^+ = \begin{bmatrix} A^+(I - ab') \\ b' \end{bmatrix} \qquad (2.4.3)$$

where A^+ is the p-inverse of A and where $b' = a'(I - AA^+)(a'(I - AA^+)a)^{-1}$.

If $a \in R(A)$ then the p-inverse of $[A,a]$ is given by

$$[A,a]^+ = \begin{bmatrix} A^+(I - ab') \\ b' \end{bmatrix} \qquad (2.4.4)$$

where A^+ is the p-inverse of A and where $b' = a'A^{+'}A^+(1 + a'A^{+'}A^+a)^{-1}$.

Theorem 2.4.2 Let the $n \times m$ matrix A and the vector $a \in R^n$ be given. Let $[A,a]^- = \begin{bmatrix} B \\ b \end{bmatrix}$ be any g-inverse of $[A,a]$. If $a \notin R(A)$ then B and $B(I - ab')$ are g-inverses of A. If $a \in R(A)$ and $a'b \neq 1$ then $B(I + a(1 - a'b)^{-1}b')$ is a g-inverse of A. Further, suppose $[A,a]^+ = \begin{bmatrix} B \\ b \end{bmatrix}$. If $a \notin R(A)$ then $B(I - b(b'b)^{-1}b')$ is the p-inverse of A. If $a \in R(A)$ and $a'b \neq 1$ then $B(I + a(1 - a'b)^{-1}b')$ is the p-inverse of A.

The results of both these theorems are presented in (RAO 71, section 3.6). Although no formal proof is given, the author states that the results follow by straightforward computation. The parts of theorem 2.4.1 and 2.4.2 dealing with p-inverses have also been published by (GRE 60) and (CLI 64) respectively. The results of theorem 2.4.2 can be formulated for the cases where the vector a is a matrix. This is done by Cline (CLI 64). Such generalizations are involved and of questionable computational value.

The usefulness of theorems 2.4.1 and 2.4.2 is easily demonstrated. Theorem 2.4.1 yields a direct method for obtaining a g-inverse, or the

p-inverse of any matrix. For instance, consider the $n \times m$ matrix $A = [a_1,\ldots,a_m]$ where m and n are arbitrary. It can be directly verified that the p-inverse of the single column matrix a_1 is given by $a_1'(a_1'a_1)^{-1}$. Denoting the matrix $[a_1,\ldots,a_{i-1}]$ by A_{i-1}, if some g-inverse of A_{i-1} is known, in order to apply theorem 2.4.1 to the matrix $[A_{i-1},a_i]$ it is only necessary to determine if $a_i \in R(A_{i-1})$. The matrix $A_{i-1}A_{i-1}^-$ can be used to check if $a_i \in R(A_{i-1})$ since this operator projects vectors onto $R(A_{i-1})$. Specifically, $a_i \in R(A_{i-1})$ iff $(I - A_{i-1}A_{i-1}^-)a_i = 0$.

To determine the family of approximations to a data set $S_i = (U_i,V_i)$ over T it is necessary to compute the p-inverse of U_i. If no information about the set S_i is available, repeated application of the procedure just outlined will yield the p-inverse of U_i. However, if the p-inverse of a matrix U_j is known, where U_j arises from a data set $S_j = (U_j,V_j)$ and S_{ij}^p is not empty then the p-inverse of U_j can be utilized to simplify the computation of the p-inverse of U_i. To accomplish this, notice that if P is any permutation matrix and U_j^+ is the p-inverse of U_j then the p-inverse of U_jP is given by $P'U_j^+$. Therefore, given the p-inverse of U_j under some ordering of S_j the p-inverse of U_j' can easily be found where U_j' arises from a data set $S_j' = (U_j',V_j')$ with the same elements as S_j but with a different ordering. If the ordering for S_j' is chosen such that the elements of S_j' also contained in S_{ij}^p precede all other elements of S_j' then, having computed the p-inverse of U_j', the computation specified by theorem 2.4.2 can be performed repeatedly until the p-inverse of U_{ij}^p is formed. Having determined the p-inverse of U_{ij}^p

the p-inverse of U_i can be computed by repeated application of theorem 2.4.1. The same process is valid for g-inverses of U_i other than the p-inverse. This procedure is discussed in greater detail in section 3.3 and is employed in the computations of Chapter 5.

The procedures discussed up to this point for calculating the p-inverse of U_i from the p-inverse of U_j are useful in determining the family of approximations to S_i over $\top$ whether or not these approximations also emulate S_i. If it is known a priori that an A_i exists which emulates S_i over $\top$ and an A_j exists which emulates S_j over $\top$ then further results can be obtained.

Theorem 2.4.3 Suppose $S_j = (U_j, V_j)$ and $S_i = (U_i, V_i)$ are not empty and that A_j emulates S_j over $\top$. If S_{ij}^p and S_{ij}^q are not empty then A_i emulate S_i over $\top$ iff

$$(A_i - A_j)U_{ij}^p = 0 \tag{2.4.5}$$

and

$$A_i U_{ij}^q = V_{ij}^q . \tag{2.4.6}$$

Further, if S_{ij}^q is empty then A_i emulates S_i over $\top$ iff

$$(A_i - A_j)U_{ij}^p = 0 . \tag{2.4.7}$$

Proof: By definition if S_{ij}^p and S_{ij}^q are not empty A_i emulates S_i over $\top$ iff

$$A_i U_{ij}^p = V_{ij}^p \tag{2.4.8}$$

and

$$A_i U_{ij}^q = V_{ij}^q \tag{2.4.9}$$

and if S_{ij}^q is empty A_i emulates S_i over $\top$ provided

$$A_i U_{ij}^p = V_{ij}^p \,. \qquad (2.4.10)$$

Since A_j emulates S_j over $\top$ and S_{ij}^p is contained in S_j

$$A_j U_{ij}^p = V_{ij}^p \,, \qquad (2.4.11)$$

from which the result of the theorem follows immediately. Q.E.D.

The case where S_{ij}^p and S_{ij}^q are both empty is excluded since this implies S_i is empty. The case S_{ij}^p empty produces no simplification since if S_{ij}^p is empty then S_i and S_j are disjoint.

Because of the desirability of determining an emulation or approximation to S_i given an emulation or approximation to S_j a particular choice for the matrix Y_i in equation 2.2.17 seems logical. If A_j approximates S_j over $\top$ then the choice $Y_i = A_j$ yields the following particular approximation to S_i over $\top$:

$$A_i = A_j + V_i U_i^+ - A_j U_i U_i^+ . \qquad (2.4.12)$$

If it is known that emulations to both S_i and S_j over $\top$ exist, a different but similar formula can be obtained.

Theorem 2.4.4 Suppose $S_i = (U_i, V_i)$ and $S_j = (U_j, V_j)$ are not empty and that A_j emulates S_j over $\top$. Suppose an A_i exists which emulates S_i over $\top$ and that S_{ij}^p and S_{ij}^q are not empty. If the elements of S_i

are ordered such that all elements of S_i contained in S_{ij}^p precede all elements of S_i contained in S_{ij}^q and if

$$U_i^- = [U_{ij}^p, U_{ij}^q]^- = \begin{bmatrix} U_i^1 \\ U_i^2 \end{bmatrix} \qquad (2.4.13)$$

is any g-inverse of U_i where U_i^1 has as many rows as U_{ij}^p has columns and U_i^2 has as many rows as U_{ij}^q has columns then

$$A_i = A_j + V_{ij}^q U_i^2 - A_j U_{ij}^q U_i^2 \qquad (2.4.14)$$

emulates S_i over $\top$. Further, if S_{ij}^q is empty then $A_i = A_j$ emulates S_i over $\top$.

Proof: We will verify directly that the equation $A_i U_i = V_i$ is satisfied. We have

$$A_i U_i = A_j U_i + V_{ij}^q U_i^2 U_i - A_j U_{ij}^q U_i^2 U_i \qquad (2.4.15)$$

$$= A_j U_i + \bar{A} U_{ij}^q U_i^2 U_i - A_j U_{ij}^q U_i^2 U_i$$

where $\bar{A}$ is some matrix which emulates S_i over $\top$. It follows from theorem 2.4.3 that

$$A_i U_i = A_j U_i + \bar{A} U_i U_i^- U_i - A_j U_i U_i^- U_i \qquad (2.4.16)$$

$$- \bar{A} U_{ij}^p U_i^1 U_i - A_j U_{ij}^p U_i^1 U_i$$

$$= A_j U_i + \bar{A} U_i - A_j U_i - (\bar{A} - A_j) U_{ij}^p U_i^1 U_i$$

$$= V_i.$$

The second result of the theorem is evident since if S_{ij}^q is empty then $S_i \subset S_j$ and if A_j emulates S_j over $\top$, A_i must also emulate S_i over $\top$. Q.E.D

Notice that theorem 2.3.2 can be applied to obtain the entire family of A_i which emulate S_i over $\top$ from the particular A_i given by either equation 2.4.12 or equation 2.4.14. Similarly theorem 2.3.3 can be applied to obtain the family of Y_i in equation 2.3.4 which yield the particular A_i of equation 2.4.12 or 2.4.14.

Theorem 2.4.4 is interesting because equation 2.4.14 is quite similar in form to some of the recursive formulas used in minimization algorithms and because equation 2.4.14 promises to reduce the number of calculations required to determine an A_i which emulates S_i over $\top$. However, the problem of determining U_i^2 without first determining a g-inverse of U_i has not been solved. Another formula for a specific emulation which circumvents this difficulty will now be obtained. Although different from the formula of theorem 2.4.4 the result is very similar. As will be shown in Chapter 3, specific choices for the matrices Z_i^1 and Z_i^2 in the following theorem yield generalizations of the recursive formulas used in several minimization algorithms.

Theorem 2.4.5 Suppose $S_i = (U_i, V_i)$ and $S_j = (U_j, V_j)$ are not empty and that A_j emulates S_j over $\top$. Suppose S_{ij}^p and S_{ij}^q are not empty and that the elements of S_i are ordered such that all elements of S_i contained in S_{ij}^p precede all elements of S_i contained in S_{ij}^q. If there exist matrices Z_i^1 $(r \times n)$ and Z_i^2 $(k \times n)$ for some $r,k > 0$ which satisfy

$$Z_i^1 U_{ij}^p = 0 \tag{2.4.17}$$

$$Z_i^2 U_{ij}^p = 0 \tag{2.4.18}$$

$$V_{ij}^q (Z_i^1 U_{ij}^q)^- (Z_i^1 U_{ij}^q) = V_{ij}^q \tag{2.4.19}$$

$$A_j U^q_{ij} (Z^2_i U^q_{ij})^- (Z^2_i U^q_{ij}) = A_j U^q_{ij} \qquad (2.4.20)$$

where $(Z^1_i U^q_{ij})^-$ is any g-inverse of $Z^1_i U^q_{ij}$ and $(Z^2_i U^q_{ij})^-$ is any g-inverse of $Z^2_i U^q_{ij}$, then

$$A_i = A_j + V^q_{ij} (Z^1_i U^q_{ij})^- Z^1_i - A_j U^q_{ij} (Z^2_i V^q_{ij})^- Z^2_i \qquad (2.4.21)$$

emulates S_i over $\top$. Further, if S^p_{ij} is empty and there exist matrices Z^1_i and Z^2_i such that equations 2.4.19 and 2.4.20 are satisfied then A_i given by equation 2.4.21 emulates S_i over $\top$.

Proof: The proof verifies directly that the equation $A_i U_i = V_i$ is satisfied. If S^p_{ij} is not empty

$$A_i U_i = A_j U_i + V^q_{ij} (Z^1_i U^q_{ij})^- Z^1_i [U^p_{ij}, U^q_{ij}] \qquad (2.4.22)$$

$$-A_j U^q_{ij} (Z^2_i U^q_{ij})^- Z^2_i [U^p_{ij}, U^q_{ij}]$$

$$= A_j U_i + [0, V^q_{ij}] - [0, A_j U^q_{ij}]$$

$$= [A_j U^p_{ij}, A_j U^q_{ij} + V^q_{ij} - A_j U^q_{ij}]$$

$$= [V^p_{ij}, V^q_{ij}]$$

$$= V_i.$$

If S^p_{ij} is empty the proof follows the same argument. Q.E.D.

It is interesting to consider when certain of the four conditions of theorem 2.4.5 are necessary. Suppose S_i, S_j, S^p_{ij}, and S^q_{ij} are not empty and that A_j emulates S_j over $\top$. If either equation 2.4.17 or equation 2.4.18 is satisfied then it is necessary that the other condition be satisfied if A_i given by equation 2.4.21 is to emulate S_i over $\top$. Similarly, if either one of equations 2.4.19 and 2.4.20 are satisfied

then it is necessary that the other condition be satisfied if A_i given by equation 2.4.21 is to emulate S_i over $\top$. Also of interest is the case where $Z_i^1 = Z_i^2$. If $Z_i^1 = Z_i^2$ then equations 2.4.17 and 2.4.18 are equivalent and equations 2.4.19 and 2.4.20 may be restated as follows:

$$(V_{ij}^q - A_j V_{ij}^q)(Z_i^1 U_{ij}^q)^- (Z_i^1 U_{ij}^q) = (V_{ij}^q - A_j U_{ij}^q). \quad (2.4.23)$$

Also in this case it is necessary that equation 2.4.17 (or 2.4.18) and equation 2.4.23 be satisfied if A_i is to emulate S_i over $\top$.

It is possible to state equations 2.4.19 and 2.4.20 in a slightly different manner as the following lemma shows.

Lemma 2.4.6 Let C^- be any g-inverse of the $n \times m$ matrix C. The equation $DC^-C = D$ is satisfied iff there exists a solution for X of the matrix equation $XC = D$ or equivalently iff $\text{Rank } C = \text{Rank} \begin{bmatrix} C \\ D \end{bmatrix}$.

The proof of lemma 2.4.6 follows immediately from theorem 2.2.3 when B is chosen to be the identity. Equations 2.4.19 and 2.4.20 therefore are equivalent to the existence of a solution for X of the equations

$$XZ_i^1 U_{ij}^q = V_{ij}^q \quad (2.4.24)$$

and

$$XZ_j^2 U_{ij}^q = A_j U_{ij}^q \quad (2.4.25)$$

respectively. Using the rank condition, equations 2.4.19 and 2.4.20 are equivalent to requiring that

$$\text{Rank}(Z_i^1 U_{ij}^q) = \text{Rank} \begin{bmatrix} Z_i^1 U_{ij}^q \\ V_{ij}^q \end{bmatrix} \quad (2.4.26)$$

and

$$\text{Rank}(Z_i^2 U_{ij}^q) = \text{Rank} \begin{bmatrix} Z_i^2 U_{ij}^q \\ A_j U_{ij}^q \end{bmatrix} \quad (2.4.27)$$

Clearly, these rank conditions will always be satisfied provided $Z_i^1 U_{ij}^q$

and $Z_i^2 U_{ij}^q$ have independent columns.

The question of when matrices Z_i^1 and Z_i^2 exist which satisfy the conditions of theorem 2.4.5 is partially answered by the following theorem.

Theorem 2.4.7 Suppose $S_i = (U_i, V_i)$ and $S_j = (U_j, V_j)$ are not empty, that A_j emulates S_j over $\top$ and that there exists an A_i which emulates S_i over $\top$. Suppose S_{ij}^p and S_{ij}^q are not empty and that the elements of S_i are ordered such that all elements of S_i contained in S_{ij}^p precede all elements of S_i contained in S_{ij}^q. If $R(U_{ij}^p)$ and $R(U_{ij}^q)$ are disjoint then there exist matrices Z_i^1 and Z_i^2, given by

$$Z_i^1 = [0, U_{ij}^q] U_i^- + Y_i^1 (I - U_i U_i^-) \qquad (2.4.28)$$

$$Z_i^2 = [0, U_{ij}^q] U_i^- + Y_i^2 (I - U_i U_i^-) \qquad (2.4.29)$$

where U_i^- is any g-inverse of U_i and Y_i^1 and Y_i^2 are arbitrary, which satisfy equations 2.4.17, 2.4.18, 2.4.19, and 2.4.20. Further, if S_{ij}^p is empty then there always exist matrices Z_i^1 and Z_i^2, given by

$$Z_i^1 = U_i U_i^- + Y_i^1 (I - U_i U_i^-) \qquad (2.4.30)$$

$$Z_i^2 = U_i U_i^- + Y_i^2 (I - U_i U_i^-) \qquad (2.4.31)$$

where U_i^- is any g-inverse of U_i and Y_i^1 and Y_i^2 are arbitrary, which satisfy equations 2.4.19 and 2.4.20.

Proof: If S_{ij}^p is not empty and $R(U_{ij}^p)$ and $R(U_{ij}^q)$ are disjoint then any dependent column of $U_i = [U_{ij}^p, U_{ij}^q]$ which is a column of U_{ij}^p can be written as a linear combination of columns of U_{ij}^p and any dependent column of $[U_{ij}^p, U_{ij}^q]$ which is a column of U_{ij}^q can be written as a linear combination of columns of U_i^q. Therefore

$$\text{Rank } [U_{ij}^p, U_{ij}^q] = \text{Rank} \begin{bmatrix} U_{ij}^p, U_{ij}^q \\ 0, U_{ij}^q \end{bmatrix} \qquad (2.4.32)$$

since dependent columns of the augmented matrix

$$\begin{bmatrix} U_{ij}^{p}, U_{ij}^{q} \\ 0, U_{ij}^{q} \end{bmatrix}$$

have the same representation as dependent columns of the matrix $[U_{ij}^{p}, U_{ij}^{q}]$. This rank condition is satisfied iff there exists a solution for Z of the matrix equation

$$Z[U_{ij}^{p}, U_{ij}^{q}] = [0, U_{ij}^{q}] \qquad (2.4.33)$$

From theorem 2.3.3 the family of all such solutions is given by

$$Z = [0, U_{ij}^{q}]U_i^{-} + Y[I - U_iU_i^{-}] \qquad (2.4.34)$$

where U_i^{-} is any g-inverse of U_i and Y is arbitrary. Choosing Z_i^1 and Z_i^2 as any such solution it can be directly verified that conditions 2.4.17, 2.4.18, and 2.4.20 are satisfied. Because it was assumed that an A_i exists which emulate S_i over $\top$ there exists a matrix $\bar{A}$ such that $\bar{A}U_{ij}^{q} = V_{ij}^{q}$. Substitution of this equation into condition 2.4.19 verifies that this condition must also be satisfied for such a choice of Z_i^1 and Z_i^2. Q.E.D.

Notice that it is not necessary that the matrices Z_i^1 and Z_i^2 satisfy the matrix equation 2.4.33 in order to satisfy equations 2.4.17 through 2.4.20.

Because of the similarity between equation 2.4.14 of theorem 2.4.4 and equation 2.4.21 of theorem 2.4.5 it is natural to ask how the two theorems are related. Observe that equation 2.4.14 admits only a single g-inverse $U_i^{-} = \begin{bmatrix} U_i^1 \\ U_i^2 \end{bmatrix}$ of U_i. That is to say the proof of theorem 2.4.4 does not go through if the U_i^2 postmultiplying V_{ij}^{q} and the U_i^2 postmultiplying $A_jU_{ij}^{q}$ are the bottom halves of different g-inverses of U_i. One would expect that under certain conditions the matrices $(Z_i^1U_{ij}^{q})^{-}Z_i^1$ and $(Z_i^2U_{ij}^{q})^{-}Z_i^2$ of equation 2.4.21 are the bottom halves of some g-inverse of U_i and that $(Z_i^1U_{ij}^{q})^{-}Z_i^1 = (Z_i^2U_{ij}^{q})^{-}Z_i^2$. If this is the case

then the equation 2.4.21 is of the same form as equation 2.4.14. Clearly, if $Z_i^1 = Z_i^2$ then $(Z_i^1 U_{ij}^q)^- Z_i^1 = (Z_i^2 U_{ij}^q)^- Z_i^2$. Thus far no more general condition has been obtained under which this equality holds. The question of when the matrix $(ZU_{ij}^q)^- Z$ is the bottom half of some g-inverse of U_i is answered by the following theorem.

Theorem 2.4.8 If $U_i = [U_{ij}^p, U_{ij}^q]$, $R(U_{ij}^p)$ and $R(U_{ij}^q)$ are disjoint and the matrix Z satisfies

$$ZU_{ij}^p = 0 \tag{2.4.35}$$

$$U_{ij}^q (ZU_{ij}^q)^- (ZU_{ij}^q) = U_{ij}^q \tag{2.4.36}$$

then there exists a family of g-inverses of U_i of the form

$$U_i^- = \begin{bmatrix} X \\ (ZU_{ij}^q)^- Z \end{bmatrix} \tag{2.4.37}$$

with X given by

$$X = (U_{ij}^p)^- [U_{ij}^p, 0] U_i^- + Y - (U_{ij}^p)^- U_i^p Y U_i U_i^- \tag{2.4.38}$$

where $(U_{ij}^p)^-$ and U_i^- are any g-inverses of U_{ij}^p and U_i respectively and where Y is arbitrary.

Proof: From the definition of a g-inverse we seek matrices X which satisfy

$$[U_{ij}^p, U_{ij}^q] \begin{bmatrix} X \\ (ZU_{ij}^q)^- Z \end{bmatrix} [U_{ij}^p, U_{ij}^q] = [U_{ij}^p U_{ij}^q]. \tag{2.4.39}$$

Expanding this expression gives

$$U_{ij}^p X U_{ij}^p + U_{ij}^q (ZU_{ij}^q)^- Z U_{ij}^p = U_{ij}^p \tag{2.4.40}$$

and

$$U_{ij}^p X U_{ij}^q + U_{ij}^q (ZU_{ij}^q)^- Z U_{ij}^q = U_{ij}^q. \tag{2.4.41}$$

Because of equations 2.4.35 and 2.4.36 this reduces to

$$U_{ij}^p X U_{ij}^q = U_{ij}^p, \tag{2.4.42}$$

$$U_{ij}^p X U_{ij}^q = 0 \tag{2.4.43}$$

or

$$U_{ij}^{p} X U_i = [U_{ij}^{p}, 0]. \tag{2.4.44}$$

From theorem 2.2.3 there exists a solution for X iff

$$U_{ij}^{p}(U_{ij}^{p})^{-} [U_{ij}^{p}, 0] U_i^{-} U_i = [U_{ij}^{p}, 0] \tag{2.4.45}$$

or

$$[U_{ij}^{p}, 0] U_i U_i^{-} = [U_{ij}^{p}, 0] \tag{2.4.46}$$

By lemma 2.4.6 this is equivalent to requiring that

$$\text{Rank } [U_{ij}^{p} U_{ij}^{q}] = \text{Rank} \begin{bmatrix} U_{ij}^{p} U_{ij}^{q} \\ 0 \quad U_{ij}^{q} \end{bmatrix} \tag{2.4.47}$$

Because $R(U_{ij}^{p})$ and $R(U_{ij}^{q})$ are disjoint this rank condition must be satisfied by the same argument that was employed in the proof of theorem 2.4.7. Again, employing theorem 2.2.3 the family of solutions for X is given by equation 2.4.38. Q.E.D.

2.5 Summary

In this chapter we have considered in detail the problem of approximating a set of data pairs by a linear transformation. In theorem 2.2.4 the family of approximations to a data set $S_i = (U_i, V_i)$ over $\top$ was characterized using the p-inverse. If an approximation A_i to a data set S_i transforms U_i into V_i then we say that A_i emulates S_i. In this case a less restrictive characterization of the family of emulations of a data set S_i is possible. This characterization, involving an arbitrary g-inverse, is presented in theorem 2.3.1.

If two data sets S_i and S_j have common elements it is possible to obtain further results about emulations or approximations to S_i over $\top$ when an emulation or approximation to S_j over $\top$ is given. In the first part of section 2.4 we consider how this can be accomplished using the results of theorems 2.4.1 and 2.4.2 which are standard results from the

theory of g-inverses. If an A_j is given which emulates S_j over T and if it is known that an A_i exists which emulates S_i over T then it is possible to state several formulas which determine a matrix A_i that emulates S_i over T. These are presented in theorems 2.4.3, 2.4.4, and 2.4.5. The conditions of theorem 2.4.5 are particularly important since they can be utilized to derive new recursion formulas for minimization algorithms as well as most of the currently known formulas. The remainder of Chapter 2 is concerned with the existence of matrices which satisfy the conditions of theorem 2.4.5 and with the relationship between theorem 2.4.5 and theorem 2.4.3.

The material of Chapter 2 suggests two other directions for research which have not been pursued in depth. In the minimization problem it would be advantageous to consider approximations over the set of positive definite symmetric linear transformations. A preliminary result toward that end was presented in theorem 2.2.2. An analysis of different norms on $V^{k(i)}$, such as the operator norm defined for $A \in V^{k(i)}$ by $\|A\|_1 = \max_{1\leq j\leq N} \sum_{i=1}^{m} |a_{ij}|$, may lead to computationally useful results.

CHAPTER 3

ROOT FINDING AND MINIMIZATION ALGORITHMS

3.1 Introduction and Notation

We are concerned in this chapter with the relationship between the emulation and approximation problems considered in Chapter 2 and some of the algorithms which have been proposed for function minimization and root finding. In the minimization problem an objective function $f(x) : D \subset R^n \to R^1$ is to be minimized over the interior of its domain denoted by $\underline{D}$. It will be assumed that $f(x)$ has a derivative $g(x) : \underline{D} \subset R^n \to R^n$ and a second derivative (Hessian) $H(x) : \underline{D} \subset R^n \to R^{n2}$. In the root finding problem an $x \in \underline{D}$ is sought such that $F(x) = 0$ where $F(x) : D \subset R^n \to R^m$. It will be assumed that $F(x)$ has a derivative on $\underline{D}$ denoted by $J(x) : D \subset R^n \to R^{nm}$.

Although these two problems are distinct they are closely related. If $f(x)$ is strictly convex on $\underline{D}$ then the problem of determining the unique minimum of $f(x)$ on $\underline{D}$ is equivalent to the problem of determining the unique root of $g(x)$ on $\underline{D}$. Even if $f(x)$ is not strictly convex on $\underline{D}$ the minimization of $f(x)$ is often approached by seeking roots of $g(x)$ on $\underline{D}$ since it must be that, at a minimum of $f(x)$ on $\underline{D}$, $g(x) = 0$. If a minimum of $f(x)$ is being sought by solving for roots of $g(x)$, then additional structure is present in the root finding problem since $H(x)$, the derivative of $g(x)$, is square and symmetric on $\underline{D}$. Further, at a minimum of $f(x)$ on $\underline{D}$ $H(x)$ is positive semi-definite.

On the other hand the problem of finding a root of $F(x)$ on $\underline{D}$ can be transformed into a minimization problem by considering a function $h(y) : R^m \to [0,\infty)$ such that $h(0) = 0$ and $h(y) > 0$ for all $y \neq 0$. The problem of determining a root of $F(x)$ on $\underline{D}$ is equivalent to the

problem of determining a minimum of $f(x) = h(F(x))$ on $\underline{D}$. The choice $h(y) = y'y$ is particularly appealing since the derivative of $F(x)'F(x)$ is $2J(x)F(x)$. The roots of $F(x)$ may not be unique, and if $m > n$, a root of $F(x)$ will usually not exist. In any case the minimization of $F'(x)F(x)$ yields a point in R^n at which $F(x)$ has minimum Euclidean norm.

3.2 Newton's Method

It is of interest to state what constitutes an algorithm for the solution of the problems under consideration. An algorithm is simply a rule which utilizes computable expressions to generate a sequence of points $\{x_k\} \subset \underline{D}$ which converge to a solution of the problem.

In most algorithms of practical interest, given an initial point $x_0 \in \underline{D}$, the rule can be expressed as a recursion of the form

$$x_{k+1} = G_k(x_k, x_{k-1}, \ldots, x_0) \qquad (3.2.1)$$

where $\{G_k\}$ is some sequence of operators whose k th element G_k is derived from the points $x_0, \ldots, x_k$ and maps the points $x_0, \ldots, x_k$ into the point x_{k+1}.

Newton's method is the foundation for all the algorithms considered in this chapter. It is based on the truncated power series expansion

$$F(x) \tilde{=} F(x_k) + J(x_k)(x - x_k). \qquad (3.2.2)$$

An approximation to a root of $F(x)$ is arrived at by assuming that the approximate equation 3.2.2 is valid in a neighborhood containing a root of $F(x)$. If $J(x_k)$ is nonsingular this leads to the iteration

$$x_{k+1} = x_k - J^{-1}(x_k)F(x_k). \qquad (3.2.3)$$

If a minimum of a function $f(x)$ is sought, Newton's algorithm can be applied to $g(x)$, the derivative of $f(x)$, in which case the iteration becomes

$$x_{k+1} = x_k - H^{-1}(x_k)g(x_k). \qquad (3.2.4)$$

There are several drawbacks to the Newton procedure. For even simple functions $F(x)$ (or $f(x)$) there is no guarantee that the Jacobian (or Hessian function) is nonsingular on $\underline{D}$. Hence, the sequence $\{x_k\}$ may not be defined. If the sequence $\{x_k\}$ is defined and remains in $\underline{D}$ there is no guarantee that, starting at an arbitary $x_0 \in \underline{D}$, the sequence $\{x_k\}$ will converge to a solution. Fortunately, certain convergence results can be obtained. Let $\hat{x} \in \underline{D}$ be a solution point and assume that $J(\hat{x})$ (or $H(\hat{x})$) is nonsingular. Then for $x_0 \in \underline{D}$ and $\hat{x} - x_0$ sufficiently small, the sequence $\{x_k\}$ generated by Newton's method converges to $\hat{x}$ and furthermore the rate of convergence is quadratic (ORT 70, theorem 10.2.2). That is

$$\lim_{k \to \infty} \frac{\|x_{k+1} - \hat{x}\|}{\|x_k - \hat{x}\|^2} = C < \infty. \qquad (3.2.5)$$

In terms of the general recursion 3.2.1, the operator G_k of Newton's method is derived only from the point x_k and operates only on the point x_k to generate the point x_{k+1}. While this makes the recursion simple in form, the calculation of J (or H) and its inverse may pose great difficulties in practical problems. Thus a large number of algorithms have been developed which utilize a set of function (or gradient) evaluations to estimate the inverse of J (or H) at x_k. It is these algorithms with which we are concerned in this chapter. Newton's method itself cannot be put into the context of Chapter 2.

3.3 Secant Methods

A large class of algorithms, known as secant methods, has been developed for determining the root of a function $F(x) : D \subset R^n \to R^n$. In order to describe the secant methods it is useful to introduce the notion

of points in general position.

Definition 3.3.1 The $n + 1$ points $x_0,\ldots,x_n \in R^n$ are in general position if the vectors $x_0 - x_i$, $i = 1,\ldots,n$, are linearly independent.

The notion of points in general position leads to the following result.

Theorem 3.3.2 Let $x_0,\ldots,x_n \in R^n$ and $y_0,\ldots,y_n \in R^n$ be given. Then there exists a unique affine transformation $L(x) = Ax + a$ such that $L(x_i) = y_i$, $i = 0,\ldots,n$, iff $x_0,\ldots,x_n$ are in general position. Further A is nonsingular iff $y_0,\ldots,y_n$ are in general position.

The proof of this theorem is found in (ORT 70, p. 192).

If the points $y_0,\ldots,y_n$ are taken to be evaluations of the function $F(x)$ at the points $x_0,\ldots,x_n$, then the basis for all secant algorithms is contained in the following definition.

Definition 3.3.3 If $x_0,\ldots,x_n \in D \subset R^n$ and $F(x_0),\ldots,F(x_n) \in R^n$ are in general position and if A and a satisfy

$$a + Ax_i = F(x_i), \quad i = 0,\ldots,n, \tag{3.3.1}$$

then the point

$$x_s = A^{-1}a \tag{3.3.2}$$

is the "basic secant approximation" with respect to $x_0,\ldots,x_n$.

Theorem 3.3.2 insures that the basic secant approximation is defined and unique provided $x_0,\ldots,x_n$ and $F(x_0),\ldots,F(x_n)$ are in general position. In all secant algorithms a secant step (a basic secant approximation) is computed by determining an affine function that has the same value as $F(x)$ at the points $x_0,\ldots,x_n$ and choosing the root of the affine function as the estimate of the root of $F(x)$. If the problem of determining a secant step is attacked directly it is necessary to first obtain the affine transformation $Ax + a$ and then to solve the equation $Ax + a = 0$. The following result, known as the

Newton formulation of the secant step, simplifies the calculation of a secant step and at the same time makes apparent the relationship between the basic secant approximation and the material of Chapter 2.

Theorem 3.3.4 If $x_0,\ldots,x_n$ and $F(x_0),\ldots,F(x_n)$ are in general position then the basic secant approximation with respect to $x_0,\ldots,x_n$ is given by

$$x_s = x_0 - VU^{-1}F(x_0) \tag{3.3.3}$$

where

$$V = [x_1 - x_0,\ldots,x_n - x_0]$$

and

$$U = [F(x_1) - F(x_0),\ldots,F(x_n) - F(x_0)].$$

Proof: For any column v_i of V

$$Av_i = Ax_i - Ax_0 = F(x_i) - a - F(x_0) + a = F(x_i) - F(x_0)$$

it follows that $AV = U$. The basic secant approximation must satisfy the equation

$$Ax_s + a = 0. \tag{3.3.4}$$

Substituting yields

$$A(x_0 - VU^{-1}F(x_0)) + a = F(x_0) - AVU^{-1}F(x_0) \tag{3.3.5}$$

$$= F(x_0) - F(x_0)$$

$$= 0. \qquad \text{Q.E.D.}$$

Many, although not all, of the secant methods make use of the Newton formulation. In order to completely specify a secant algorithm it is necessary to define a procedure for the generation of the points $x_0,\ldots,x_n$ at each stage k of the algorithm. A formulation of one secant method with a particular procedure for generating the points $x_0,\ldots,x_n$

is as follows.

Algorithm 3.3.5 Assume the two initial points $x_{-1}, x_0 \in \underline{D}$ are given and set $k = 0$.

1. If $F(x_k) = 0$ stop.
2. Set $x_{kj} = x_k + (x^j_{k-1} - x^j_k)e_j$, $j = 1,\ldots,n$, and set $x_{k0} = x_k$ where x^j_k is the j th component of x_k and where e_j is the vector whose components are zero except for the j th component which equals one.
3. Set $x_{k+1} = x_k - V_k U_k^{-1} F(x_k)$ where $V_k = [x_{k1} - x_{k0},\ldots,x_{kn} - x_{k0}]$ and $U_k = [F(x_{k1}) - F(x_{k0}),\ldots,F(x_{kn}) - F(x_{k0})]$.
4. Set $k = k+1$ and return to step 1.

Many other specification procedures have been proposed for generating the auxiliary points $x_{k0},\ldots,x_{kn}$ (ORT 70, pp. 195-200). Almost all of these procedures either completely refresh the data set at each stage k or, as in the sequential secant algorithm, utilize the points $x_{k-n},\ldots,x_k$ generated during the last n stages of the algorithm. In Chapter 4 procedures will be explored for allowing a more complex choice of auxiliary points in the context of the minimization problem.

Notice that if a secant algorithm is applied to determine the root of a function $F(x) : D \subset R^n \to R^n$ there is no guarantee that the sequence $\{x_k\}$ generated by the algorithm will be defined or will remain in $\underline{D}$. Unless some auxiliary procedure is employed the secant algorithms become stuck if it ever happens that the points $F(x_{k0}),\ldots,F(x_{kn})$ are not in general position. Even if the sequence $\{x_k\}$ is defined there is no guarantee that convergence to a root of $F(x)$ can be obtained from an

arbitrary set of initial points in $\underline{D}$. What can be shown (ORT 70, sections 11.2 and 11.3, theorem 11.3.4) is that if the auxiliary points are chosen in an appropriate fashion, if the initial points are close enough to a root $\hat{x} \in \underline{D}$ of $F(x)$ and if $J(\hat{x})$ is nonsingular then the sequence $\{x_k\}$ is defined and converges to $\hat{x}$. Moreover, the rate of convergence is superlinear. That is

$$\lim_{k \to \infty} \frac{\|x_{k+1} - \hat{x}\|}{\|x_k - \hat{x}\|} = 0. \qquad (3.3.6)$$

In Chapter 2 results were presented concerning a sequence of data sets $\{S_k\}$. Each data set S_k was represented by a pair of matrices (U_k, V_k). In the secant methods which utilize the Newton formulation of theorem 3.3.4 the data sets S_k contain n elements and these elements are of the form

$$s_j = (u_j, v_j) = (F(x_{kj}) - F(x_{k0}),\ x_{kj} - x_{k0}),\ j = 1,\ldots,n. \qquad (3.3.7)$$

The matrices (U_k, V_k) which arise in the Newton secant algorithms represent a data set S_k of the form discussed in Chapter 2. In most Newton secant algorithms the elements of S_k are constrained so that the matrix V_k is nonsingular by choosing the auxiliary points $x_{k0},\ldots,x_{kn}$ in general position. If a Newton secant algorithm is to be defined for all $k > 0$ then the matrices U_k must also be nonsingular or, what is equivalent, the points $F(x_{k0}),\ldots,F(x_{kn})$ must be in general position. If the matrix U_k is nonsingular then by theorems 2.2.4 and 2.3.1 the matrix $V_k U_k^{-1}$ is the unique matrix which emulates and approximates S_k over $\top$. Under most of the selection rules proposed for the auxiliary points $x_{k0},\ldots,x_{kn}$ in Newton secant algorithms the data set S_{k+1} is disjoint from the data set S_k so that no simplification of the calculation of $V_{k+1}U_{k+1}^{-1}$ is possible using information obtained in the calculation

of $V_k U_k^{-1}$.

It is worthwhile to consider another secant method, the sequential secant method, which utilizes the points $x_{k-n},\ldots,x_k$. The following theorem forms the basis for the sequential secant method.

Theorem 3.3.6 If $x_0,\ldots,x_n$ and $F(x_0),\ldots,F(x_n)$ are in general position then the basic secant approximation is given by

$$x_s = x_0 - \bar{V}\bar{U}^{-1}F(x_0) \tag{3.3.8}$$

where

$$\bar{V} = [x_1 - x_0,\ldots,x_n - x_{n-1}]$$

and

$$\bar{U} = [F(x_1) - F(x_0),\ldots,F(x_n) - F(x_{n-1})].$$

Proof: Let U and V denote the matrices given in theorem 3.3.4. We have $\bar{U} = UP$ and $\bar{V} = VP$ where

$$P = \begin{bmatrix} 1 & -1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & -1 \\ 0 & & & 1 \end{bmatrix} \tag{3.3.9}$$

Since P is nonsingular $\bar{U}$ and $\bar{V}$ are nonsingular iff U and V are nonsingular. From theorem 3.3.4

$$x_s = x_0 - VU^{-1}F(x_0)$$

is the basic secant approximation with respect to $x_0,\ldots,x_n$. However,

$$x_s = x_0 - VPP^{-1}U^{-1}F(x_0) \tag{3.3.10}$$

$$= x_0 - \bar{V}\bar{U}^{-1}F(x_0). \qquad \text{Q.E.D.}$$

In the sequential secant algorithm $n + 1$ initial points $x_{-n},x_{-n+1},\ldots,x_0$ are given and at the k th stage the points $x_{k-n},\ldots,x_k$ are taken as the auxiliary points for the calculation of the secant step. The sequential secant algorithm can be stated as

follows.

Algorithm 3.3.7 Assume initial points $x_{-n},\ldots,x_0$ are given and set $k = 0$.

1. If $F(x_k) = 0$ stop.

2. Set $x_{kj} = x_{k-j}$, $j = 0,\ldots,n$.

3. Set $x_{k+1} = x_k - \overline{V}_k\overline{U}_k^{-1}F(x_k)$

   where $\overline{V}_k = [x_{k(n+1)} - x_{kn},\ldots,x_k - x_{k1}]$

   and $\overline{U}_k = [F(x_{k(n+1)}) - F(x_{kn}),\ldots,F(x_k) - F(x_{k1})]$.

4. Set $k = k + 1$ and return to step 1.

In addition to the obvious computational advantage that only one function evaluation need be made at each stage k there is also a savings in the calculation of $\overline{U}_{k+1}^{-1}$ since $\overline{U}_{k+1}$ and $\overline{U}_k$ have $n - 1$ columns in common. In particular, if $\overline{U}_k = [u_{k-n+1},\ldots,u_k]$ then the inverse of the matrix $[u_{k-n+2},\ldots,u_k,u_{k-n+1}]$ is given by $P'\overline{U}_k^{-1}$ where P is the permutation matrix such that $\overline{U}_kP = [u_{k-n+2},\ldots,u_k,u_{k-n+1}]$. Application of theorem 2.4.2 followed by application of theorem 2.4.1 yields $\overline{U}_{k+1}^{-1} = [u_{k-n+2},\ldots,u_k,u_{k+1}]^{-1}$. A more direct procedure for calculating $\overline{U}_{k+1}^{-1}$ is obtained by noting that

$$\overline{U}_{k+1} = \overline{U}_kP + (u_{k+1} - u_{k-n+1})e_n' \qquad (3.3.11)$$

where P is the permutation matrix just described and $e_n' = (0,\ldots,0,1)$. Since it is assumed that $\overline{U}_k$ and $\overline{U}_{k+1}$ are nonsingular the Sherman-Morrison formula (ORT 70, pp. 50) can be applied to equation 3.3.11 to obtain a closed formula for $\overline{U}_{k+1}^{-1}$.

Although the sequential secant algorithm affords some computational advantages, it is more prone to undesirable behavior than the Newton secant algorithms. Since at stage k there is no guarantee that the

points $x_{k-n},\ldots,x_k$ are in general position it may happen that the matrix $\overline{V}_k$ is singular. If this occurs and if the vector $U_k^{-1}F(x_k)$ is in $N(\overline{V}_k)$ then the algorithm becomes stuck since $x_{k+1} = x_k$. For this reason superlinear convergence cannot be proven even if the initial points are close to a solution $\hat{x}$ where $J(\hat{x})$ is nonsingular (ORT 70, theorem 11.3.5). In addition the theoretical difficulties mentioned earlier with the Newton secant methods are still present in the sequential secant algorithm.

As with the Newton secant algorithms, if $\overline{U}_k$ is nonsingular then the matrix $\overline{V}_k\overline{U}_k^{-1}$ is the unique matrix which emulates and approximates the data set $\overline{S}_k$ over $\top$. Because the data set $\overline{S}_{k+1}$ is generated by deleting one element of $\overline{S}_k$, the oldest element, and adding a new element, the calculation of $\overline{V}_{k+1}\overline{U}_{k+1}^{-1}$ is simplified. If a different permutation matrix P is chosen, any element of the set $\overline{S}_k$ can be deleted without computational burden. Such a procedure could be used to counteract the tendency of the matrices $\overline{U}_k$ and $\overline{V}_k$ to become singular or ill-conditioned. In Chapters 4 and 5 minimization algorithms will be proposed which make use of this procedure as well as more general replacement procedures.

In any of the secant algorithms difficulty arises if at some stage k the matrix U_k is singular since, if this occurs, the matrix $V_kU_k^{-1}$ will be undefined. When $V_kU_k^{-1}$ does not exist it seems logical to choose some approximation to S_k over $\top$. Recall from Chapter 2 that the family of approximations to $S_k = (V_k,U_k)$ over $\top$ is given by

$$A_k = V_kU_k^{+} + Y_k(I - U_kU_k^{+}) \qquad (3.3.12)$$

where Y_k is arbitrary. It is not obvious what choice to make for the matrix Y_k when $R(U_k)$ is not all of R^n. In fact, as the following lemma shows, under certain conditions the set of points x_{k+1} which can be

obtained by using an approximation to S_k over $\top$ in a secant algorithm from the point x_k is all of R^n.

Lemma 3.3.8 Suppose the data set $S_k = (U_k, V_k)$ and $F(x_k) \in R^n$ are given. If $F(x_k) \notin R(U_k)$, then for any vector $\omega \in R^n$ there exists a solution for Y_k of the equation

$$\omega = V_k U_k^+ F(x_k) + Y_k (I - U_k U_k^+) F(x_k). \qquad (3.3.13)$$

Proof: If $F(x_k) \notin R(U_k)$ then $(I - U_k U_k^+) F(x_k) \neq 0$. The result follows immediately as there exists a Y_k which transforms any nonzero vector in R^n into any vector in R^n. Q.E.D.

If $F(x_k) \in R(U_k)$ then $(I - U_k U_k^+) F(x_k) = 0$ and the choice of Y_k has no effect on ω. In this case the secant step should be replaced by

$$x_{k+1} = x_k - V_k U_k^+ F(x_k). \qquad (3.3.14)$$

A clever choice for Y_k when $F(x_k) \notin R(U_k)$ may yield interesting algorithms. In the minimization problem additional structure is present which indicates a particular choice for Y_k when $F(x_k) \notin R(U_k)$. This will be explored in Chapter 4.

Since an approximation to a data set S_k over $\top$ exists regardless of the number of elements in S_k, secant-like algorithms could be constructed which utilize more or fewer than n auxiliary points at each stage. Also, since elements of the data sets S_k can be pairs $(u,v) \in R^n \times R^m$, $m \neq n$, secant-like algorithms could be constructed for finding the roots of a function $F(x) : D \; R^n \to R^m$. Neither of these possibilities will be explored in this dissertation.

3.4 Minimization Methods

In 1963 a paper by Fletcher and Powell (FLE 63) stimulated research on a class of minimization algorithms which have proven to be very effec-

tive. The algorithm considered by Fletcher and Powell was originally proposed by Davidon (DAV 59). Since 1963 Broyden (BRO 70, 70a), Pearson (PEA 69), and Murtagh and Sargent (MUR 70) among others have proposed algorithms which bear a similarity to the Davidon algorithm but which are distinct from it. In this section we will demonstrate the underlying commonality of these algorithms and suggest modifications and generalizations of them.

Using the notation introduced at the beginning of this chapter for the objective function $f(x):D \subset R^n \to R^1$ a general minimization algorithm can be stated which includes the above mentioned class of algorithms.

Algorithm 3.4.1

Assume $x_0 \in \underline{D}$ and an initial $n \times n$ matrix A_0 are given and set $k = 0$.

1. If $g_k = g(x_k) = 0$ stop.
2. Set $x_{k+1} = x_k - \lambda_k A_k' g_k$ where λ_k is chosen so that
$$f(x_k - \lambda_k A_k' g_k) = \min_{\lambda \in R^1} f(x_k - \lambda A_k' g_k).$$
3. Compute A_{k+1} by a recursion relation.
4. Set $k = k + 1$ and return to step 1.

Notice that step 2 of the algorithm involves a subalgorithm which minimizes the objective function on a line. This "linear search" algorithm is of no interest in itself here. However, as will be seen later, the properties of specific algorithms will be changed when the linear minimization is not carried out exactly. If $A_k' g_k \neq 0$ exact solution of the linear minimization causes $g_{k+1} A_k' g_k$ to be zero.

In order to complete the specification of the algorithm it is

necessary to define the recursion of step 3. The following recursions have been proposed by various researchers.

$$A_{k+1} = A_k + v_k(v_k'u_k)^{-1}v_k' - A_k u_k(u_k'A_k u_k)^{-1}u_k'A_k \quad \text{Davidon} \quad (3.4.1)$$

$$A_{k+1} = A_k + e_k(e_k'u_k)^{-1}e_k' \quad \text{Murtagh and Sargent} \quad (3.4.2)$$

$$A_{k+1} = A_k + e_k(v_k'u_k)^{-1}v_k' \quad \text{Pearson 1} \quad (3.4.3)$$

$$A_{k+1} = A_k + e_k(u_k'A_k u_k)^{-1}u_k'A_k \quad \text{Pearson 2} \quad (3.4.4)$$

$$A_{k+1} = A_k + (1 + \beta_k u_k'A_k u_k)\, v_k\, (v_k'u_k)^{-1}v_k' \quad \text{Broyden} \quad (3.4.5)$$

$$- \beta_k(A_k u_k v_k' - v_k u_k' A_k)$$

$$- (1 - \beta_k v_k'u_k)\, A_k u_k\, (u_k'A_k u_k)^{-1} u_k A_k, \quad \beta_k \geq 0$$

In these formulas $v_k = x_{k+1} - x_k$, $u_k = g_{k+1} - g_k$ and $e_k = v_k - A_k u_k$. The names associated with the recursions are not necessarily their originators but rather individuals who have analyzed them extensively.

Because it is difficult to obtain convergence results for an arbitrary twice differentiable objective function, algorithms have historically been constructed to have good performance when applied to a quadratic function, that is a function whose Hessian matrix is a constant function of x on D. Since in some small neighborhood of the minimum of $f(x)$, the Hessian is "almost constant", it is reasoned that an algorithm so constructed will perform well on any $f(x)$ in the terminal stage. Following this logic we temporarily restrict attention to a quadratic objective function. In algorithm 3.4.1 it is desirable for the sequence $\{A_k\}$ to be an emulating sequence. The following theorem demonstrates why.

Theorem 3.4.2 Suppose an initial point $x_0 \in R^n$ is given. Suppose the matrices $A_0, A_1, \ldots, A_k$ of algorithm 3.4.1 are chosen such that $g_k' A_k' g_k \neq 0$ when $g_k \neq 0$ and such that A_k emulates S_k over $\top$ where $S_0 = \emptyset$, $S_k = (U_k, V_k) = [(u_0, \ldots, u_{k-1}), (v_0, \ldots, v_{k-1})]$, $k = 1, \ldots, n$ $v_i = x_{i+1} - x_i$ and $u_i = g_{i+1} - g_i$, $i = 0, \ldots, k$. If algorithm 3.4.1 is applied to an objective function $f(x) : R^n \to R^1$ with a constant positive definite Hessian matrix H, then either $g_k = 0$ for some $k < n$ or

$$U_k' v_k = V_k' u_k = 0, \quad k = 1, \ldots, n, \tag{3.4.6}$$

$$A_n = H^{-1} \tag{3.4.7}$$

and

$$g_n = 0. \tag{3.4.8}$$

Proof: Since $g_k' A_k' g_k \neq 0$ when $g_k \neq 0$, the search direction $A_k' g_k$ is nonzero unless $g_k = 0$. Because H is positive definite $f(x)$ is strictly convex and therefore a solution must exist to the linear search problem. It follows that the algorithm is determined for n steps unless $g_k = 0$ for some $k < n$. It must be that $\lambda_k \neq 0$ for $k = 0, \ldots, n-1$, since if $\lambda_k = 0$ the minimum has been achieved along the line $x_k + \lambda A_k' g_k$ which implies that $g_k' A_k' g_k = 0$. Because of the linear search

$$g_{k+1}' v_k = 0, \quad k = 0, \ldots, n-1. \tag{3.4.9}$$

With $k = 1$ equation 3.4.9 gives

$$g_1' V_1 = 0. \tag{3.4.10}$$

We will now show that $g_k' V_k = 0$ implies that $g_{k+1}' V_{k+1} = 0$ and hence establish by induction that

$$g_k' V_k = 0, \quad k = 1, \ldots, n. \tag{3.4.11}$$

Since $g_k = Hx_k + c$, we have

$$\lambda_k^{-1} u_k' V_k = \lambda_k^{-1} v_k' H V_k \qquad (3.4.12)$$

$$= \lambda_k^{-1} v_k' U_k$$

$$= g_k' A_k U_k$$

$$= g_k' V_k = 0.$$

Since λ_k is nonzero and finite it must be that $u_k' V_k = v_k' U_k = 0$. From this it follows that

$$g_{k+1}' V_{k+1} = [(u_k + g_k)' V_k,\ g_{k+1}' V_k] \qquad (3.4.13)$$

$$= [0 + 0, 0]$$

which completes the induction. Next, since $v_k' U_k = 0$ and $U_k = HV_k$, $k = 1,\ldots,n$, it must be that

$$v_k' H V_k = u_k' H^{-1} U_k = 0, \quad k = 1,\ldots,n. \qquad (3.4.12)$$

Therefore, because H is positive definite and $\lambda_k \neq 0$, the vectors $(v_0,\ldots,v_{n-1})$ span R^n and there exists a unique matrix H^{-1} which emulates S_n over $\top$, that is H^{-1} is the unique matrix such that $H^{-1} U_n = V_n$. Since A_n was chosen to emulate S_n over $\top$, $A_n = H^{-1}$. Finally, since $g_n' V_n = 0$ and since the vectors $(v_0,\ldots,v_{n-1})$ span R^n, $g_n = 0$. Q.E.D.

Notice that the requirement that $g_k' A_k' g_k \neq 0$ can be checked at each stage since g_k is known when the matrix A_k is determined. If A_k is chosen positive definite, $g_k' A_k' g_k$ is automatically nonzero. It is also true that if A_k is chosen negative definite $g_k' A_k g_k$ is nonzero. However, if A_k emulates (U_k, V_k) and $U_k = HV_k$ where H is positive definite then A_k cannot be negative definite since this would imply that $x' A_k x = x' H^{-1} x < 0$ for all $x \in R(U_k)$ which is a contradiction.

If the search direction is chosen to be $A_k'\omega_k$ where ω_k is any nonzero vector such that $\omega_k'V_k = 0$ and $g_k'A_k'\omega_k \neq 0$ rather than $A_k'g_k$ the proof of theorem 3.4.2 is unchanged. This suggests the following more general minimization procedure for quadratic functions.

Algorithm 3.4.3

Assume $x_0 \in D$ and an initial $n \times m$ matrix A_0 are given and set $k = 0$.

1. If $g_k = 0$ stop.
2. If $k = 0$ set $x_{k+1} = x_k - \lambda_k A_k'\omega_k$ where ω_k is an arbitrary nonzero vector and where λ_k is chosen such that
$$f(x_k - \lambda_k A_k'\omega_k) = \min_{\lambda \in R^1} f(x_k - \lambda A_k'\omega_k).$$
3. If $k > 0$ set $x_{k+1} = x_k - \lambda_k A_k'\omega_k$ where ω_k is any nonzero vector such that $\omega_k'V_k = 0$, λ_k is chosen as in step 2 and $S_k = (U_k, V_k)$ is defined in theorem 3.4.2.
4. Compute A_{k+1} by a recursion relation.
5. Set $k = k + 1$ and return to step 1.

For this algorithm a theorem similar to theorem 3.4.2 can be stated.

Theorem 3.4.4 Suppose an initial point $x_0 \in R^n$ is given. Suppose the matrices $A_0, A_1, \ldots, A_k$ of algorithm 3.4.3 are chosen such that $g_k'A_k'\omega_k \neq 0$ when $g_k \neq 0$ and such that A_k emulates S_k over $\top$ where S_k is as defined in theorem 3.4.2. If algorithm 3.4.3 is applied to an objective function $f(x) : R^n \to R^1$ with a constant positive definite Hessian matrix H, then either $g_k = 0$ for some $k < n$ or

$$U_k'V_k = V_k'U_k = 0, \quad k = 1, \ldots, n, \qquad (3.4.14)$$

$$A_n = H^{-1} \tag{3.4.15}$$

and

$$g_n = 0. \tag{3.4.16}$$

The proof of theorem 3.4.4 is identical to the proof of theorem 3.4.2 except that in equation 3.4.12 ω_k replaces g_k. Since in theorem 3.4.2 it was shown that $g_k'V_k = 0$ for $k = 1,\ldots,n$, theorem 3.4.2 is a special case of theorem 3.4.4.

Algorithm 3.4.3 is not entirely sensible on a non-quadratic surface for more than n steps since after n steps V_k will span R^n. To apply algorithm 3.4.3 past n steps some method must be used to delete old columns from U_k and V_k. One such procedure would be to reset every n steps. In Chapter 4 other deletion procedures are considered in conjunction with a different algorithm.

If the linear search in algorithm 3.4.3 is not performed exactly, then the proof of theorem 3.4.4 breaks down. However, if the matrix $A_n = H^{-1}$ in algorithm 3.4.3, then the step

$$x_{n+1} = x_n - A_n'g_n \tag{3.4.17}$$

makes $g_{n+1} = 0$. In the case of one particular recursion, the Murtagh and Sargent formula 3.4.2, under certain conditions $A_n = H^{-1}$ even in the absence of an exact linear search. This recursion will be considered further later in the section.

We now turn our attention to the relationship between the specific recursion formulas 3.4.1 through 3.4.5 and theorem 2.4.5. Our main objective will be to show that the matrices A_k generated by these formulas emulate the data sets S_k of theorem 3.4.2. Recall from Chapter 2 that if $S_i = (U_i,V_i)$ and $S_j = (U_j,V_j)$ are two data sets, S_{ij}^p is the subset of S_i whose elements are also contained in S_j and S_{ij}^q is the

subset of S_i whose elements are not contained in S_j. The ordering of S_i is assumed to be such that the elements of S_{ij}^p precede the elements of S_{ij}^q in S_i. If in theorem 2.4.5 $m = n$ and the particular choice

$$Z_i^1 = a_i^1 V_{ij}^{q'} + a_i^2 U_{ij}^{q'} A_j \tag{3.4.18}$$

$$Z_i^2 = a_i^3 V_{ij}^{q'} + a_i^4 U_{ij}^{q'} A_j \tag{3.4.19}$$

is made where $a_i^1,\ldots,a_i^4$ are constants then a formula arises which contains the recursions 3.4.1 through 3.4.5.

The choice of Z_i^1 and Z_i^2 in equations 3.4.18 and 3.4.19 was made after consideration of the existing recursion formulas in an attempt to relate these formulas to the material of Chapter 2. Other choices of Z_i^1 and Z_i^2 in theorem 2.4.5 may lead to still other recursion formulas for emulating sequences. Direct substitution of equations 3.4.18 and 3.4.19 in theorem 2.4.5 yields:

Corollary 3.4.5 Suppose $S_i = (U_i, V_i)$ and $S_j = (U_j, V_j)$ are non-empty data sets and that A_j emulates S_j over $\top$. If S_{ij}^p and S_{ij}^q are not empty and if

$$a_i^1 V_{ij}^{q'} U_{ij}^p + a_i^2 U_{ij}^{q'} V_{ij}^p = 0, \tag{3.4.20}$$

$$a_i^3 V_{ij}^{q'} U_{ij}^p + a_i^4 U_{ij}^{q'} V_{ij}^p = 0, \tag{3.4.21}$$

$$V_{ij}^q (a_i^1 V_{ij}^{q'} U_{ij}^q + a_i^2 U_{ij}^{q'} A_j U_{ij}^q)^- (a_i^1 V_{ij}^{q'} U_{ij}^q + a_i^2 U_{ij}^{q'} A_j U_{ij}^q) \tag{3.4.22}$$

$$= V_{ij}^q,$$

$$A_j U_{ij}^q (a_i^3 V_{ij}^{q'} U_{ij}^q + a_i^4 U_{ij}^{q'} A_j U_{ij}^q)^- (a_i^3 V_{ij}^{q'} U_{ij}^q + a_i^4 U_{ij}^q A_j U_{ij}^q) \tag{3.4.23}$$

$$= A_j U_{ij}^q,$$

where the g-inverses are arbitrary, then

$$A_i = A_j + V_{ij}^q(a_i^1 V_{ij}^q U_{ij}^q + a_i^2 U_{ij}^{q'} A_j U_{ij}^q)^- (a_i^1 V_{ij}^q + a_i^2 U_{ij}^{q'} A_j) \quad (3.4.24)$$

$$- A_j U_{ij}^q(a_i^3 V_{ij}^q U_{ij}^q + a_i^4 U_{ij}^q A_j U_{ij}^q)^- (a_i^3 V_{ij}^q + a_i^4 U_{ij}^q A_j)$$

emulates S_i over T. Further, if S_{ij}^p is empty and if conditions 3.4.22 and 3.4.23 are satisfied then A_i given by equation 3.4.24 emulates S_i over T.

Corollary 3.4.5 allows the sets S_i and S_j to differ in an arbitrary manner. In order to arrive at the recursion formulas 3.4.1 through 3.4.5 it is necessary to consider a particular sequence of data sets.

Corollary 3.4.6 Consider the sequence of data sets $S_0 = \emptyset$, $S_k = (U_k, V_k) = ([u_0,\ldots,u_{k-1}], [v_0,\ldots,v_{k-1}])$, $k = 1,\ldots,n$. If A_k emulates S_k over T and if

$$a_k^1 v_k' U_k + a_k^2 u_k' V_k = 0, \quad k > 0, \quad (3.4.25)$$

$$a_k^3 v_k' U_k + a_k^4 u_k' V_k = 0, \quad k > 0, \quad (3.4.26)$$

$$v_k(a_k^1 v_k' u_k + a_k^2 u_k' A_k u_k)^- (a_k^1 v_k' u_k + a_k^2 u_k' A_k u_k) = v_k, \; k \geq 0, \quad (3.4.27)$$

$$A_k u_k(a_k^3 v_k' u_k + a_k^4 u_k' A_k u_k)^- (a_k^3 v_k' u_k + a_k^4 u_k' A_k u_k) = A_k u_k, \; k \geq 0, \quad (3.4.28)$$

then

$$A_{k+1} = A_k + v_k(a_k^1 v_k' u_k + a_k^2 u_k' A_k u_k)^- (a_k^1 v_k' + a_k^2 u_k' A_k) \quad (3.4.29)$$

$$- A_k u_k(a_k^3 v_k' u_k + a_k^4 u_k' A_k u_k)^- (a_k^3 v_k' + a_k^4 u_k' A_k), \quad k \geq 1,$$

emulates S_{k+1} over T.

Proof: The proof follows immediately from corollary 3.4.5 by taking $S_j = S_k$, $S_i = S_{k+1}$, $S_{ij}^p = S_k$ and $S_{ij}^q = \{(u_k, v_k)\}$. Notice that by

definition any matrix A_0 emulates S_0 and that $S_{10}^p = \emptyset$.

Conditions 3.4.27 and 3.4.28 of corollary 3.4.6 are satisfied iff

$$a_k^1 v_k' u_k + a_k^2 u_k' A_k u_k \neq 0 \qquad (3.4.30)$$

and

$$a_k^3 v_k' u_k + a_k^4 u_k' A_k u_k \neq 0 \qquad (3.4.31)$$

respectively, since the g-inverse of a nonzero scalar is its reciprocal and a g-inverse of the zero scalar is zero. Conditions 3.4.25 and 3.4.26 of corollary 3.4.6 are satisfied if, but not only if, the orthogonality conditions

$$v_k' U_k = 0 \qquad (3.4.32)$$

and

$$u_k' V_k = 0 \qquad (3.4.33)$$

are satisfied.

Before proceeding it should be pointed out that two of the coefficients in equation 3.4.29 are redundant. If inequalities 3.4.30 and 3.4.31 hold, then equation 3.4.29 can be written as

$$A_{k+1} = A_k + v_k(v_k'u_k + b_k^1 u_k' A_k u_k)^{-1} (v_k' + b_k^1 u_k' A_k) \qquad (3.4.34)$$

$$- A_k u_k (v_k' u_k + b_k^2 u_k' A_k u_k)^{-1} (v_k' + b_k^2 u_k' A_k)$$

where $b_k^1 = a_k^2/a_k^1$ and $b_k^2 = a_k^4/a_k^3$, provided a_k^1 and a_k^3 are nonzero. If a_k^1 or a_k^3 is zero, similar formulas can be written involving a_k^1/a_k^2 and a_k^3/a_k^4. Therefore equation 3.4.29 depends only on the ratios a_k^2/a_k^1 and a_k^4/a_k^3.

We are now in a position to state a result about the family of recursions 3.4.29 when applied in minimization algorithm 3.4.3.

Theorem 3.4.7 Suppose an initial point $x_0 \in R^n$ is given. Suppose the matrix A_0 and the vector ω_0 of algorithm 3.4.3 are chosen so that

$g_0'A_0'\omega_0 \neq 0$ and the matrices A_k, $k > 0$, are chosen by the recursion

$$A_{k+1} = A_k + v_k(a_k^1 v_k'u_k + a_k^2 u_k'A_k u_k)^- (a_k^1 v_k' + a_k^2 u_k'A_k) \qquad (3.4.35)$$

$$- A_k u_k(a_k^3 v_k'u_k + a_k^4 u_k'A_k u_k)^- (a_k^3 v_k' + a_k^4 u_k'A_k)$$

where $v_k = x_{k+1} - x_k$ and $u_k = g_{k+1} - g_k$. Suppose the coefficients $a_k^1,\ldots,a_k^4$ are chosen so that $g_k'A_k'\omega_k \neq 0$, $k > 0$, and so that

$$a_k^1 v_k'u_k + a_k^2 u_k'A_k u_k \neq 0 \qquad (3.4.36)$$

$$a_k^3 v_k'u_k + a_k^4 u_k'A_k u_k \neq 0 \qquad (3.4.37)$$

for $k \geq 0$. Suppose algorithm 3.4.3 is applied to an objective function $f(x) : R^n \to R^1$ with constant positive definite Hessian H. Then either $g_k = 0$ for some $k < n$ or A_k emulates S_k over $\top$, where $S_0 = \emptyset$ and $S_k = (U_k,V_k) = ([u_0,\ldots,u_{k-1}], [v_0,\ldots,v_{k-1}])$ $k = 1,\ldots,n$. Further,

$$U_k'v_k = V_k'u_k = 0, \quad k = 1,\ldots,n, \qquad (3.4.38)$$

$$A_n = H^{-1} \qquad (3.4.39)$$

and

$$g_n = 0. \qquad (3.4.40)$$

Proof: Assume $g_k \neq 0$ for $k < n$. A_0 emulates S_0 over $\top$ by definition. Therefore, by corollary 3.4.6 and equations 3.4.36 and 3.4.37, A_1 emulates S_1 over $\top$ and by theorem 3.4.4

$$U_1'v_1 = 0 \quad \text{and} \quad V_1'u_1 = 0. \qquad (3.4.41)$$

Assume then that A_k emulates S_k over $\top$ and that

$$U_k'v_k = 0 \quad \text{and} \quad V_k'u_k = 0. \qquad (3.4.42)$$

By corollary 3.4.6, equations 3.4.36, 3.4.37, and 3.4.42, A_{k+1} emulates

S_{k+1} over $\top$ and by theorem 3.4.4

$$U'_{k+1} v_{k+1} = 0 \quad \text{and} \quad V'_{k+1} u_{k+1} = 0. \tag{3.4.43}$$

By induction then A_k emulates S_k over $\top$ for $k = 0,\ldots,n$. Equations 3.4.39 and 3.4.40 follow from theorem 3.4.4. Q.E.D.

The algorithms of theorem 3.4.7 contain all of the algorithms specified by general algorithm 3.4.1 and the recursion formulas 3.4.1 through 3.4.5. To obtain these formulas the following choices are made for the coefficients $a_k^1,\ldots,a_k^4$.

$$a_k^1 \neq 0,\ a_k^2 = 0,\ a_k^3 = 0,\ a_k^4 \neq 0 \qquad \text{Davidon} \tag{3.4.44}$$

$$a_k^1 = -a_k^2 \neq 0,\ a_k^3 = -a_k^4 \neq 0 \qquad \text{Murtagh and Sargent} \tag{3.4.45}$$

$$a_k^1 \neq 0,\ a_k^2 \neq 0,\ a_k^3 = 0,\ a_k^4 = 0 \qquad \text{Pearson 1} \tag{3.4.46}$$

$$a_k^1 = 0,\ a_k^2 \neq 0,\ a_k^3 = 0,\ a_k^4 \neq 0 \qquad \text{Pearson 2} \tag{3.4.47}$$

$$a_k^1 = 1 + \beta_k u_k' A_k u_k,\ a_k^2 = -\beta_k v_k' u_k,\ a_k^3 = \beta_k u_k' A_k u_k$$

$$a_k^4 = 1 - \beta_k v_k' u_k,\quad \beta_k \geq 0 \qquad \text{Broyden} \tag{3.4.48}$$

The algorithms of theorem 3.4.7 are more general in four respects than these known algorithms. First, general algorithm 3.4.3 allows a far more general choice of a search direction for a given matrix A_k. Second, in algorithm 3.4.3 we recognize that it is only necessary to choose the matrix A_k and the vector ω_k such that $g_k' A_k \omega_k \neq 0$ in order to insure that the algorithm does not become stuck at any stage. Third, the family of recursion formulas given by equation 3.4.29 includes all of the recursion formulas 3.4.1 through 3.4.5 as well as a large class of new recursions. Finally, theorem 3.4.7 does not restrict recursion formulas to be the

same at each stage of the algorithm. With the wide latitude afforded by the algorithms of theorem 3.4.7 it should be possible to improve upon the performance of existing algorithms.

It may be that the more general recursion formula of corollary 3.4.5 can be utilized in minimization algorithms constructed to generate more than one new element for the data set S_k at each stage. Also the formula of corollary 3.4.5 could be useful in problems other than the minimization problem. These aspects are not pursued here in depth.

In order to fulfill the hypothesis of theorem 3.4.7 it is necessary to choose the coefficients $a_k^1,\ldots,a_k^4$ so that $g_k'A_k\omega_k \neq 0$ and so that equations 3.4.36 and 3.4.37 are satisfied. In the known algorithms ω_k is chosen to be g_k. Two choices for $a_k^1,\ldots,a_k^4$ have been shown, in this case, to guarantee that $g_k'A_k'g_k \neq 0$ and that equations 3.4.36 and 3.4.37 are satisfied. The first of these is the choice which yields the Davidon formula, equation 3.4.1. In (FLE 63) it is shown that if A_0 is chosen to be positive definite and the Davidon formula is applied to any nonzero sequence of vectors (u_k,v_k), $k = 1,2,\ldots$, then A_k is positive definite for all k. In (POW 71) it is further shown that if the Davidon algorithm is applied to an arbitrary twice differentiable function and if an exact linear search is employed then $v_k'u_k \neq 0$ and $u_k'A_ku_k \neq 0$ unless $g_{k+1} = 0$. The same results have been shown to hold (BRO 70, 70a) for the choice which yields the Broyden formula given by equation 3.4.5. No such results can be obtained for equations 3.4.2, 3.4.3, and 3.4.4.

The Murtagh and Sargent formula given by equation 3.4.2 has a further interesting property. If the Murtagh and Sargent recursion is applied to

a sequence of data sets $S_0 = \emptyset$, $S_k = (u_k, V_k) = ([u_0,\ldots,u_{k-1}], [v_0,\ldots,v_{k-1}])$, $k = 1,\ldots,n$, where $u_i = \bar{H}v_i$, $i = 0,\ldots,n-1$, for some symmetric matrix $\bar{H}$, then equations 3.4.25 and 3.4.26 of corollary 3.4.6 are automatically satisfied since

$$a_k^1 v_k' U_k + a_k^2 u_k' V_k = a_k^1 (v_k' \bar{H} V_k - v_k' \bar{H}' V_k) = 0 \qquad (3.4.49)$$

and

$$a_k^3 v_k' U_k + a_k^4 u_k' V_k = a_k^3 (v_k' \bar{H} V_k - v_k' \bar{H}' V_k) = 0. \qquad (3.4.50)$$

This property leads to the consideration of the following algorithm for minimizing $f(x) : D \subset R^n \to R^1$ on $\underline{D}$ which does not involve an exact linear search at each stage.

Algorithm 3.4.8 Assume $x_0 \in \underline{D}$ and an initial matrix A_0 are given and set $k = 0$.

1. If $g_k = g(x_k) = 0$ stop.
2. If $k < n$ set $x_{k+1} = x_k + v_k$ where v_k is any vector not in $R(V_k)$ and where $V_k = [v_0,\ldots,v_{k-1}]$.
3. If $k \geq n$ set $x_{k+1} = x_k - A_k g_k$ and set $v_k = x_{k+1} - x_k$.
4. Set $A_{k+1} = A_k + e_k(e_k' u_k)^{-1} e_k'$ where $e_k = v_k - A_k u_k$ and $u_k = g_{k+1} - g_k$.
5. Set $k = k+1$ and go to step 1.

As with algorithm 3.4.3 some modification of this procedure may be desirable when the objective function is not quadratic. If the objective function $f(x)$ is quadratic and if its Hessian matrix is nonsingular then the following theorem holds.

Theorem 3.4.9 Suppose an initial point $x_0 \in R^n$ and an initial matrix A_0 are given. Suppose algorithm 3.4.8 is applied to an objective function with constant nonsingular Hessian H. If $e_k' u_k \neq 0$ for

$k = 0,\ldots,n-1$ then either $g_k = 0$ for some $k < n+1$ or

$$A_k = H^{-1} \tag{3.4.51}$$

and

$$g_{k+1} = 0. \tag{3.4.52}$$

Proof: First notice that algorithm 3.4.8 is determined for $k = 0,\ldots,n$ provided $g_k \neq 0$ and $e_k' u_k \neq 0$ for $k = 0,\ldots,n$. Because of equations 3.4.49 and 3.4.50, conditions 3.4.25 and 3.4.26 of corollary 3.4.6 are satisfied. Conditions 3.4.27 and 3.4.28 of corollary 3.4.6 are satisfied iff $e_k' u_k \neq 0$. Therefore A_k emulates S_k over $\top$ for $k = 0,\ldots,n$ where $S_k = (U_k, V_k) = ([u_0,\ldots,u_{k-1}], [v_0,\ldots,v_{k-1}])$. Because $v_k \notin R(V_k)$, $k = 1,\ldots,n-1$, the vectors $(v_0,\ldots,v_{k-1})$ are a basis for R^n and therefore there exists a unique matrix $A_n = H^{-1}$ which emulates S_n over $\top$. Finally, since on a quadratic surface

$$g(x) = g(y) - H(x - y) \tag{3.4.53}$$

we have

$$g_{n+1} = g_n + H(x_n - H^{-1}g_n - x_n)$$

$$= g_n - HH^{-1}g_n$$

$$= 0. \qquad \text{Q.E.D.}$$

It is necessary to require that H be nonsingular since if H is singular and the data set S_n is such that $(v_0,\ldots,v_{n-1})$ spans R^n, no matrix A_n exists which emulates S_n over $\top$. It does not seem to be possible without a priori knowledge of H to choose the vectors v_k so that $e_k' u_k \neq 0$.

One would hope that the results of theorems 3.4.7 and 3.4.9 could be extended at least in part when algorithms 3.4.3 and 3.4.8 are applied to an arbitrary twice differentiable function $f(x) : D \subset R^n \to R^1$.

Unfortunately this is not the case. The proofs of theorems 3.4.6 and 3.4.9 are predicated on the existence of a nonsingular symmetric matrix H such that $U_k = HV_k$, $k = 1,\ldots,n$. If $f(x)$ is not quadratic it may happen that no matrix exists which transforms V_k into U_k. More importantly there may exist no matrix which emulates $S_k = (V_k,U_k)$ over $\top$. Even if an A_k exists which emulates S_k over $\top$ it may not be symmetric. In the absence of a matrix which emulates S_k over $\top$ it would seem desirable to obtain a matrix which approximates S_k over $\top$. As will be shown in section 3.5 it is unlikely that any of the formulas of this section generate a sequence which approximates S_k over $\top$ when $f(x)$ is not quadratic.

3.5 Recursive Formulas for Approximating Sequences

Recall from theorem 2.2.4 that the family of approximations to a data set $S_k = (U_k,V_k)$ over $\top$ is given by

$$A_k = V_kU_k^+ + Y_k(I - U_kU_k^+) \tag{3.5.1}$$

where Y_k is arbitrary. If a sequence of data sets $S_k = (U_k,V_k) = ([u_0,\ldots,u_{k-1}], [v_0,\ldots,v_{k-1}])$, $k = 1,2,\ldots$, is being considered, theorem 2.5.1 can be used to compute U_k^+ from U_{k-1}^+ and arrive at the family of approximation to S_k over $\top$. Although this procedure is quite efficient it is worthwhile to derive from theorem 2.4.1 an explicit recursive formula for the family of approximations to S_k over $\top$.

We begin by developing an expression for the matrix $[X,x][A,a]^+$. Direct application of theorem 2.4.1 yields

$$[X,x][A,a]^+ = XA^+ + (x - XA^+a)(a'(I - AA^+)a)^{-1} \tag{3.5.2}$$

$$a'(I - AA^+), \qquad a \notin R(A),$$

$$[X,x][A,a]^+ = XA^+ + (x - XA^+a)(1 + a'A^{+\prime}A^+a)^{-1} \tag{3.5.3}$$

$$a'A^{+'}A^{+}, \qquad a \in R(A).$$

Setting $[X,x] = [A,a]$, an expression for the operator $I - [A,a][A,a]^{+}$ is obtained.

$$I - [A,a][A,a]^{+} = (I - AA^{+}) + (I - AA^{+})a\,(a'(I - AA^{+})a)^{-1} \qquad (3.5.4)$$

$$a'(I - AA^{+}),\ a \notin R(A)$$

$$I - [A,a][A,a]^{+} = (I - AA^{+}) + (I - AA^{+})a(1 + a'A^{+'}A^{+}a)^{-1} \qquad (3.5.5)$$

$$a'A^{+'}A^{+},\ a \in R(A).$$

Next, applying theorem 2.4.1 to the matrix $[A,a]^{+'}[A,a]^{+}$ yields

$$[A,a]^{+'}[A,a]^{+} = (I - ab')'A^{+'}A^{+}(I - ab') + bb' \qquad (3.5.6)$$

where

$$b = (I - AA^{+})a(a'(I - AA^{+})a)^{-1},\ a \notin R(A) \qquad (3.5.7)$$

$$b = A^{+'}A^{+}a(1 + a'A^{+'}A^{+}a)^{-1},\ a \in R(A). \qquad (3.5.8)$$

Substituting equations 3.5.7 and 3.5.8 into equation 3.5.6 gives

$$[A,a]^{+'}[A,a]^{+} = A^{+'}A^{+} - A^{+'}A^{+}a(a'(I - AA^{+})a)^{-1}a'(I - AA^{+}) \qquad (3.5.9)$$

$$- (I - AA^{+})a(a'(I - AA^{+})a)^{-1}a'A^{+'}A^{+}$$

$$+ (I - AA^{+})a(a'(I - AA^{+})a)^{-1}a'A^{+'}A^{+}a$$

$$(a'(I - AA^{+})a)^{-1}a'(I - AA^{+})$$

$$+ (a'(I - AA^{+})a)^{-1}(I - AA^{+})a$$

$$(a'(I - AA^{+})^{-1}a'(I - AA^{+}), \quad a \notin R(A),$$

$$[A,a]^{+'}[A,a]^{+} = A^{+'}A^{+} - 2A^{+'}A^{+}a(1 + a'A^{+'}A^{+}a)^{-1}a'A^{+'}A^{+} \qquad (3.5.10)$$

$$+ A^{+'}A^{+}a(1 + a'A^{+'}A^{+}a)^{-1}a'A^{+'}A^{+}a(1 + a'A^{+'}A^{+}a)^{-1}a'A^{+'}A^{+}$$

$$+ (1 + a'A^{+'}A^{+})^{-1}A^{+'}A^{+}a(1 + a'A^{+'}A^{+}a)^{-1}a'A^{+'}A^{+}, \quad a \in R(A).$$

Notice that equations 3.5.4, 3.5.5, 3.5.9, and 3.5.10 are coupled in the sense that the expressions for $I - [Aa][Aa]^{+}$ involve $A^{+'}A^{+}$ and the expressions for $A^{+'}A^{+}$ involve $I - AA^{+}$.

Assume that a sequence of data sets $S_0 = \emptyset$, $S_k = (U_k, V_k) = ([u_0,\ldots,u_{k-1}], [v_0,\ldots,v_{k-1}])$ are given for $k = 1,2,\ldots$. From equations 3.5.4, 3.5.5, 3.5.9, and 3.5.10 we may write a recursive formula for the matrices $M_k = (I - U_kU_k^{+})$ and $N_k = U_k^{+'}U_k^{+}$ for $k = 1,2,\ldots$.

$$M_0 = I \tag{3.5.11}$$

$$M_{k+1} = M_k - M_ku_k(u_k'M_ku_k)^{-1}u_k'M_k, \; u_k \notin R(U_k), \tag{3.5.12}$$

$$M_{k+1} = M_k - M_ku_k(1 + u_k'N_ku_k)^{-1}u_k'M_k, \; u_k \in R(U_K) \tag{3.5.13}$$

$$N_0 = 0 \tag{3.5.14}$$

$$N_{K=1} = N_k - N_ku_k(u_k'M_ku_k)^{-1}u_k'M_k - M_ku_k(u_k'M_ku_k)^{-1}u_kN_k \tag{3.5.15}$$

$$+ (1 + u_k'N_ku_k)M_ku_k(u_k'M_ku_k)^{-2}u_k'M_k, \qquad u_k \notin R(U_k),$$

$$N_{k+1} = N_k - N_ku_k(1 + u_k'N_ku_k)^{-1}u_k'N_k, \quad u_k \in R(U_k) \tag{3.5.16}$$

Since the operator M_k is an orthogonal projection operator, $u_k \in R(U_k)$ iff $M_ku_k = 0$ or equivalently iff $u_k'M_k'M_ku_k = u_k'M_ku_k = 0$. Therefore,

it is a straightforward matter to determine whether $u_k \in R(U_k)$ and consequently which formulas to use.

The matrix N_k is of interest here because it allows the calculation of M_k in the event $u_k \in R(U_k)$. If $u_k \notin R(U_k)$ for all $k < n$ then M_k is given by equations 3.5.11 and 3.5.12.

Before continuing it is worthwhile to observe that equations 3.5.11 and 3.5.12 define a well known conjugate gradient recursion formula (FLE 69). If the vectors u_k, $k = 0,\ldots,n-1$, are independent then this conjugate gradient formula can be interpreted as the recursive calculation of the operator $(I - U_k U_k^+)$. In the conjugate gradient algorithm the search direction is taken to be

$$v_k = (I - U_k U_k^+) g_k. \tag{3.5.17}$$

Because $(I - U_k U_k^+)$ projects vectors onto $R^{\perp}(U_k)$, $v_k \in R^{\perp}(U_k)$. If the objective function has a constant Hessian H then v_k is H orthogonal to the previous directions $V_k = [v_0,\ldots,v_{k-1}]$ since

$$v_k' H V_k = v_k' U_k = 0. \tag{3.5.18}$$

This, of course, is the fundamental property of the conjugate gradient algorithms.

Using the matrices M_k and N_k, the matrix $P_k = V_k U_k^+$ can be calculated recursively. From equations 3.5.2 and 3.5.3 we have

$$P_0 = 0 \tag{3.5.19}$$

$$P_{k+1} = P_k + (v_k - P_k u_k)(u_k' M_k u_k)^{-1} u_k' M_k, \quad u_k \notin R(U_k), \tag{3.5.20}$$

$$P_{k+1} = P_k + (v_k - P_k u_k)(1 + u_k' N_k u_k)^{-1} u_k' N_k, \quad u_k \in R(U_k). \tag{3.5.21}$$

The matrix P_k is the approximation to S_k over $\top$ obtained by choosing $Y_k = 0$ in equation 3.5.1. The entire family of approximations to S_k

over $\top$ is therefore given by

$$A_k = P_k + Y_k M_k \tag{3.5.22}$$

where Y_k is arbitrary.

If the matrix $Y_k = Y$ for $k = 1,2,\ldots$ then a particular sequence of approximations to S_k over $\top$ results.

$$A_0 = Y \tag{3.5.23}$$

$$A_{k+1} = P_{k+1} + YM_{k+1} \tag{3.5.24}$$

$$A_{k+1} = P_k + (v_k - P_k u_k)(u_k' M_k u_k)^{-1} u_k' M_k + YM_k \tag{3.5.25}$$

$$- YM_k u_k (u_k' M_k u_k)^{-1} u_k' M_k$$

$$= A_k + (v_k - A_k u_k)(u_k' M_k u_k)^{-1} u_k' M_k, \; u_k \notin R(U_k)$$

$$A_{k+1} = P_k + (v_k - P_k u_k)(1 + u_k' N_k u_k)^{-1} u_k' N_k + YM_k \tag{3.5.26}$$

$$- YM_k u_k (1 + u_k' N_k u_k)^{-1} u_k' M_k$$

$$= A_k + (v_k - A_k u_k)(1 + u_k N_k u_k)^{-1} u_k N_k, \; u_k \in R(U_k)$$

Formulas 3.5.20, 3.5.21, 3.5.22, 3.5.25, and 3.5.26 are distinct from the formulas of corollary 3.4.6. Even in the case where the vectors u_k, $k = 0,\ldots,n-1$, are independent these recursion formulas cannot be arrived at by a particular choice of the coefficients $a_{k1},\ldots,a_{k4}$ in corollary 3.4.6.

Application of the recursions 3.5.23, 3.5.24, and 3.5.25 in minimization algorithm 3.4.3 might yield a minimization algorithm with nice properties. However, the matrix A_k of the recursion approximates

S_k over $\top$ for all k. In minimization and root finding problems old data becomes invalid as the local character of the function changes. Therefore, it probably is desirable to have a procedure for removing elements from the data sets as the elements become old. In principle it should be possible to obtain recursive formulas for approximations to a data set generated by deleting an element from its predecessor. This would be an interesting area for further investigation.

In a process where an approximation to a data set with a large number of elements, relative to n, is to be recursively calculated the formulas 3.5.23, 3.5.25, and 3.5.26 are very useful since they only require the storage of three $n \times n$ matrices, two of which are symmetric. Further, if it is known that for all $k > \bar{k}$ the vectors $u_0,\ldots,u_{k-1}$ of the data set S_k span R^n, then $u_k \in R(U_k)$ for all $k > \bar{k}$, and only equations 3.5.16 and 3.5.26 are needed for the calculation of an approximation to S_k over $\top$. In this case then it is unnecessary to compute the matrix M_k. If the data sets contain n or fewer elements it seems more efficient to apply theorems 2.5.1 and 2.5.2 directly.

3.6 Summary

In this chapter the material of Chapter 2 has been utilized to unify a large class of root finding and minimization algorithms. Many other algorithms not discussed specifically herein, such as the Pshenichnii algorithm for minimization (PSH 69) and the Fletcher algorithm for root finding (FLE 68) can also be interpreted in the context of Chapters 2 and 3.

In section 3.3 the secant methods for root finding were shown under appropriate conditions to generate a sequence of matrices which emulate a sequence of data sets. It was pointed out that by considering

generalized secant methods involving approximations to the sequence of data sets it should be possible to get stronger convergence and rate of convergence results for the secant methods. Also, since the theory of Chapter 2 allows data sets in $R^n \times R^m$, $m \neq n$, it is possible to formulate secant-like algorithms for determining the root of $F(x)$: $R^n \to R^m$ or for minimizing the norm of $F(x)$.

In section 3.4 the relationship between theorem 2.5.5 and a number of proven minimization algorithms was considered. This led to a large new class of algorithms which, when applied to a quadratic function with positive definite Hessian, generate H conjugate directions, yield the matrix H^{-1} and minimize the function in n steps. Theorem 3.4.5 establishes the properties of these algorithms on a quadratic surface. Also in section 3.4 the special properties of the Murtagh and Sargent algorithm were discussed. It was shown that, because equations 3.4.22 and 3.4.23 of corollary 3.4.4 are automatically satisfied when the Murtagh and Sargent algorithm is applied to a quadratic surface, an exact linear minimization is not required at each stage.

The general choice of Z_i^1 and Z_i^2 given in corollary 3.4.3 allows the calculation of an emulating sequence, if one exists, when the sequence of data sets is arbitrary. This result may be of value in algorithms where the data sets differ by more than a single element. In section 3.5 a recursive expression was developed for the calculation of an approximating sequence when the data sets are generated by adjoining a single element at each stage. Such a formula could be useful when the number of elements in the data sets becomes large.

CHAPTER 4

FUNCTION MINIMIZATION USING THE PSEUDOINVERSE

4.1 Introduction, Definitions and Preliminary Lemmas

In this chapter a general algorithm will be developed for determining the minimum of a function $f(x) : D \subset R^n \to R^1$. As in Chapter 3 $g(x)$ and $H(x)$ will denote the first and second derivatives of $f(x)$ where they are defined. Under fairly weak hypotheses it will be shown that the general algorithm generates a sequence $\{x_k\}$ which converges to the minimum of $f(x)$. Under somewhat stronger conditions the terminal rate of convergence will be shown to be superlinear.

In developing these results three norms on R^n will be utilized:

$$\|x\|_1 = \sum_{i=1}^{n} |x^i| \tag{4.1.1}$$

$$\|x\|_2 = \left(\sum_{i=1}^{n} (x^i)^2 \right)^{1/2} \tag{4.1.2}$$

$$\|x\|_\infty = \max_{i=1,\ldots,n} |x^i| \tag{4.1.3}$$

where x^i is the i th component of x. When an arbitrary norm will suffice the subscript will be omitted. Associated with each of these norms will be the matrix norm on the space T of all n x n matrices defined for $A \in \mathsf{T}$ by

$$\|A\| \quad \sup_{x \in R^n} \frac{\|Ax\|}{\|x\|} . \tag{4.1.4}$$

Matrix norms will be subscripted to correspond to the vector norm which generates them. We will make use of two results concerning norms. First, the matrix norm associated with the vector norm $\|x\|_\infty$ is given by

$$\|A\|_\infty = \max_{k=1,\ldots,n} \sum_{j=1}^{n} |a_{kj}|. \qquad (4.1.5)$$

Second, for any two vector or matrix norms $\|\cdot\|_a$ and $\|\cdot\|_b$ there exist constants $C_1 > 0$ and $C_2 > 0$ such that

$$C_1 \|x\|_a \leq \|x\|_b \leq C_2 \|x\|_a \qquad (4.1.6)$$

for all $x \in R^n$ or

$$C_1 \|A\|_a \leq \|A\|_b \leq C_2 \|A\|_a \qquad (4.1.7)$$

for all $A \in R^{n^2}$. A proof of these results may be found in (ORT 70, section 2.2).

The following standard definitions will be needed in the development.

Definition 4.1.1 The open sphere of radius $r > 0$ centered at $\hat{x} \in R^n$, denoted by $S(\hat{x},r)$, is the set of all $x \in R^n$ such that $\|x - \hat{x}\|_2 < r$.

Definition 4.1.2 The interior points of a set $A \subset R^n$, denoted by $\underline{A}$, are all points of A contained in some open sphere contained in A.

Definition 4.1.3 A set $A \subset R^n$ is closed if $\lim_{k \to \infty} x_k = \hat{x}$ and $\{x_k\} \subset A$ implies that $\hat{x} \in A$.

Definition 4.1.4 A set $A \subset R^n$ is bounded if $A \subset S(0,r)$ for some $r > 0$.

Definition 4.1.5 A set $A \subset R^n$ is compact if it is closed and bounded.

Definition 4.1.6 A function $F(x) : D \subset R^n \to R^m$ is continuous at $\hat{x} \in \underline{D}$ if given $\varepsilon > 0$ $\exists$ a $\delta > 0$ $\ni$ $F(x) \in S(F(\hat{x}),\varepsilon)$ whenever $x \in S(\hat{x},\delta)$.

Definition 4.1.7 A function $F(x) : D \subset R^n \to R^m$ is differentiable at $\hat{x} \in \underline{D}$ if given $\varepsilon > 0$ $\exists$ a linear transformation $F'(\hat{x}) : R^n \to R^m$ and a $\delta > 0$ $\ni$

$$\|F(x) - F(\hat{x}) - F'(\hat{x})(x - \hat{x})\| < \varepsilon \|x - \hat{x}\| \qquad (4.1.8)$$

whenever $x \in S(\hat{x},\delta)$. The linear transformation $F'(x)$ is the derivative of $F(x)$ at $\hat{x}$.

Definition 4.1.8 The derivative of $F(x) : D \subset R^n \to R^m$ at $\hat{x} \in \underline{D}$ is strong if given $\varepsilon > 0$ $\exists$ a $\delta > 0$ $\ni$

$$\|F(x) - F(y) - F'(\hat{x})(x - y)\| < \varepsilon \|x - y\| \qquad (4.1.9)$$

for all $x,y \in S(\hat{x},\delta) \subset \underline{D}$.

Definition 4.1.9 A function $F(x) : D \subset R^n \to R^m$ is continuously differentiable at $\hat{x} \in \underline{D}$ if $\exists$ a $\delta > 0$ $\ni$ $F(x)$ is differentiable at all points of $S(\hat{x},\delta) \subset \underline{D}$ and if $F'(x) : S(\hat{x},\delta) \to R^{n \times m}$ is continuous at $\hat{x}$.

To require a strong derivative at a point is less restrictive than to require a continuous derivative at a point since functions exist which have a strong derivative at a point $\hat{x}$ but which are not differentiable at all points of any open sphere containing $\hat{x}$. However, in many practical situations the concepts are equivalent as the following lemma shows.

Lemma 4.1.10 Suppose $F(x) : D \subset R^n \to R^m$ is differentiable at each point of $S(\hat{x},\delta) \subset \underline{D}$ for some $\delta > 0$. Then $F'(\hat{x})$ is strong iff $F'(x)$ is continuous at $\hat{x}$.

A proof of this result may be found in (ORT 70, lemma 3.2.10).

In proving the rate of convergence results it is essential that the derivative of $f(x)$ have certain properties in a neighborhood of a minimum. The following lemma establishes conditions for these properties.

Lemma 4.1.11 Suppose $f(x) : D \subset R^n \to R^1$ is twice differentiable at $\hat{x} \in \underline{D}$ and that $H(\hat{x})$ is positive definite and strong. Then there exist constants $C_1 > 0$ and $C_2 > 0$ and an open sphere $S(\hat{x},\delta)$ such that

$$C_1 \|x - y\| \leq \|g(x) - g(y)\| \leq C_2 \|x - y\| \qquad (4.1.10)$$

$$\forall\, x,y \in S(\hat{x},\delta) \subset \underline{D}.$$

Proof: By the definition of a strong derivative, given any

$\varepsilon > 0$ $\exists$ a $\delta > 0$ $\ni$

$$\|g(x) - g(y) - H(\hat{x})(x - y)\| \leq \varepsilon \|x - y\| \qquad (4.1.11)$$

$\forall$ $x,y \in S(\hat{x},\delta) \subset \underline{D}$. This implies that

$$\|H(x)(x - y)\| - \varepsilon \|x - y\| \leq \|g(x) - g(y)\| \leq \qquad (4.1.12)$$

$$\|H(\hat{x})(x - y)\| + \varepsilon \|x - y\|$$

$\forall$ $x,y \in S(\hat{x},\delta)$. Further, because $H(\hat{x})$ is positive definite

$$(\overline{C}_1 - \varepsilon) \|x - y\| \leq \|g(x) - g(y)\| \leq (\overline{C}_2 + \varepsilon) \|x - y\| \qquad (4.1.13)$$

where $\overline{C}_1 > 0$ and $\overline{C}_2 > 0$. If ε is chosen less than $\overline{C}_1$ then the choices $C_1 = \overline{C}_1 - \varepsilon$ and $C_2 = \overline{C}_2 + \varepsilon$ yield the result of the lemma. Q.E.D.

4.2 Fundamental Definitions and Lemmas

The results of this section form the foundation for the construction of the general minimization algorithm. In what follows the matrices U and V can be viewed as representing a data set S as discussed in chapter 2. We begin with a lemma.

Lemma 4.2.1 Suppose $f(x) : D \subset R^n \to R^1$ is twice differentiable at $\hat{x} \in \underline{D}$ and that $H(\hat{x})$ is positive definite and strong. Let U be any $n \times m$ matrix, $m > 0$. For $\alpha \in (0,\infty)$ sufficiently small $\exists$ a $\delta > 0$ $\ni$ if $x,y \in S(\hat{x},\delta)$ and if $v = x - y \in R^1(U)$ then

$$\|(I - UU^+)u\| \geq \alpha \|u\| \qquad (4.2.1)$$

where $u = g(x) - g(y)$.

Proof: If $v = 0$ the lemma is clearly satisfied. Assume then that $v \neq 0$. First we have

$$\|(I - UU^+)u\| = \|v\| \|(I - UU^+)u\| \|v\|^{-1} \qquad (4.2.2)$$

$$\geq C_1 v'(I - UU^+)u \|v\|^{-1}$$

where the constant $C_1 > 0$ depends on the norm $\|\cdot\|$. Since $v \in R^{\perp}(U)$ it follows that

$$\|(I - UU^{+})u\| \geq C_1 v'u \|v\|^{-1}. \tag{4.2.3}$$

Letting $e = u - H(\hat{x})v$

$$\|(I - UU^{+})u\| \geq C_1 v'H(\hat{x})v \|v\|^{-1} + C_1 v'e \|v\|^{-1}. \tag{4.2.4}$$

Because $H(\hat{x})$ is positive definite

$$\|(I - UU^{+})u\| \geq C_2 \|v\| + C_1 v'e \|v\|^{-1} \tag{4.2.5}$$

for some $C_2 > 0$. By lemma 4.1.11 $\exists$ a $C_3 > 0$ dependent on $H(\hat{x})$ and a $\delta_1 > 0$ $\ni$

$$\|v\| \geq C_3 \|u\| \tag{4.2.6}$$

$\forall$ $x,y \in S(\hat{x},\delta_1)$. Because $g(x)$ has a strong derivative at $\hat{x}$, for any $\varepsilon > 0$ $\exists$ a $\delta_2 > 0$ $\ni$

$$\|e\| = \|u - H(\hat{x})v\| \leq \varepsilon \|v\| \tag{4.2.7}$$

$\forall$ $x,y \in S(\hat{x},\delta_2)$. Further, by lemma 4.1.11 $\exists$ a $\delta_3 > 0$ and a $C_4 > 0$ $\ni$

$$\|v\| \leq C_4 \|u\| \tag{4.2.8}$$

$\forall$ $x,y \in S(\hat{x},\delta_3)$. It follows that for all $x,y \in S(\hat{x},\delta_4)$, $\delta_4 = \min\{\delta_1,\delta_2,\delta_3\}$,

$$\|(I - UU^{+})u\| \geq C_2C_3 \|u\| - C_1 |v'e| \|v\|^{-1} \tag{4.2.9}$$

$$\geq C_2C_3 \|u\| - C_1 \|v\| \|e\| \|v\|^{-1}$$

$$\geq C_2C_3 \|u\| - C_1 \|e\|$$

$$\geq (C_2C_3 - C_1C_4\ \varepsilon) \|u\|.$$

If ε is chosen less than $C_2C_3(C_1C_4)^{-1}$ the choice $\delta = \delta_4$ and $0 < \alpha < (C_2C_3 - C_1C_4\ \varepsilon)$ yields the result of the lemma. Q.E.D.

Remark 4.2.2 A review of the proof of lemma 4.2.11 will verify that the

maximum value of α depends only on the norm chosen and on the matrix $H(\hat{x})$.

Remark 4.2.3 When the norm of lemma 4.2.1 is $\|.\|_2$, $\|(I - UU^+)u\|_2 \leq \|u\|_2$ for any u and therefore only $\alpha \in (0,1]$ need be considered. This follows because the operator $I - UU^+$ is the projection operator whose range is $R^{\perp}(U)$.

Next a family of data sets will be defined which has properties useful in proving the rate of convergence results. The general minimization algorithm will be constructed to choose only data sets S_k from this family of data sets.

Definition 4.2.4 Suppose $f : D \subset R^n \to R^1$ is differentiable on $\underline{D}$. Let $S(\hat{x},\delta,\alpha)$, $x \in \underline{D}$, $\delta > 0$, $\alpha \in (0,1]$, be the family of data sets $S \equiv (U,V)$ with the following properties.

1. Each pair (U,V) consists of a matrix U with n rows and m columns and a matrix V with n rows and m columns where m may have any value in $\{1,\ldots,n\}$.
2. The columns v_i, $i = 1,\ldots,m$, of V are nonzero and of the form $x_i - y_i$ where $x_i, y_i \in S(\hat{x},\delta) \cap \underline{D}$.
3. The columns u_i, $i = 1,\ldots,m$, of U are nonzero and of the form $g(x_i) - g(y_i)$ where $x_i - y_i$ is the column v_i of V.
4. The columns u_i, $i = 2,\ldots,m$, of U are such that

$$\|(I - U_{i-1}U_{i-1}^+)u_i\|_2 \geq \alpha \|u_i\|_2 \qquad (4.2.10)$$

where U_{i-1} is the n x(i-1) matrix whose columns are the first i-1 columns of U with the same ordering as the columns of U.

As remark 4.2.3 indicates, it is enough to consider $\alpha \in (0,1]$ since if $\alpha > 1$ condition 4.2.10 is never satisfied. We next verify that, under appropriate conditions on $f(x)$, the family $S(\hat{x},\delta,\alpha)$ is

not empty.

Lemma 4.2.5 Suppose $f(x)$: $D \subset R^n \to R^1$ is differentiable on $\underline{D}$, twice differentiable at $\hat{x} \in \underline{D}$ and that $H(\hat{x})$ is positive definite and strong. Then for $\alpha \in (0,1]$ sufficiently small and for any $\delta > 0$, the family $S(\hat{x},\delta,\alpha)$ is not empty and in fact contains data sets with m elements for $m = 1,\ldots,n$.

Proof: By lemma 4.1.11 $\exists$ a $\delta_1 > 0$ $\ni$ $u = g(x) - g(\hat{x}) \neq 0$ whenever $v = x - \hat{x} \neq 0$ and $x \in S(\hat{x},\delta_1)$. Therefore the set $S(\hat{x},\delta,\alpha)$ is non-empty for any $\delta > 0$ since any pair of one column matrices (u,v) chosen so that $x \in S(\hat{x},\delta) \cap S(\hat{x},\delta_1)$ is an element of $S(\hat{x},\delta,\alpha)$. If (U_i,V_i) is a pair in $S(\hat{x},\delta,\alpha)$ with $i < n$ columns and if α is sufficiently small, a pair $(U_{i+1},V_{i+1}) \in S(\hat{x},\delta,\alpha)$ with $i+1$ columns can be constructed as follows. Let $V_{i+1} = [V_i, v_i]$ and $U_{i+1} = [U_i, u_i]$ where $v_i = x_i - \hat{x} \in R^{\perp}(U_i)$, $x_i \neq \hat{x}$, $x_i \in S(\hat{x},\delta)$, $u_i = g(x_i) - g(\hat{x}) \neq 0$, and $\|(I - U_iU_i^+)u_i\|_2 \geq \alpha \|u_i\|_2$. That such a choice can be made follows from lemmas 4.1.11 and 4.2.1. Since this construction can be accomplished for $i = 1,\ldots,n-1$, the family $S(\hat{x},\delta,\alpha)$ contains data sets with m elements, $m = 1,\ldots,n$. Q.E.D.

The following lemma establishes a bound on a matrix generated from elements of $S(\hat{x},\delta,\alpha)$ which will be needed in the rate of convergence proof.

Lemma 4.2.6 Assume $f(x)$: $D \subset R^n \to R^1$ is twice differentiable at $\hat{x} \in \underline{D}$ and that $H(\hat{x})$ is positive definite and strong. Given any $\alpha \in (0,1]$ for every $\varepsilon > 0$ $\exists$ a $\delta = \delta(\varepsilon) > 0$ $\ni$ $\forall$ data sets $(U,V) \in S(\hat{x},\delta,\alpha)$

$$\|(U - H(\hat{x})V)U^+\| \leq C(\alpha)\,\varepsilon \tag{4.2.11}$$

where the constant C depends only on α.

Proof: Because of the definition of $S(\hat{x},\delta,\alpha)$ if $(U,V) \in S(\hat{x},\delta,\alpha)$ the columns of U are linearly independent. Therefore, by theorem 2.5.1

$$U_i^+ = [U_{i-1},u_i]^+ = \begin{bmatrix} U_{i-1}^+(I - u_i(z_i'z_i)^{-1}z_i') \\ (z_i'z_i)^{-1}z_i' \end{bmatrix} \qquad (4.2.12)$$

where u_i is the i th column of U, U_i (U_{i-1}) is the matrix whose columns are the first i (i-1) columns of U with the same ordering as the columns of U and

$$z_i = (I - U_iU_i^+)u_i. \qquad (4.2.13)$$

Using the explicit representation given in equation 4.1.5 for the matrix norm $\| \ \|_\infty$ we have that if $(U,V) \in S(\hat{x},\delta,\alpha)$ and u_i is the i th column of U

$$\begin{aligned} \|I - u_i(z_iz_i)^{-1}z_i'\|_\infty &= \sum_{j=1}^{k-1} |u_i^k z_i^j|(z_i'z_i)^{-1} \\ &\quad + |1 - u_i^k z_i^k (z_i'z_i)^{-1}| \\ &\quad + \sum_{j=k+1}^{n} |u_i^k z_i^j|(z_i'z_i)^{-1} \end{aligned} \qquad (4.2.14)$$

for some component u_i^k of u_i. Thus

$$\begin{aligned} \|I - u_i(z_i'z_i)z_i'^{-1}\|_\infty &\le \sum_{j=1}^{n} |u_i^k z_i^j|(z_i'z_i)^{-1} + 1 \\ &\le \max_{k=1,\ldots,n} |u_i^k| \sum_{j=1}^{n} |z_i^j|(z_i'z_i)^{-1} + 1 \\ &\le \|u_i\|_\infty \|z_i\|_1 \|z_i\|_2^{-2} + 1 \\ &\le C_1 \|u_i\| \, \|z_i\|^{-1} + 1 \end{aligned} \qquad (4.2.15)$$

where the constant C_1 depends on the particular vector norm chosen for u_i and z_i. Because of condition 4.2.10, $\|z_i\|_2 \ge \alpha \|u_i\|_2$ and

therefore

$$\|I - u_i(z_i'z_i)^{-1}z_i'\|_\infty \leq C_2\,\alpha^{-1} + 1 \tag{4.2.16}$$

for some constant C_2. Further, from equation 4.1.7

$$\|I - u_i(z_i'z_i)^{-1}z_i'\| \leq C_3(C_2\,\alpha^{-1} + 1) \tag{4.2.17}$$

where C_3 depends on the matrix norm chosen. Next, let u_i^+ denote the i th row of the matrix U^+ and notice that repeated applications of theorem 2.5.1 yield

$$\|u_i^+\| = \Big\|\prod_{j=i+1}^{m} z_i'(I - u_j(z_j'z_j)^{-1}z_j')(z_j'z_j)^{-1}\Big\|, \quad i < m, \tag{4.2.18}$$

$$\|u_m^+\| = \|(z_m'z_m)^{-1}z_m'\| \tag{4.2.19}$$

where m is the number of columns of U. From equation 4.2.17 it can be concluded that

$$\|u_i^+\| \leq C_4\,(C_3(C_2\,\alpha^{-1} + 1))^{m-1}\,\|z_i\|^{-1} \tag{4.2.20}$$

$$\leq C_4\,(C_3(C_2\,\alpha^{-1} + 1))^{m-1}\,C_5\,\alpha^{-1}\,\|u_i\|^{-1}$$

where the constants C_4 and C_5 depend on the choice of norm and where C_3 is chosen greater than 1. It follows by lemma 4.1.11 that for some $\delta_1 > 0$ all columns u_i and v_i of any pair $(U,V) \in S(\hat{x},\delta_1,\alpha)$ are such that $\|v_i\| \leq C_6\|u_i\|$ and therefore

$$u_i^+ \leq C_4(C_3(C_2\,\alpha^{-1} + 1))^{m-1}\,C_5C_6\,\alpha^{-1}\,\|v_i\|^{-1}. \tag{4.2.21}$$

Because $H(\hat{x})$ is strong, given $\varepsilon < 0$ $\exists$ a $\delta_2 = \delta_2(\varepsilon) > 0$ $\ni$ all columns u_i and v_i of any pair $(U,V) \in S(\hat{x},\delta_2,\alpha)$ are such that

$$\|u_i - H(\hat{x})v_i\| \leq \varepsilon\,\|v_i\|. \tag{4.2.22}$$

Finally, since

$$\|(U - H(x)V)U^+\| = \left\|\sum_{i=1}^{m} (u_i - H(\hat{x})v_i)u_i^+\right\| \qquad (4.2.23)$$

$$\leq \sum_{i=1}^{m} \|u_i - H(\hat{x})v_i\| \quad \|u_i^+\|$$

we have that for any $\varepsilon > 0$ $\exists$ a $\delta(\varepsilon) = \min\{\delta_1, \delta_2(\varepsilon)\}$ $\ni$

$$\|(U - H(x)V)U^+\| \leq \sum_{i=1}^{m} \varepsilon \|v_i\| C_4(C_3(C_2 \alpha^{-1} + 1))^{m-1} \qquad (4.2.24)$$

$$C_5C_6 \alpha^{-1} \|v_i\|^{-1}$$

$$\leq \varepsilon m C_4(C_3(C_2 \alpha^{-1} + 1))^{m-1} C_5C_6 \alpha^{-1}. \qquad \text{Q.E.D.}$$

The next lemma shows that if δ is sufficiently small and if the vector ω is nearly in $R(U)$, then the vector $p^* \equiv VU^+\omega$ nearly equals $H^{-1}(\hat{x})\omega$ when $(U,V) \in S(\hat{x},\delta,\alpha)$.

Lemma 4.2.7 Assume $f : D \subset R^n \to R^1$ is twice differentiable at $\hat{x} \in D$ and that $H(\hat{x})$ is positive definite and strong. Let $\alpha \in (0,1]$ and $\beta \geq 0$ be given and consider the vectors

$$\bar{p} \equiv (I - UU^+)\omega \qquad (4.2.25)$$

$$p^* \equiv VU^+\omega \qquad (4.2.26)$$

$$h \equiv p^* - H^{-1}(\hat{x})\omega \qquad (4.2.27)$$

where $(U,V) \in S(\hat{x},\delta,\alpha)$ and $\omega \in R^n$. For every $\varepsilon > 0$ $\exists$ a $\delta(\varepsilon) > 0$ $\ni$ $\forall$ $(U,V) \in S(\hat{x},\delta,\alpha)$ and any $\omega \in R^n$ if

$$\|\bar{p}\| \leq \beta \|\omega\| \qquad (4.2.28)$$

then

$$\|h\| \leq (\beta + C\varepsilon) \|H^{-1}(\hat{x})\| \quad \|\omega\| \qquad (4.2.29)$$

where the constant C depends only on α.

Proof: If $\omega = 0$ the lemma is clearly satisfied, so assume $\omega \neq 0$.

We have

$$\|h\| = \|(VU^+ - H^{-1}(\hat{x}))\omega\| \qquad (4.2.30)$$

$$= \|H^{-1}(\hat{x})(H(\hat{x})VU^+ - I)\omega\|$$

$$\leq \|H^{-1}(\hat{x})\| \|(H(\hat{x})VU^+ - I)\omega\|$$

$$\leq \|H^{-1}(\hat{x})\| \|(H(\hat{x})V - U)U^+\omega - (I - UU^+)\omega\|$$

$$\leq \|H^{-1}(\hat{x})\| (\|(H(\hat{x})V - U)U^+\| \|\omega\|$$

$$+ \|(I - UU^+)\omega\|).$$

By lemma 4.2.6, for every $\epsilon > 0$ $\exists$ a $\delta > 0$ $\ni$ $\forall$ pairs $(U,V) \in S(\hat{x},\delta,\alpha)$

$$\|h\| \leq \|H^{-1}(\hat{x})\| (C \epsilon \|\omega\| + \beta \|\omega\|). \qquad (4.2.31)$$

Q.E.D.

Remark 4.2.8 If $(U,V) \in S(\hat{x},\delta,\alpha)$ and U and V have n columns then U is nonsingular and $\bar{p} = 0$ for every $\omega \in R^n$. In this case $\beta = 0$ satisfies equation 4.2.29 for any $\omega \in R^n$ and equation 4.2.31 reduces to

$$\|h\| \leq C \epsilon \|H^{-1}(\hat{x})\| \|\omega\|. \qquad (4.2.32)$$

Also, if the norm of equation 4.2.28 is $\|\ \|_2$ then by remark 4.2.3 β may be chosen in $[0,1]$ without loss of generality.

Finally we show that for δ sufficiently small if the vector ω is "nearly in" $R(U)$ then the angle between p^* of the previous lemma and ω is bounded less than 90° when (U,V) is any pair in $S(\hat{x},\delta,\alpha)$.

Lemma 4.2.9 Assume $f : D \subset R^n \to R^1$ is twice differentiable at $\hat{x} \in D$ and that $H(\hat{x})$ is positive definite and strong. Let $\alpha \in (0,1]$ be given and consider the vectors

$$\bar{p} \equiv (I - UU^+)\omega \qquad (4.2.33)$$

and

$$p^* \equiv VU^+\omega \qquad (4.2.34)$$

where $(V,U) \in S(\hat{x},\delta,\alpha)$ and $\omega \in R^n$, $\omega \neq 0$. For $\beta \geq 0$ sufficiently small $\exists$ a $\delta > 0$ $\ni$ if

$$\|\bar{p}\| \leq \beta\|\omega\| \tag{4.2.35}$$

then

$$\frac{\omega' p^*}{\|\omega\| \ \|p^*\|} > \beta . \tag{4.2.36}$$

Proof: Notice first that with $h = p^* - H^{-1}(\hat{x})\omega$

$$\frac{\omega' H^{-1}(\hat{x})\omega}{\|\omega\| \ \|H^{-1}(\hat{x})\omega\|} - \frac{\omega' p^*}{\|\omega\| \ \|p^*\|} = \frac{\omega' H^{-1}(\hat{x})\omega - \omega' p^* \ \|H^{-1}(\hat{x})\omega\| \ \|p^*\|^{-1}}{\|\omega\| \ \|H^{-1}(\hat{x})\omega\|}$$

$$= \frac{-\omega' h + \omega' p^* - \omega' p^* \ \|H^{-1}(\hat{x})\omega\| \ \|p^*\|^{-1}}{\|\omega\| \ \|H^{-1}(\hat{x})\omega\|} \tag{4.2.37}$$

$$\leq \frac{|\omega' h| + |\omega' p^*| \, |1 - \|H^{-1}(\hat{x})\omega\| \ \|p^*\|^{-1}|}{\|\omega\| \ \|H^{-1}(\hat{x})\omega\|}$$

$$\leq \frac{\|h\|}{\|H^{-1}(\hat{x})\omega\|} + \frac{|\,\|p^*\| - \|H^{-1}(\hat{x})\omega\|\,|}{\|H^{-1}(\hat{x})\omega\|} .$$

Utilizing the result of lemma 4.2.7 and the fact that $\|x - y\| \leq c$ implies that $|\,\|x\| - \|y\|\,| \leq c$, we have that given $\alpha \in (0,1]$ for every $\varepsilon > 0$ $\exists$ a $\delta(\varepsilon) > 0$ $\ni$ $\forall$ $(U,V) \in S(\hat{x},\delta,\alpha)$

$$\|\bar{p}\| \leq \beta \|\omega\| \quad , \ \omega \neq 0, \ \omega \in R^n$$

implies

$$\frac{\omega' H^{-1}(\hat{x})\omega}{\|\omega\| \ \|H^{-1}(\hat{x})\omega\|} - \frac{\omega' p^*}{\|\omega\| \ \|p^*\|} \leq 2(\beta + C_1 \varepsilon) \frac{\|H^{-1}(\hat{x})\| \ \|\omega\|}{\|H^{-1}(\hat{x})\omega\|}$$

$$\leq C_2(\beta + C_1 \varepsilon) \tag{4.2.38}$$

where $\beta \geq 0$, $C_1 > 0$ depends only on α and $C_2 > 0$ depends only on $H(\hat{x})$. Because $H(\hat{x})$ is positive definite $\exists$ a constant $C_3 > 0$ $\ni$ $\forall$ $\omega \neq 0$

$$\frac{\omega' H^{-1}(\hat{x})\omega}{\|\omega\| \ \|H^{-1}(\hat{x})\omega\|} \geq C_3. \tag{4.2.39}$$

Therefore, given $\alpha \in (0,1]$ for every $\varepsilon > 0$ $\exists$ a $\delta > 0$ ∋ $(U,V) \in S(\hat{x},\delta,\alpha)$ and any non-zero $\omega \in R^n$,

$$\|\bar{p}\| \leq \beta \|\omega\|$$

implies

$$\frac{\omega' p^*}{\|\omega\| \ \|p^*\|} \geq C_3 - C_2(\beta + C_1\varepsilon). \tag{4.2.40}$$

Choosing β less than $C_3(1 + C_2)^{-1}$ any δ corresponding to an ε less than $(C_3 - (1 + C_2)\beta)(C_2C_1)^{-1}$ yields

$$\begin{aligned}\frac{\omega' p^*}{\|\omega\| \ \|p^*\|} &> C_3 - C_2(\beta + C_1(C_3 - (1 + C_2)\beta)(C_2C_1)^{-1}) \\ &> C_3 - C_2\beta - C_3 + (1 + C_2)\beta \\ &> \beta.\end{aligned} \tag{4.2.41}$$

Notice that β must be less than $C_3(1 + C_2)^{-1}$ in order for there to exist a positive ε less than $(C_3 - (1 + C_2)\beta)(C_2C_1)^{-1}$. Q.E.D.

Remark 4.2.10 If the norm is $\| \ \|_2$ then the inequality

$$\|\bar{p}\|_2 \leq \beta \|\omega\|_2 \tag{4.2.42}$$

is equivalent to the inequality

$$\frac{\bar{p}'\omega}{\|\bar{p}\| \ \|\omega\|} \leq \beta. \tag{4.2.43}$$

This may be seen as follows. If $\|\bar{p}\|_2 \leq \beta \|\omega\|_2$ then

$$\|\bar{p}\|_2^2 \leq \beta \|\omega\|_2 \|\bar{p}\|_2. \tag{4.2.44}$$

It follows that

$$\frac{\omega'(I - UU^+)'(I - UU^+)\omega}{\|\omega\|_2 \|\bar{p}\|_2} \leq \beta. \qquad (4.2.45)$$

However

$$\omega'(I - UU^+)'(I - UU^+)\omega = \omega'(I - UU^+)\omega \qquad (4.2.46)$$

$$= \omega'\bar{p}$$

which leads to the conclusion that

$$\frac{\omega'\bar{p}}{\|\omega\|_2 \|\bar{p}\|_2} \leq \beta. \qquad (4.2.47)$$

The argument is reversible.

4.3 General Minimization Algorithm

We now propose a general algorithm for the minimization of a function $f(x) : D \subset R^n \to R^1$. The algorithm is divided into three stages. In Stage I a direction is chosen for a linear search based on a data set S_k and the gradient $g(x_k)$ and an approximate linear minimization is performed starting at x_k. In Stage II a tentative new data set S_{k0} is chosen which lies in an appropriate family $S(\hat{x},\delta,\alpha)$. In Stage III, if it is necessary, the data set S_{k0} is augmented in stages S_{kj}, $j = 1,2,\ldots$, to form the data set S_{k+1} by choosing a sequence of auxiliary steps which insure that for k sufficiently large the data set S_{k+1} contains n elements.

The general algorithm is constructed to allow as much latitude as possible in the choice of a sequence of data sets S_k. There is also, under certain circumstances, latitude in the choice of a direction for the linear search. Convergence and rate of convergence results will be presented for the general algorithm. In Chapter 5 these results are supplemented by numerical experiments with algorithms constructed within

the framework of the general minimization algorithm.

Before proceeding a definition is needed.

Definition 4.3.1 A function $\sigma(t) : [0,\infty) \to [0,\infty)$ is a forcing function if $\lim_{k \to \infty} \sigma(t_k) = 0$ implies that $\lim_{k \to \infty} t_k = 0$.

We now state the general minimization algorithm.

Algorithm 4.3.2 Suppose the function $f(x) : D \subset R^n \to R^1$ is to be minimized on $\underline{D}$. Let $x_0 \in \underline{D}$ be given and set $k = 0$. Let K_1 be any finite subset of I_p, the positive integers, and let K_2 by any subset of I_p. Let $\alpha \in (0,1]$, $\beta \in (0,1]$ and $\ell \in I_p$ be given. Also, let the forcing functions $\sigma_1(\cdot)$, $\sigma_2(\cdot)$ and $\sigma_3(\cdot)$ be given.

Stage I

1. If $g_k = g(x_k) = 0$ stop.
2. If U_k and V_k are not defined, go to step 5 of Stage I.
3. If $\bar{p}_k \equiv (I - U_k U_k^+) g_k \neq 0$ and if $\bar{p}_k$ satisfies $\bar{p}_k' g_k (\|\bar{p}_k\|_2 \|g_k\|_2)^{-1} \geq \beta$, set $p_k = \bar{p}_k$ and go to step 7, Stage I.
4. If $p_k^* \equiv V_k U_k^+ g_k \neq 0$ and if p_k^* satisfies $p_k^{*'} g_k (\|p_k^*\|_2 \|g_k\|_2)^{-1} \geq \beta$, set $p_k = p^*$ and go to step 6, Stage I.
5. Set p_k equal to any nonzero vector such that $p_k' g_k (\|p_k\|_2 \|g_k\|_2)^{-1} \geq \beta$ and go to step 7, Stage I.
6. If $f(x_k) - f(x_k - p_k) \geq \sigma_1(\|g_k\|_2)$, set $x_{k+1} = x_k - p_k$, $\Delta x_k = x_{k+1} - x_k$, $\Delta g_k = g_{k+1} - g_k$ and go to Stage II.
7. Set $x_{k+1} = x_k - \lambda_k p_k$, where λ_k is chosen by a steplength algorithm such that $f(x_{k+1}) - f(x_k) \geq \sigma_2(p_k' g_k \|p_k\|_2^{-1})$ $\Delta x_k = x_{k+1} - x_k$, $\Delta g_k = g_{k+1} - g_k$ and go to Stage II.

Stage II

1. Set $j = 0$ and let V_{kj}, U_{kj} be any two $n \times m$, $m \in \{1,\ldots,n\}$,

matrices with the following properties.

a. The columns of V_{kj} are a non-empty subset of the vectors $\Delta x_{\bar{k}}$ and (when defined) $\Delta x_{\bar{k}\bar{j}}$, $k - \ell \leq \bar{k} < k$, $\bar{j} \geq 0$. The columns of U_{kj} are nonzero and are the corresponding vectors $\Delta g_{\bar{k}}$ and (or) $\Delta g_{\bar{k}\bar{j}}$ with the same ordering as the columns of V_{kj}. (The specific ordering of the columns of V_{kj} and U_{kj} is of no consequence. However, it is important that the correspondence between the columns of U_{kj} and V_{kj} be preserved.)

b. The i th column u_i of U_{kj} is such that

$$\|(I - U_{kj}^{i-1} (U_{kj}^{i-1})^{+})u_i\|_2 \geq \alpha \|u_i\|, \quad i = 2,\ldots,m, \qquad (4.3.1)$$

where U_k^{i-1} is the matrix consisting of the first i-1 columns of U_{kj} with the same ordering as the columns of U_{kj}.

2. If $k \in K_1$, either set $U_{k+1} = U_{kj}$, $V_{k+1} = V_{kj}$, $k = k+1$ and return to Stage I or leave U_{k+1} and V_{k+1} undefined, set $k = k+1$ and return to Stage I, step 1. If $k \notin K_1$ go to Stage III.

Stage III

1. If U_k and V_k are undefined, or if U_{kj} has n columns, or if U_{kj} has more columns than U_k, or if U_{kj} has as many columns as U_k and $k \in K_2$, set $U_{k+1} = U_{kj}$, $V_{k+1} = V_{kj}$, $k = k+1$ and return to Stage I, step 1.

2. Set p_{kj} equal to any nonzero vector in $R^1(U_{kj})$ and set $x_{kj} = x_k - \lambda_{kj} p_{kj}$ where $\lambda_{kj} > 0$ is chosen so that $\sigma_3(\|x_{kj} - x_k\|) \leq \|g_k\|$. Set $\Delta x_{kj} = x_{kj} - x_k$ and $\Delta g_{kj} = g_{kj} - g_k$.

3. If $\Delta g_{kj} \neq 0$ and $\|(I - U_{kj}U_{kj}^{+})\Delta g_{kj}\|_2 \geq \alpha \|\Delta g_{kj}\|_2$, set $U_{k(j+1)} = [U_{kj}, \Delta g_{kj}]$, $V_{k(j+1)} = [V_{kj}, \Delta x_{kj}]$, $j = j+1$ and go to step 1, Stage III.

4. Set $U_{k+1} = U_{kj}$, $V_{k+1} = V_{kj}$, $k = k+1$ and return to Stage I, step 1.

Remark 4.3.3 Stage III of algorithm 4.3.2 can be formulated in a slightly different manner to make better use of the auxiliary points x_{kj}. Step 1 of Stage III remains unchanged. Steps 2, 3 and 4 of Stage III are replaced by the following procedure.

2. If $\bar{p}_{kj} \equiv (I - U_{kj}U_{kj}^{+})g_k \neq 0$ and $\bar{p}_{kj}'g_k(\|\bar{p}_{kj}\|_2 \|g_k\|_2)^{-1} \geq \beta$, set $x_{kj} = x_k - \lambda_{kj}\bar{p}_{kj}$ where λ_{kj} is chosen by a step-length algorithm such that $f(x_k) - f(x_{kj}) \geq \sigma_2(\bar{p}_{kj}'g_k \|g_k\|^{-1})$. Set $\Delta x_{kj} = x_{kj} - x_k$, $\Delta g_{kj} = g_{kj} - g_k$, $x_k = x_{kj}$ and go to step 4.

3. Set $x_{kj} = x_k - \lambda_{kj}p_{kj}$ where p_{kj} is any vector in $R^{\perp}(U_{kj})$ and $\lambda_{kj} > 0$ is such that $\sigma_3(\|x_{kj} - x_k\|) \leq \|g_k\|$. Set $\Delta x_{kj} = x_{kj} - x_k$ and $\Delta g_{kj} = g_{kj} - g_k$.

4. If $\Delta g_{kj} \neq 0$ and $\|(I - U_{kj}U_{kj}^{+})\Delta g_{kj}\|_2 \geq \alpha \|\Delta g_{kj}\|_2$, set $U_{k(j+1)} = [U_{kj}, \Delta g_{kj}]$, $V_{k(j+1)} = [V_{kj}, \Delta x_{kj}]$, $j = j+1$ and go to step 1, Stage III.

5. Set $U_{k+1} = U_{kj}$, $V_{k+1} = V_{kj}$, $k = k+1$ and return to Stage I, step 1.

Notice that because $I - U_{kj}V_{kj}^{+}$ is a projection operator, if $\bar{p}_{kj}'g_k(\|p_{kj}\|_2 \|g_k\|_2)^{-1} < \beta$ then $\omega'g_k(\|\omega\|_2 \|g_k\|_2)^{-1} < \beta$ for any nonzero vector ω in $R^{\perp}(U_{kj})$. Therefore, if the test of step 2 fails, it is futile to search for other vectors in $R^{\perp}(U_{kj})$ bounded away from being orthogonal to g_k by β. Vectors in $R^{\perp}(U_{kj})$ are chosen because, as will be seen, such a choice results eventually in the satisfaction of the condition of step 4 of remark 4.3.3. The convergence and rate of

convergence results of the next section are given only for algorithm 4.3.2. The modifications to the proofs of these results when the procedure of remark 4.3.3 is employed are straightforward.

4.4 Convergence and Rate of Convergence

Before presenting the convergence and rate of convergence results several preliminary definitions and lemmas are needed.

Definition 4.4.1 Let $f(x) : D \subset R^n \to R^1$ be any function. A level set of f, denoted by $L(c)$ is the set of all $x \in D$ such that $f(x) \leq c$.

Definition 4.4.2 Let $f(x) : D \subset R^n \to R^1$ be any function. The path connected component of the level set $L(c)$ containing $\bar{x} \in D$ and denoted by $L^{\bar{x}}(c)$ is the set of all $x \in L(c)$ such that there exists a continuous function $p(t) : [0,1] \to L(c)$ with $p(0) = \bar{x}$ and $p(1) = x$.

Definition 4.4.3 Let $f(x) : D \subset R^n \to R^1$ be any function differentiable on D and let $\{x_k\}$ be any sequence contained in D. A sequence of non-zero vectors $\{p_k\} \subset R^n$ is gradient related to $\{x_k\}$ if there exists a forcing function $\sigma(\cdot)$ such that $|g(x_k)' p_k| \, \|p_k\|^{-1} \geq \sigma(\|g_k\|)$ for all k.

Definition 4.4.4 Let $f(x) : D \subset R^n \to R^1$ be any function differentiable on D. A critical point of $f(x)$ is any point $\hat{x} \in D$ such that $g(\hat{x}) = 0$.

Definition 4.4.5 Let $f(x) : D \subset R^n \to R^1$ be continuously differentiable on D and assume that for $D_0 \subset D$

$$\alpha = \sup \{ \| g(x) - g(y) \| : x,y \in D_0 \} > 0. \tag{4.4.1}$$

The mapping $\delta(t) : [0,\infty) \to [0,\infty)$, defined by

$$\delta(t) = \begin{cases} \inf \|x - y\| : x,y \in D_0, \ \|g(x) - g(y)\| \geq t, & t \in [0,\alpha), \\ \lim_{s \to \alpha} \delta(s), & t \in [\alpha,\infty), \end{cases} \tag{4.4.2}$$

is the reverse modulus of continuity of $g(x)$ on D_0.

The following two lemmas are necessary to prove that, under appropriate conditions, the sequence $\{x_k\}$ generated by algorithm 4.3.2 is determined.

Lemma 4.4.6 Assume that $f(x) : D \subset R^n \to R^1$ has a continuous derivative on $\underline{D}$, that $D_0 \subset \underline{D}$ is compact and that α of equation 4.4.1 is positive. Then $\delta(t)$, the reverse modulus of continuity of $g(x)$ on D_0, is positive for all $t > 0$ and $\delta(t)$ is a forcing function.

A proof of this result may be found in (ORT 70, lemma 14.2.6).

Lemma 4.4.7 Assume that $f(x) : D \subset R^n \to R^1$ has a continuous derivative on $\underline{D}$, that for $x \in \underline{D}$, $L(f(x))$ is compact and that for $p \in R^n$, $g(x)'p > 0$. There exists a solution for $\bar{\lambda}$ of the equation

$$\bar{\lambda} = \min \{\lambda \geq 0 : p'g(x - \lambda p) = \mu p'g(x)\} \qquad (4.4.3)$$

with $(x - \bar{\lambda}p) \in L^x(f(x))$ and $\mu \in [0,1)$. Further

$$f(x) - f(x - \bar{\lambda}p) \geq \sigma(g'(x)p \, \|p\|_2^{-1}) \qquad (4.4.4)$$

where $\sigma(t) = \mu t\delta([1 - \mu]t)$, $t \geq 0$, and where $\delta(t)$ is the reverse modulus of continuity of $g(x)$ on $L^x(f(x))$.

Proof: A proof of this result is contained in the proof of lemma 14.2.7 of (ORT 70). In order to apply lemma 14.2.7 and its proof, it is necessary to note that any connected component of a compact set is compact.

It will now be shown that if $\sigma_2(\cdot)$ is properly chosen, algorithm 4.3.2 generates a well-defined sequence $\{x_k\}$.

Theorem 4.4.8 Suppose $f(x) : D \subset R^n \to R^1$ is continuously differentiable on $\underline{D}$ and that for $x_0 \in \underline{D}$, $L(f(x_0))$ is compact. Suppose the forcing function $\sigma_2(t)$ of algorithm 4.3.2 is such that

$$\sigma_2(t) \leq \mu t \, \delta([1 - \mu]t) \qquad (4.4.5)$$

for all $t \geq 0$ where $\delta(t)$ is the reverse modulus of continuity of $g(x)$ on $L(f(x_0))$ and μ is a fixed constant in $(0,1)$. If algorithm 4.3.2 is applied to $f(x)$ starting at x_0 then the sequence $\{x_k\}$ is contained in $L(f(x_0))$ and is determined for all $k \geq 0$ unless for some $k \geq 0$, $g_k = 0$.

Proof: Notice first that Stage II of algorithm 4.3.2 can always be completed since a pair $\Delta x_{\bar{k}}$, $\Delta g_{\bar{k}}$ always exists for $k - \ell \leq \bar{k} \leq k$ and since a one column matrix always satisfies condition b of Stage II, step 1. Also, Stage III of algorithm 4.3.2 can always be completed since a return to Stage I is generated if U_{kj} has n columns or if step 2, Stage III fails to generate a column with which to form $U_{k(j+1)}$ and $V_{k(j+1)}$. It remains to be shown that Stage I can always be completed. Since for $k = 0$, $x_k \in L(f(x_0))$ is given and $g_k \neq 0$, unless the theorem is trivially satisfied, assume that at stage k, $x_k \in L(f(x_0))$ and $g_k \neq 0$. In order to specify x_{k+1} it is necessary that $p_k \neq 0$ and λ_k be determined. Either p_k will be set to $\bar{p}_k \neq 0$ or p_k will be set to $p_k^* \neq 0$ or step 5 of Stage I will be executed. If step 5 of Stage I is executed then the choice $p_k = g_k \neq 0$ is always satisfactory so that p_k can always be determined. Now λ_k will either be chosen equal to one in step 6 of Stage I or step 7 of Stage I will be executed. By lemma 4.4.7, since $f(x)$ is continuously differentiable on $\underline{D}$, there always exists a λ_k such that

$$f(x_k) - f(x_k - \lambda_k p_k) \geq \sigma_2(g_k' p_k \| p_k \|_2^{-1}) \qquad (4.4.6)$$

so that if step 7 of Stage I is executed λ_k is determined and therefore x_{k+1} is determined. Further, if either step 6 or step 7 of

Stage I is executed, $x_{k+1} \in L(f(x_0))$. At stage k+1 then either $g_k = 0$, in which case the sequence $\{x_k\} \subset L(f(x_0))$ is determined up to stage k or $g_{k+1} \neq 0$, in which case an inductive cycle is complete. Q.E.D.

Before proceeding with the convergence results one further lemma is required.

Lemma 4.4.9 Suppose $f(x) : D \subset R^n \to R^1$ is continuously differentiable on $\underline{D}$, that $D_0 \subset \underline{D}$ is compact and that $\{x_k\} \subset D_0$ is any sequence such that $\lim_{k \to \infty} g(x_k) = 0$. Then the set Ω of all critical points of $f(x)$ in D_0 is not empty and

$$\lim_{k \to \infty} \{\inf_{x \in \Omega} \|x_k - x\|\} = 0. \tag{4.4.7}$$

Proof: Because D_0 is compact, $\{x_k\}$ has a convergent subsequence $\{\bar{x}_i\}$ and if $\lim_{i \to \infty} \{\bar{x}_i\} = \bar{x}$ then because $g(x)$ is continuous on D_0 $g(\bar{x}) = 0$ so that $\bar{x} \in \Omega$ and Ω is not empty. Next assume that

$$\lim_{k \to \infty} \{\inf_{x \in \Omega} \|x_k - x\|\} \neq 0. \tag{4.4.8}$$

Again, because D_0 is compact there must exist a subsequence $\{\tilde{x}_j\}$ of $\{x_k\}$ such that

$$\lim_{j \to \infty} \{\inf_{x \in \Omega} \|\tilde{x}_j - x\|\} = \delta > 0. \tag{4.4.9}$$

But because for any convergent subsequence $\{x_j\}$ $\lim_{j \to \infty} \tilde{x}_j = \tilde{x}$ and $\tilde{x} \in \Omega$,

$$\lim_{k \to \infty} \{\inf_{x \in \Omega} \|\tilde{x}_j - x\|\} \leq \lim_{j \to \infty} \|\tilde{x}_j - \tilde{x}\| = 0 \tag{4.4.10}$$

which is a contradiction. Q.E.D.

We next demonstrate that under appropriate hypotheses the sequence $\{x_k\}$ generated by algorithm 4.3.2 converges in the sense of equation

4.4.7 to the set of critical points of $f(x)$.

Theorem 4.4.10 Suppose $f(x) : D \subset R^n \to R^1$ is continuously differentiable on $\underline{D}$ and that for $x_0 \in \underline{D}$, $L(f(x_0))\ \underline{D}$ is compact. Then the set $\Omega(x_0)$ of critical points of $f(x)$ in $L(f(x_0))$ is not empty and if algorithm 4.3.2 is determined for all $k \geq 0$ then

$$\lim_{k \to \infty} \{\inf_{x \in \Omega(x_0)} \|x_k - x\| \} = 0. \tag{4.4.11}$$

Proof: First, because $f(x)$ is continuous on $\underline{D}$ and because $L(f(x_0))$ is compact, $f(x)$ is minimized on $L(f(x_0))$ at some $\hat{x} \in L(f(x_0))$. Since $g(x)$ is continuous on $\underline{D}$ it must be that $g(\hat{x}) = 0$ and therefore $\Omega(x_0)$ is not empty. Because of steps 6 and 7 of Stage I, $\{x_k\} \subset L(f(x_0))$. In order to establish the result of the theorem it is necessary to show that $\lim_{k \to \infty} g_k = 0$ and to apply lemma 4.4.9. Notice that the sequence $\{p_k\}$ generated by the algorithm is gradient related to the sequence $\{x_k\}$ since, because of steps 3, 4 and 5 of Stage I,

$$|g_k' p_k| \ \|p_k\|^{-1} \geq g_k' p_k \|p_k\|^{-1} \geq \beta \|g_k\| = \bar{\sigma}(\|g_k\|) \tag{4.4.12}$$

for all k, where $\bar{\sigma}(\cdot) = \beta t$. Consider the subsequence $\{\bar{x}_j\}$ of $\{x_k\}$ consisting of all elements of $\{x_k\}$ generated by step 7 of Stage I of the algorithm. Because $f(x_{k+1}) < f(x_k)$ for all k and because of step 7 it must be that

$$f(\bar{x}_j) - f(\bar{x}_{j+1}) \geq \sigma_2(\bar{p}_j' \bar{g}_j \|\bar{p}_j\|^{-1}). \tag{4.4.13}$$

The function $f(x)$ is continuous and bounded below on $L(f(x_0))$ and therefore

$$\lim_{j \to \infty} f(\bar{x}_j) - f(\bar{x}_{j+1}) = 0. \tag{4.4.14}$$

It follows from the definition of a forcing function that $\lim_{j \to \infty} \bar{p}_j' \bar{g}_j \|\bar{p}_j\|^{-1} = 0$. Since $\{\bar{p}_i\}$ is gradient related to $\{\bar{x}_j\}$,

$\lim_{j \to \infty} \bar{\sigma}(\| \bar{g}_j \|) = 0$. This in turn implies that

$$\lim_{j \to \infty} \| \bar{g}_j \| = 0. \tag{4.4.15}$$

Consider next the subsequence $\{\tilde{x}_j\}$ of $\{x_k\}$ consisting of all elements of $\{x_k\}$ generated by step 6 of Stage I. The subsequence $\{\tilde{x}_i\}$ contains all elements of $\{x_k\}$ not contained in $\{\tilde{x}_j\}$ since for all k either step 6 or step 7 of Stage I is executed. Because $f(x_{k+1}) < f(x_k)$ for all k and because of step 6

$$f(\tilde{x}_i) - f(\tilde{x}_{i+1}) \geq \sigma_1(\| \tilde{g}_i \|). \tag{4.4.16}$$

Again, because $f(x)$ is continuous and bounded below on $L(f(x_0))$

$$\lim_{i \to \infty} f(\tilde{x}_i) - f(\tilde{x}_{i+1}) = 0 \tag{4.4.17}$$

and therefore

$$\lim_{i \to \infty} \| \tilde{g}_i \| = 0. \tag{4.4.18}$$

Since $\{\tilde{x}_i\}$ and $\{\bar{x}_j\}$ contains all elements of $\{x_k\}$

$$\lim_{k \to \infty} \| g_k \| = 0.$$

Application of lemma 4.4.9 completes the proof. Q.E.D.

Theorem 4.4.10 demonstrates that the sequence $\{x_k\}$ generated by algorithm 4.3.2 gets arbitrarily close to the set $\Omega(x_0)$ of critical points of $f(x)$ in $L(f(x_0))$. What has not been shown is that

$$\lim_{k \to \infty} \| x_{k+1} - x_k \| = 0. \tag{4.4.19}$$

It may in fact be possible for the sequence $\{x_k\}$ to "hop around" in the neighborhood of different elements of $\Omega(x_0)$. It is evident that if $\Omega(x_0)$ contains only one element $\hat{x}$ then

$$\lim_{k \to \infty} x_k = \hat{x} \tag{4.4.20}$$

and

$$\lim_{k \to \infty} \| x_{k+1} - x_k \| = 0. \tag{4.4.21}$$

If $\Omega(x_0)$ contains only a finite number of elements, then the following result holds.

Theorem 4.4.11 Suppose $f(x) : D \subset R^n \to R^1$ is continuously differentiable on D and that for $x_0 \in D$, $L(f(x_0)) \subset D$ is compact. Suppose the set $\Omega(x_0)$ of critical points of $f(x)$ in $L(f(x_0))$ has a finite number of elements. If algorithm 4.3.2 is applied to $f(x)$ starting at x_0, if the sequence $\{x_k\}$ is determined for all $k \geq 0$ and if $\{\tilde{x}_i\}$ and $\{\bar{x}_j\}$ are any two convergent subsequences of the sequence $\{x_k\}$ with limit points $\tilde{x}$ and $\bar{x}$ respectively, then

$$f(\bar{x}) = f(\tilde{x}).$$

Proof: By theorem 4.4.10

$$\lim_{k \to \infty} \{\inf_{x \in \Omega(x_0)} \| x_k - x \| \} = 0. \tag{4.4.22}$$

Assume $f(\tilde{x}) < f(\bar{x})$. Because the subsequence $\{\bar{x}_j\}$ is convergent, given $\delta > 0$ there exists a j_1 such that for all $j > j_1$, $\| \bar{x}_j - \bar{x} \| \leq \delta$. Because f is continuous on D, given $\varepsilon > 0$ there exists a $\delta > 0$ such that $\| f(\bar{x}_j) - f(\bar{x}) \| \leq \varepsilon$ whenever $\| \bar{x}_j - \bar{x} \| < \delta$. Therefore, given $\varepsilon > 0$ there exists a j_1 such that for all $j > j_1$, $\| f(\bar{x}_j) - f(\bar{x}) \| < \varepsilon$. Because the sequence $\{f(\tilde{x}_i)\}$ is non-increasing $f(\tilde{x})_i > f(\tilde{x})$ for all i. If ε is chosen less than $f(\tilde{x}) - f(\bar{x})$, then for all $j > j_1$ and for all i, $f(\tilde{x}_i) > f(\bar{x}_j)$. Because both sequences $\{\tilde{x}_i\}$ and $\{\bar{x}_j\}$ are not finite it must be that the sequence

$\{f(x_k)\}$ is not non-increasing which is a contradiction. Q.E.D.

Finally, it will be demonstrated that under appropriate conditions the sequence $\{x_k\}$ of algorithm 4.3.2 for large enough k is generated by the recursion

$$x_{k+1} = x_k - V_k U_k^{-1} g_k \tag{4.4.23}$$

where the matrices V_k and U_k have n columns and are elements of $S(\hat{x},\delta,\alpha)$ for arbitrarily small δ and that the rate of convergence of the sequence $\{x_k\}$ is superlinear.

Theorem 4.4.12 Suppose $f(x) : D \subset R^n \to R^1$ is continuously differentiable on $\underline{D}$ and that for $x_0 \in \underline{D}$, $L(f(x_0)) \subset \underline{D}$ is compact. Suppose algorithm 4.3.2 is applied to $f(x)$ starting at x_0 and that

$$\lim_{k \to \infty} x_k = \hat{x} \tag{4.4.24}$$

where $\hat{x} \in \Omega(x_0)$, the set of critical points of f in $L(f(x_0))$. Suppose $f(x)$ is twice differentiable at $\hat{x}$ and that $H(\hat{x})$ is positive definite and strong. If the constants α and β of algorithm 4.3.2 are chosen sufficiently small, if the forcing function $\sigma_1(\cdot)$ is such that

$$\sigma_1(t) \leq C \min_{\omega \in R^n} \frac{\omega' H(\hat{x}) \omega}{\|\omega\|^2} \|H(\hat{x})\|^{-2} t^2 \tag{4.4.25}$$

for all t sufficiently small and for some $C \in (0,1)$ and if the set $K_3 = I_p - K_1 \cup K_2$ contains n elements each sufficiently large, then there exists a k^* such that for all $k > k^*$

$$x_{K+1} = x_k - P_k^*. \tag{4.4.26}$$

Further

$$\lim_{k \to \infty} \frac{\|x_{k+1} - \hat{x}\|}{\|x_k - \hat{x}\|} = 0. \tag{4.4.27}$$

Proof: Since $\lim_{k \to \infty} x_k = \hat{x}$ for any $\delta_1 > 0$ $\exists$ a k_1 $\ni$ $\forall$ $k \geq k_1$ $x_k \in S(\hat{x},\delta_1)$. Since $\lim_{k \to \infty} g_k = 0$, for any $\delta_2 > 0$ $\exists$ a k_2 $\ni$ $\forall$ $k \geq k_2$ and $\forall$ $j \geq 0$ such that x_{kj} is defined, $x_{kj} \in S(x_k,\delta_2)$.

Choosing δ_1 and δ_2 such that $1/2 \min \{\delta_1,\delta_2\} = \delta_3$ and $k_3 = \max \{k_1,k_2\}$ for any $\delta_3 > 0$ $\exists$ a k_3 $\ni$ $\forall$ $k \geq k_3$ and $\forall$ $j \geq 0$, $x_{kj},x_k \in S(\hat{x},\delta_3)$. Because of the age bound ℓ in step 1 of Stage II, for any $\delta_3 > 0$ $\exists$ a k_3 $\ni$ $\forall$ $k \geq k_3 + \ell$ and $\forall$ $j \geq 0$, (U_k,V_k) and (U_{kj},V_{kj}) are in $S(\hat{x},\delta_3,\alpha)$ where α is the constant of the algorithm.

By lemma 4.2.1 and lemma 4.1.11 choosing δ_3 above sufficiently small we have that for all $k > k_3 + \ell$ and α sufficiently small $\Delta g_{kj} \neq 0$,

$$\| (I - U_{kj} U_{kj}^{+}) \Delta g_{kj} \|_2 \geq \alpha \| \Delta g_{kj} \|_2 \qquad (4.4.28)$$

in Stage III, step 3. It must be then that with α chosen sufficiently small $\forall$ $k > k_3 + \ell$ step 4 of Stage III is never executed. Therefore, for $k > k_3 + \ell$ the only returns to Stage I are generated in Stage II, step 2 or in Stage III, step 1. Since K_1 is finite it has a maximal element k_4 and for $k > k_5 = \max \{k_3 + \ell, k_4\}$ the only returns to Stage I are generated in Stage III, step 1. Because U_k and V_k can only be left undefined by a return to Stage I from Stage II, step 2, for all $k > k_5$, U_k and V_k are defined.

Next, by lemma 4.2.9, remark 4.2.10 and lemma 4.1.11 choosing δ_3 above sufficiently small we have that in Stage I for $\beta > 0$ sufficiently small

$$g_k' \bar{p}_k (\| g_k \|_2 \| \bar{p}_k \|_2)^{-1} \geq \beta \qquad (4.4.29)$$

and

$$g_k' p_k^* (\| g_k \|_2 \| p_k^* \|_2)^{-1} \geq \beta \qquad (4.4.30)$$

cannot occur together for $k > k_5$ in Stage I. Therefore, for $k > k_5$ $p_k = \bar{p}_k$ or $p_k = p_k^*$.

For $k > k_5$ Stage III, step 1 generates a return to Stage I with U_{k+1} having n columns or with U_{k+1} having more columns than U_k or with U_{k+1} having as many columns as U_k but with $k \in K_2$. For $k > k_5$ the number of columns of U_k and V_k cannot decrease and further since K_3 was assumed to have at least n elements, each sufficiently large, there exists a $k_6 > k_5$ such that for all $k > k_6$, U_k and V_k have exactly n columns. By the definition of $S(\hat{x},\delta,\alpha)$, U_k has full rank for all k and therefore for $k > k_6$ U_k is nonsingular, $(I - U_k U_k^+) = 0$, $\bar{p}_k = 0$ and $p_k = p_k^*$.

Because f is twice differentiable at $\hat{x}$ and because $\lim_{k \to \infty} x_k = \hat{x}$ for every $\varepsilon > 0$ $\exists$ a k_7 $\ni$ $\forall$ $k > k_7$

$$f(x_k) = f(\hat{x}) + \frac{1}{2}(x_k - \hat{x})'H(\hat{x})(x_k - \hat{x}) + r_k, \qquad (4.4.31)$$

$$\|r_k\| \leq \varepsilon \|x_k - \hat{x}\|^2.$$

A verification of this result appears in (ORT 70, lemma 3.3.12).

Further, because $H(\hat{x})$ is strong and because $\lim_{k \to \infty} x_k = \hat{x}$ for every $\varepsilon > 0$ $\exists$ a k_8 $\ni$ $\forall$ $k > k_8$

$$g(x_k) = H(\hat{x})(x_k - \hat{x}) + e_k, \quad \|e_k\| \leq \varepsilon \|x_k - \hat{x}\|. \qquad (4.4.32)$$

By remark 4.2.8 and because $\bar{p}_k = 0$ for $k > k_6$ it follows that for every $\varepsilon > 0$ $\exists$ a k_9 $\ni$ $\forall$ $k > k_9$

$$h_{k\ 2} = \|p_k^* - H^{-1}(\hat{x})g_k\|_2 \leq C_1 \varepsilon \|H^{-1}(\hat{x})\|_2 \|g_k\|_2. \qquad (4.4.33)$$

By lemma 4.1.11 $\exists$ a k_{10} $\ni$ $\forall$ $k > k_{10}$

$$\|g_k\| \leq C_2 \|x_k - \hat{x}\|. \qquad (4.4.34)$$

Next, we have that

$$x_k - p_k^* = x_k - H^{-1}(\hat{x})g_k - h_k \tag{4.4.35}$$

$$= x_k - H^{-1}(\hat{x})(H(\hat{x})(x_k - \hat{x}) + e_k) - h_k$$

$$= \hat{x} - H^{-1}(\hat{x})e_k - h_k.$$

Therefore, because f is twice differentiable at $\hat{x}$ $\exists$ a $k_{11} > k_6$ $\ni$ $\forall$ $k > k_{11}$

$$f(x_k - p_k^*) = f(\hat{x}) + (H^{-1}(\hat{x})\, e_k + h_k)'H(\hat{x}) \tag{4.4.36}$$

$$(H^{-1}(\hat{x})\, e_k + h_k) + r(x_k - p_k^*)$$

where $\|r(x_k - p_k^*)\| \leq \varepsilon \|H^{-1}(\hat{x})\, e_k - h_k\|^2$. It follows that for every $\varepsilon > 0$ $\exists$ a $k_{12} = \max\{k_7, k_8, k_9, k_{10}, k_{11}\}$ $\ni$ $\forall$ $k > k_{12}$

$$f(x_k) - f(x_k - p_k^*) = (x_k - \hat{x})'H(\hat{x})(x_k - \hat{x}) \tag{4.4.37}$$

$$- (H^{-1}(\hat{x})\, e_k + h_k)H(\hat{x})(H^{-1}(\hat{x})\, e_k + h_k)$$

$$+ r_k - r(x_k - p_k^*)$$

where

$$\|r_k\| \leq \varepsilon \|x_k - \hat{x}\|^2 \tag{4.4.38}$$

$$\|e_k\| \leq \varepsilon \|x_k - \hat{x}\| \tag{4.4.39}$$

$$\|h_k\| \leq \varepsilon\, C_1 C_2 \|H^{-1}(\hat{x})\|_2 \|x_k - \hat{x}\| \tag{4.4.40}$$

$$\|r(x_k - p_k^*)\| \leq \varepsilon\, C_3 \|x_k - \hat{x}\|^2. \tag{4.4.41}$$

Because $H(\hat{x})$ is strong, for any $\varepsilon > 0$ $\exists$ a $\delta > 0$ $\ni$ $\forall$ $x \in S(\hat{x}, \delta)$

$$\|g(x) - g(\hat{x}) - H(\hat{x})(x - \hat{x})\| \leq \varepsilon \|x - \hat{x}\| \tag{4.4.42}$$

or

$$\|g(x)\| \leq \|H(\hat{x})(x - \hat{x})\| + \varepsilon \|x - \hat{x}\| \qquad (4.4.43)$$

$$\leq (\|H(\hat{x})\| + \varepsilon) \|x - \hat{x}\| .$$

It follows that for every $\varepsilon > 0$ $\exists$ a k_{13} $\ni$ $\forall$ $k > k_{13}$

$$(x_k - \hat{x})' H(\hat{x})(x_k - \hat{x}) = \frac{(x_k - \hat{x})' H(\hat{x})(x_k - \hat{x})}{\|x_k - \hat{x}\|^2} \|x_k - \hat{x}\|^2 \qquad (4.4.44)$$

$$\geq \min_{\omega \in R^n} \frac{\omega' H(\hat{x})\omega}{\|\omega\|^2} (\|H(\hat{x})\| + \varepsilon)^{-2} \|g_k\|^2 .$$

Finally, if ε is taken sufficiently small then there exists a $k^* = \max \{k_{12}, k_{13}\}$ such that for all $k > k^*$

$$f(x_k) - f(x_k - p_k^*) \geq C \min_{\omega \in R^n} \frac{\omega' H(\hat{x})\omega}{\|\omega\|^2} \|H(\hat{x})\|^{-2} \|g_k\|^2 \qquad (4.4.45)$$

$$\geq \sigma_1 (\|g_k\|)$$

for $C \in (0,1)$. Therefore, for all $k > k^*$

$$p_k = p_k^* \qquad (4.4.46)$$

and

$$x_{k+1} = x_k - p_k . \qquad (4.4.47)$$

The final result of the theorem follows immediately since for every $\varepsilon > 0$ $\exists$ a k^* $\ni$ $\forall$ $k > k^*$

$$\|x_{k+1} - \hat{x}\| = \|x_k - p_k^* - \hat{x}\| \qquad (4.4.48)$$

$$= \|H^{-1}(\hat{x}) e_k + h_k\|$$

$$\leq \|H^{-1}(\hat{x})\| \|e_k\| + \|h_k\|$$

$$\leq \varepsilon C_4 \|x_k - \hat{x}\| . \qquad \text{Q.E.D.}$$

Remark 4.4.13 The coefficients α and β in theorem 4.4.13 are restricted to be less than some positive constants $\bar{\alpha}$ and $\bar{\beta}$, respectively. Specifically α must be chosen so that inequality 4.4.28 is eventually satisfied and β must be chosen so that the inequalities 4.4.29 and 4.4.30 eventually are not satisfied together. These conditions arise from lemmas 4.2.1, 4.2.7, and 4.1.11 and remark 4.2.8. Referring to these lemmas we see that α must be chosen less than C_2C_3 of lemma 4.2.1 and that β must be chosen less than $C_3(1 + C_2)^{-1}$ of lemma 4.2.7. Upon checking the origin of these constants it can be established that an adequate choice for $\bar{\alpha}$ is

$$\bar{\alpha} = \min_{v \in R^n} \frac{v'H(\hat{x})v}{\|v\|_2 \|H(\hat{x})v\|_2} \qquad (4.4.49)$$

and that an adequate choice for $\bar{\beta}$ is

$$\bar{\beta} = \min_{u \in R^n} \frac{u'H^{-1}(\hat{x})u}{\|u\|_2 \|H^{-1}(\hat{x})u\|} \left(1 + \max_{u \in R^n} \frac{\|H^{-1}(\hat{x})\|_2 \|u\|_2}{\|H^{-1}(\hat{x})u\|_2}\right)^{-1}. \qquad (4.4.50)$$

4.5 Conclusion

In theorems 4.4.8 and 4.4.12 it was necessary to restrict the admissible forcing functions $\sigma_1(t)$ and $\sigma_2(t)$ and the admissible constants α and β in a way which depends upon the objective function $f(x)$. Specifically in theorem 4.4.8, to insure that the sequence $\{x_k\}$ is determined, $\sigma_2(t)$ is bounded by a forcing function which depends on the reverse modulus of continuity of $g(x)$ on $L(f(x_0))$. In theorem 4.4.12, to insure that the terminal rate of convergence to $\hat{x}$ is superlinear, the constants α and β are chosen less than constants related to $H(\hat{x})$ and for small t, $\sigma_1(t)$ is chosen less than a forcing function related to $H(\hat{x})$.

In practice it is unlikely that specific information will be available about the reverse modulus of continuity of $g(x)$ on $L(f(x_0))$ or about $H(\hat{x})$. However, the choice of the forcing function $\sigma_2(t)$ does not cause difficulties since, as is seen in (ORT 70, section 14.2), step-length algorithms have been developed which insure that condition 4.4.5 is satisfied provided only that $f(x)$ is continuously differentiable on $L(f(x_0))$.

The constants α and β depend upon the conditioning of the matrix $H(\hat{x})$ and can be initially chosen small enough to cause superlinear convergence on functions with reasonably well conditioned minima. If, during the execution of the algorithm, it is determined that the α and β conditions are not being satisfied because the Hessian is not well conditioned at the minimum these constants can be reduced. A similar procedure can be used with the forcing function $\sigma_1(t)$. Initially $\sigma_1(t)$ can be chosen as γt_2 for some positive constant γ which is small enough to insure superlinear convergence provided the Hessian at the solution is reasonably well conditioned. The constant γ can then be revised if necessary.

Suppose in algorithm 4.3.2, Stage III is eliminated and a return to Stage I is always generated from Stage II. If it happens that for all k greater than some k^* matrices U_{k+1} and V_{k+1} can be chosen in Stage II having n columns and satisfying the age bound of Stage II, step 1a and the angle condition 4.3.1 then the results of theorems 4.4.8 through 4.4.12 will still hold. Further, if there exists a subsequence of the iteration sequence where matrices U_{k+1} and V_{k+1} can be chosen with n columns that satisfy the age bound and the angle condition then convergence on this subsequence will be superlinear. A

review of the proofs of theorems 4.4.8 through 4.4.12 should convince the reader that this is the case. The specific algorithms discussed in Chapter 5 do not implement Stage III of algorithm 4.3.2 as it was felt that for k sufficiently large it will nearly always be possible to choose matrices U_{k+1} and V_{k+1} with n columns while satisfying the conditions of Stage II.

In developing algorithm 4.3.2 the principal reasons for choosing the data sets $S_k = (U_k, V_k)$ from the family $S(\hat{x},\delta,\alpha)$ were:

1. To insure the existence of a matrix which emulates S_k over T for all k.
2. To insure that the norm of U_k^+ remains appropriately bounded so that for small δ, $\| H^{-1}(x)U_kU_k^+ - V_kU_k^+ \|$ becomes small and so that $p_k^* \equiv V_kU_k^+g_k$ nearly equals $H^{-1}(\hat{x})g_k$ when g_k is nearly in $R(U_k)$.

These properties of $S(\hat{x},\delta,\alpha)$ are fundamental and should be of value in developing other minimization procedures. It may be possible to obtain strong rate of convergence results even if the data sets are not restricted to some family $S(\hat{x},\delta,\alpha)$. For instance if it happens that Δg_k is nearly in $R(U_k)$, one might ignore the component of Δg_k not in $R(U_k)$ and consider the data set $(U_{k+1},V_{k+1}) = ([U_k,\tilde{\Delta g}_k],[V_k,\Delta x_k])$ where $\tilde{\Delta g}_k$ is the component of Δg_k contained in $R(U_k)$. If this is done it may happen that no emulation exists for (U_{k+1},V_{k+1}). However, the norm of U_{k+1}^+ should remain appropriately bounded and for small δ, $p_k^* \equiv V_kU_k^+g_k$ should approximate $H^{-1}(\hat{x})g_k$.

It is conjectured that if $H(x)$ is required to be Lipshitz continuous in a neighborhood of $\hat{x}$ then it can be proved for algorithm

4.3.2 that

$$\underset{k \to \infty}{\text{Limit}} \frac{\|x_{k+\ell} - \hat{x}\|}{\|x_k - \hat{x}\|^2} < C < \infty$$

where ℓ is the age bound of the algorithm. The key to proving this result lies in obtaining a stronger bound on $\|h_k\|_2$ in equation 4.4.33. To do this requires that lemma 4.2.5 be modified so as to obtain the appropriate bound in lemma 4.2.6.

CHAPTER 5

NUMERICAL EXPERIMENTS

5.1 Introduction

Minimization algorithm 4.3.2 is constructed so that convergence is insured and so that the terminal rate of convergence is superlinear for a large class of objective functions. At the same time there is considerable latitude in the choice of a specific algorithm within the general framework of algorithm 4.3.2. This latitude is intended so as to allow an algorithm to be developed which performs well away from the minimum and which retains the desirable convergence and terminal rate of convergence properties. Because it is very difficult to obtain theoretical results about performance of an algorithm away from the minimum some experimental work is usually necessary to evaluate the overall performance of a specific algorithm. To verify that it is feasible to construct an effective algorithm within the framework of algorithm 4.3.2, two algorithms were programmed in FORTRAN. These algorithms were applied to a variety of test functions and the results were compared to the results obtained using the version of Davidon's algorithm available in the I.B.M. 360 Scientific Subroutine Package. The results of these numerical experiments are presented in this chapter. We begin by describing the version of Davidon's algorithm which was used as the standard of comparison.

5.2 Davidon's Algorithm

The Scientific Subroutine Package version of Davidon's algorithm was chosen because it is widely circulated and accepted as an efficient program for minimizing functions. Certain drawbacks in the algorithm, as a standard of comparison, have been accepted in order to obtain a

program which is readily available to other investigators. The algorithm uses the Davidon recursion formula to generate a sequence of matrices $\{H_k\}$ beginning with the matrix $H_o = I$. It differs from the algorithm proposed by Fletcher and Powell (FLE 64) in two important respects. First, when certain rather pathological conditions occur the matrix H_k is reset to the identity matrix. Second, the linear search subalgorithm does not perform an exact linear minimization. Except for these variations the Scientific Subroutine Package Davidon is the Fletcher-Powell-Davidon algorithm which is specified by algorithm 3.4.1 and by the Davidon recursion formula 3.4.1 with $H_o = I$. So that there is no confusion as to how the program operates we will specify the conditions under which a reset can occur and the linear search subalgorithm.

The following conditions at stage k cause H_{k+1} to be taken as the identity.

1. The inner product of the linear search direction $- H_k g_k$ with the gradient g_k is positive or zero.
2. The ratio $\|H_k g_k\| \, \|g_k\|^{-1}$ is not greater than ε where ε is a user supplied constant.
3. The function value does not decrease by more than ε in one iteration where ε is the same user supplied constant as in step 2.
4. The argument of a square root operation involved in the cubic interpolation of the linear search is negative. This occurs if the cubic interpolation polynomial (see 5.2.1 below) has no minimum on the interpolation interval. Because of the procedure for choosing the interpolation interval this should not happen.
5. The product $(u_k' v_k)(u_k' H_k u_k) = 0$. If this occurs then the Davidon recursion formula involves division by zero.

In practice the reset was encountered very infrequently. If a reset occurred during a Davidon run the iteration at which it occurred will be noted in the data.

The linear search subalgorithm of the Davidon algorithm performs an approximate linear minimization. Assuming that at stage k an initial point x_k and a search direction p_k are given, the linear search algorithm is as follows.

Algorithm 5.2.1

1. If $0 < 2\,(f(x_k) - f^*)(g_k' H_k g_k)^{-1} < 1$ where f^* is a user supplied estimate of the minimum function value, choose as an initial step

$$x_k + \lambda p_k = x_k + 2(f_k - f^*)(g_k' H_k g_k)^{-1} p_k.$$

Otherwise choose $\lambda = 1$.

2. Compute $f(x_k + \lambda p_k)$ and $g(x_k + \lambda p_k)$. If $p_k' g(x_k + \lambda p_k) > 0$ or if $f(x_k + \lambda p_k) > f(x_k)$, set $x = x_k$, $y = x_k + \lambda p_k$ and continue at step 3.

If $p_k' g(x_k + \lambda p_k) = 0$ set $x_{k+1} = x_k + \lambda p_k$ and exit from the linear search.

If $p_k' g(x_k + \lambda p_k) < 0$ set $\lambda = 2\lambda$ and repeat step 2.

3. Interpolate cubically on the line segment (x,y) using the directional derivatives in the direction p_k $d(x) \equiv p_k' g(x) \left\| p_k \right\|^{-1}$ and $d(y) \equiv p_k' g(y) \left\| p_k \right\|^{-1}$ and compute the $\bar{x}$ which minimizes the interpolation polynomial on (x,y). If $f(\bar{x}) \leq f(x)$ and $f(\bar{x}) \leq f(y)$, set $x_{k+1} = \bar{x}$ and exit from the linear search.

4. If $d(\bar{x}) \geq 0$ or if both $d(\bar{x}) < 0$ and $f(\bar{x}) > f(x)$, set $y = \bar{x}$ and repeat step 3 unless $d(y) = d(\bar{x})$ and $f(y) = f(\bar{x})$. In this last case set $x_{k+1} = \bar{x}$ and exit from the linear search.

5. If $d(\bar{x}) < 0$ and $f(\bar{x}) \leq f(x)$, set $x = \bar{x}$ and repeat step 3

unless $d(x) = d(\bar{x})$ and $f(x) = f(\bar{x})$ In this last case set $x_{k+1} = \bar{x}$ and exit from the linear search.

This particular linear search algorithm has certain disadvantages. First, in step 1 of the algorithm the initial step size is determined from a user supplied estimate of the minimum function value. A good estimate of the minimum function value may not be available. Second, step 3 of the algorithm determines the minimum of a cubic polynomial which agrees with f and its directional derivative at the points x and y. A quadratic interpolation using three function evaluations or the directional derivative at x_k and two function evaluations would greatly reduce the total number of gradient evaluations and still provide a good estimate of the minimum. Third, in step 3 a return from the linear search is generated whenever the function value at the interpolation point is no greater than the function value at either end point. This linear search procedure, therefore, cannot guarantee that the decrease in function value at each stage is bounded greater than an appropriate forcing function. In fact, the deletion procedure in steps 4 and 5 is such that it is possible for the function value to increase as Figure 5.1 demonstrates.

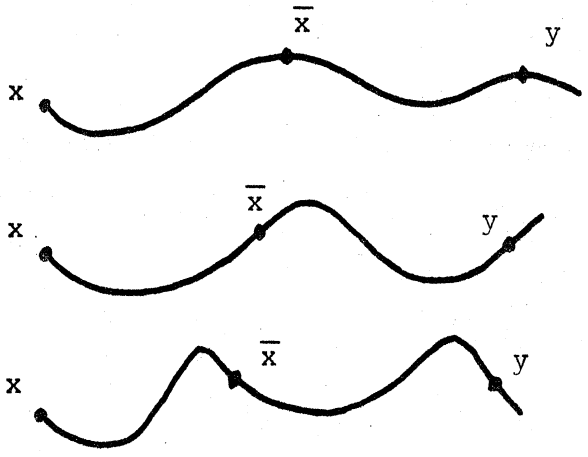


FIGURE 5.1
ABNORMAL CONDITIONS IN LINEAR SEARCH SUBALGORITHM

The curves of Figure 5.1 represent the function along the direction of the linear search. In all three cases shown the point x_{k+1} will be chosen to be $\bar{x}$ since $f(y) = f(\bar{x})$ and $d(y) = d(\bar{x})$. Although these conditions are exceptional, they can occur. Of these disadvantages the one of most concern is that the linear search subalgorithm does not guarantee a "sufficient decrease" in the function value at each stage.

A complete listing of the Scientific Subroutine Package Davidon program is given in Appendix II. Some statements have been added to display pertinent data at each stage of the algorithm. These, however, do not alter the operation of the algorithm in any way. Several tests for abnormal conditions in the algorithm cause program execution to terminate. None of these conditions were encountered in practice.

5.3 The New Algorithms

The new minimization algorithms which were programmed differ in three respects from the structure proposed in general algorithm 4.3.2. First, the linear search procedure 5.2.1 was used in both new algorithms without modification. The primary objective of the numerical study was to compare the major iteration algorithms rather than the linear search subalgorithm. For such a comparison to be valid the linear search procedure should be the same in all algorithms. Since the Davidon algorithm was chosen as the standard its linear search procedure was used. As was noted earlier the bounds imposed by forcing functions $\phi_1(t)$ and $\phi_2(t)$ in algorithm 4.3.2 will not necessarily be satisfied when linear search algorithm 5.2.1 is employed. Second, algorithm 5.2.1 does not choose a unit step size ($\lambda_k = 1$) when this step size in the direction p_k^* of algorithm 4.3.2 yields a sufficient function decrease. In practice these deviations from the structure of algorithm 4.3.2 did not cause

difficulty. Algorithm 5.2.1 performed adequately in all cases tested except when the Hessian matrix was very badly conditioned. Further, it was noted that for large k the step size approached unity when the search direction was p_k^*.

The third respect in which the new algorithms differ from algorithm 4.3.2 involves stage III of algorithm 4.3.2. As was noted in section 4.5 it may happen that for all k greater than some k* a pair of matrices (U_k, V_k) having n columns can be formed in stage II of algorithm 4.3.2 which satisfy both the age bound and the angle condition of stage II. If this occurs then stage III of the algorithm is unnecessary. It was conjectured that this would be the usual rather than the exceptional situation. Consequently, stage III of algorithm 4.3.2 was not implemented in either of the new algorithms. The experimental results verified that this was indeed the case. It was further conjectured that the age bound on the columns of U_k and V_k could be implicitly satisfied by a proper choice of the constants α and β. Therefore, initially no explict age bound was placed on the columns of U_k and V_k. This procedure will be seen to work well except in the case where the Hessian matrix is very badly conditioned.

Except for the deviations just discussed the two new algorithms are contained within the framework of general algorithm 4.3.2. Assuming an initial point $x_o \in R^n$ is given and that k is initialized to zero, the algorithms can be stated as follows.

Algorithm 5.3.1

1. If $g_k = g(x_k) = 0$ stop.
2. If U_k and V_k are not defined, set $p_k = g_k$ and go to step 6.

3. If $\bar{p}_k \equiv (I - U_k U_k^+) g_k \neq 0$ and $\bar{p}_k' g_k (\|\bar{p}_k\|_2 \|g_k\|_2)^{-1} \geq \beta$, set $p_k = \bar{p}_k$ and go to step 6.

4. If $p_k^* \equiv V_k U_k^+ g_k \neq 0$ and $p_k^{*\prime} g_k (\|p_k^*\|_2 \|g_k\|_2)^{-1} \geq \beta$ set $p_k = p_k^*$ and go to step 6.

5. If U_k and V_k have one column, make U_k and V_k undefined and return to step 2. Otherwise delete the first (oldest) column of U_k and V_k and return to step 2.

6. Set $x_{k+1} = x_k - \lambda_k p_k$ where λ_k is chosen by the steplength algorithm 5.2.1.

7. Set $v_k = x_{k+1} - x_k$, $u_k = j_{k+1} - j_k$ and $i = 1$.
If $\|(I - U_k U_k^+) u_k\|_2 \geq \alpha \|u_k\|_2$, set $U_{k+1} = [U_k, u_k]$, $V_{k+1} = [V_k, v_k]$, $k = k+1$ and return to step 1.

8. If $\|(I - U_k^i (U_k^i)^+) u_k\|_2 < \alpha \|u_k\|_2$, where U_k^i and V_k^i are the matrices U_k and V_k with the i th column deleted, set $U_{k+1} = [U_k^i, u_k]$, $V_{k+1} = [V_k^i, v_k]$, $k = k+1$ and return to step 1.

9. If $i < m$, where m is the number of columns in U_k, set $i = i+1$ and return to step 8.

10. If $i = m$, set $U_{k+1} = U_k$, $V_{k+1} = V_k$, $k = k+1$ and return to step 1.

Algorithm 5.3.2 Algorithm 5.3.2. is identical to algorithm 5.3.1 except that in step 5 p_k is always chosen equal to g_k and the algorithm continues at step 6.

In steps 7 through 10 the vectors u_k and v_k are adjoined to the matrices U_k and V_k to form U_{k+1} and V_{k+1} provided u_k is bounded away from $R(U_k)$. Failing this u_k and v_k are added to U_k^i and V_k^i to form U_{k+1} and V_{k+1} if deleting the i th column of U_k and V_k causes u_k to be bounded away from $R(U_k^i)$. An attempt is made

to delete the oldest data first. If this alsc fails, U_{k+1} and V_{k+1} are taken as U_k and V_k.

Algorithm 5.3.1 and 5.3.2 contain two unspecified constants α and β. Remark 4.4.13 indicates that the choice of these constants depends on the conditioning of the Hessian matrix at the minimum. In particular, to insure that ultimately p_k either equals $\bar{p}_k$ or p_k^*, β should be chosen less than

$$\min_{u \in R^n} \frac{u'H^{-1}(\hat{x})u}{\|u\|_2 \|H^{-1}(\hat{x})u\|_2} \left(1 + \max_{u \in R^n} \frac{\|H^{-1}(x)\|_2 \|u\|_2}{\|H^{-1}(\hat{x})\omega\|_2}\right)^{-1} \tag{5.3.1}$$

and to insure that ultimately when $p_k = \bar{p}_k$ the condition of step 7 will be satisfied, α should be chosen less than

$$\min_{v \in R^n} \frac{v'H(\hat{x})v}{\|v\|_2 \|H(\hat{x})v\|_2} . \tag{5.3.2}$$

Since these conditions cannot be checked beforehand, the reasonable thing to do is to choose α and β small enough so that conditions 5.3.1 and 5.3.2 are satisfied unless the objective function is very poorly conditioned at the minimum. If the constants α and β are chosen too small, numerical difficulties may arise either because the search direction p_k is nearly othogonal to the gradient or because the norm of U_k^+ becomes large. On some functions it may be necessary to adjust the constants α and β during the runs. In the numerical experiments particular attention was paid to the effect of α and β on convergence.

An intuitive justification can be given for the choice of search directions in steps 3, 4, and 5. In step 3, if $\bar{p}_k$, the projection of g_k onto $R^{\perp}(U_k)$, is bounded away from being orthogonal to the gradient then $\bar{p}_k$ is chosen as a search direction. On a quadratic surface with

Hessian H the direction $\bar{p}_k$ is H conjugate to the directions comprising the matrix V_k since

$$\bar{p}_k^t U_k = \bar{p}_k^t H V_k = 0. \tag{5.3.3}$$

Step 4 is entered if g_k is essentially in $R(U_k)$; i.e., $U_k U_k^+ g_k \tilde{=} g_k$. In this case $p_k^* = V_k U_k^+ g_k$ should be a good "Newton-like" step, for if the function is quadratic

$$\begin{aligned} x_{k+1} &= x_k - V_k U_k^+ g_k \qquad (5.3.4) \\ &= x_k - H^{-1} U_k U_k^+ g_k \\ &= x_k - H^{-1} g_k. \end{aligned}$$

If both steps 3 and 4 fail, step 5 chooses some other downhill direction as in this case the function is not "locally quadratic". In algorithm 5.3.1 the oldest data, the first columns of U_k and V_k, are deleted and steps 3 and 4 are repeated in the expectation that without the oldest data either a $\bar{p}_k$ or a p_k^* step will satisfy the direction requirement. At most, step 5 of algorithm 5.3.1 reduces the matrices U_k and V_k until a gradient step is taken. In algorithm 5.3.2, step 5 simply forces a gradient step.

Computation of the matrix U_{k+1}^+ from U_k^+ and u_k is performed by straightforward application of theorems 2.4.1 and 2.4.2. Notice that the columns of U_k are linearly independent for all k. Therefore, the pertinent formulas in theorems 2.4.1 and 2.4.2 are always the formulas for linearly independent vectors. Appendix III contains a listing of the program for algorithm 5.3.1. A simple modification to this program gives algorithm 5.3.2. The modification is not shown.

5.4 Rosenbrock's Banana Valley

The algorithms were tested first on the two dimensional Rosenbrock function which is given by

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_2)^2 \qquad (5.4.1)$$

where x_1 and x_2 are the components of x. The starting points used were $(-1,-1)$ and $(1,-1)$. Figure 5.2 is a plot of $\text{Log}_{10}|f(x)|$ versus iteration number for Davidon's algorithm. The single reset which occurred on the third iteration of the run starting at $(1,-1)$ did not significantly affect the performance of the algorithm. Starting at $(-1,-1)$ Davidon reduced the function value to about 10^{-20} in 16 iterations and 65 function (gradient) evaluations. Note that because of the linear search procedure a gradient evaluation is performed with each function evaluation. From $(1,-1)$ the Davidon algorithm reduced the function value to about 10^{-20} in 17 iterations and 50 function evaluations.

Algorithm 5.3.2 was run on Rosenbrock's function from the same starting points with $\alpha = 10^{-4}$ and $\beta = 10^{-4}$. Figure 5.3 is a plot of $\text{Log}_{10}|f(x)|$ versus iteration number for these runs. From both starting points the age of the oldest column of U_k and V_k reached two on the second iteration and remained at two for the entire run. By the age of the oldest column we mean the current iteration number minus the iteration number at which the first column was added to U_k or V_k. This indicates that at each iteration after the second, algorithm 5.3.2 was able to form U_{k+1} by deleting the oldest column of U_k and appending the vector u_k while still satisfying the conditions of step 8 of algorithm 5.3.2. The procedure for generating the matrices U_k and V_k in algorithm 5.3.2

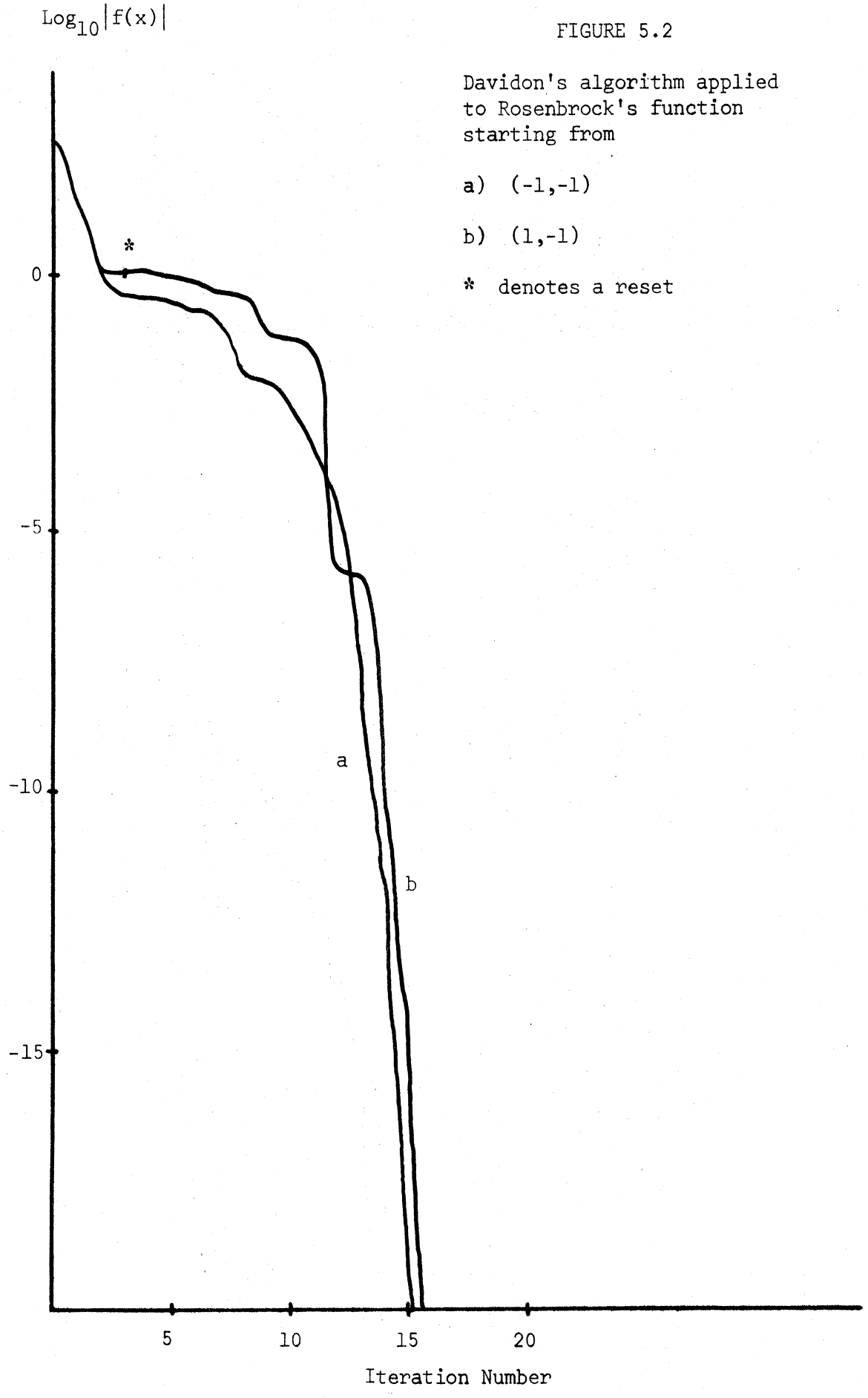


FIGURE 5.2

Davidon's algorithm applied to Rosenbrock's function starting from

a) (-1,-1)

b) (1,-1)

* denotes a reset

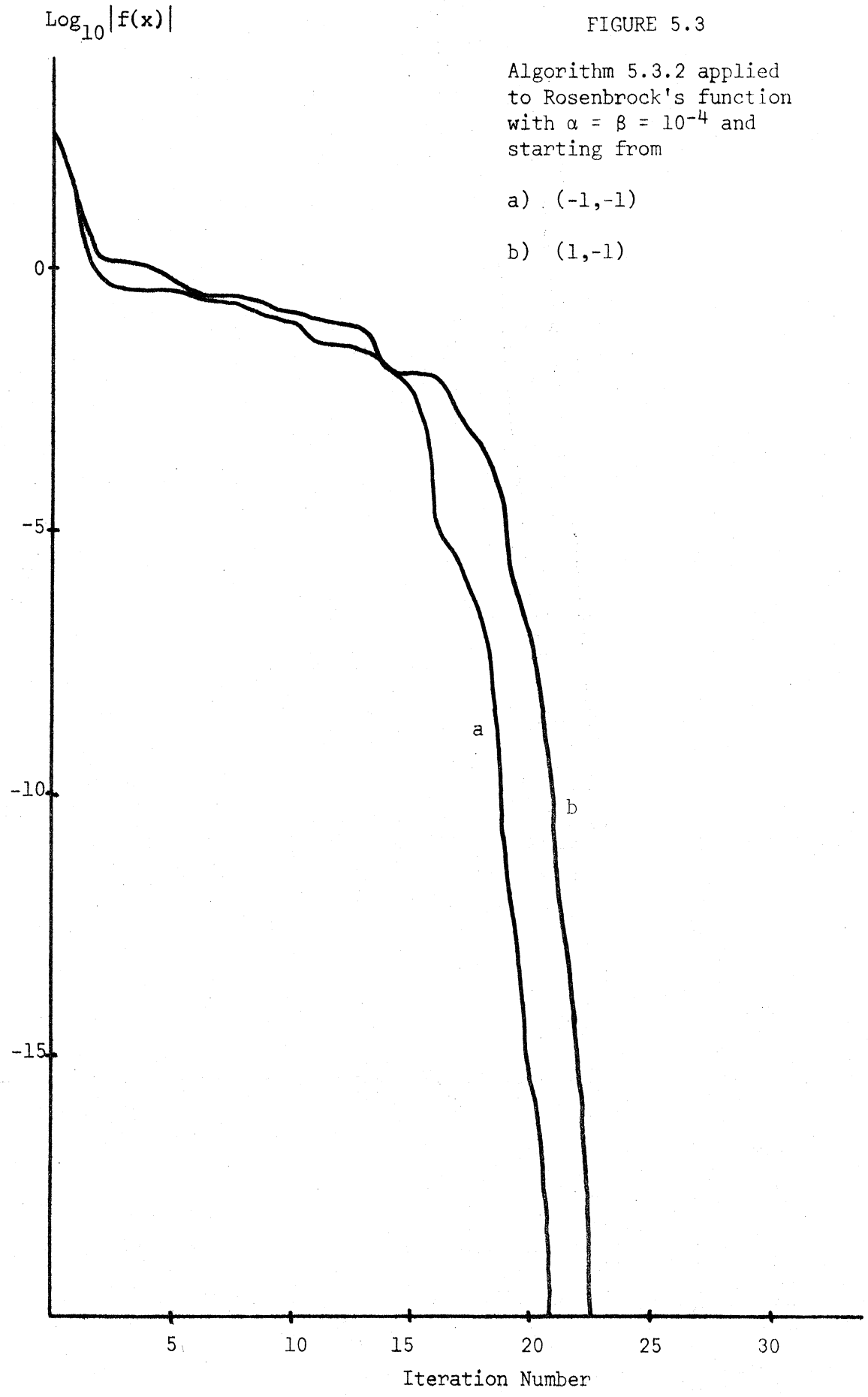


FIGURE 5.3

Algorithm 5.3.2 applied to Rosenbrock's function with $\alpha = \beta = 10^{-4}$ and starting from

a) (-1,-1)

b) (1,-1)

is such that the rank of U_k cannot decrease. For these two runs U_k became nonsingular on the second iteration and remained nonsingular. Starting at (-1,-1) algorithm 5.3.2 reduced the function value to about 10^{-20} in 22 iterations and 75 function evaluations. From (1,-1) algorithm 5.3.2 reduced the function value to about 10^{-20} in 24 iterations and 80 function evaluations. In terms of both function evaluations and total number of iterations, algorithm 5.3.2 was slightly inferior to Davidon on the Rosenbrock function from these two starting points. However, the superlinear terminal rate of convergence of algorithm 5.3.2 is verified.

In applying algorithm 5.3.1 to Rosenbrock's function, runs were made with several values of α and β. Again the starting points (-1,-1) and (1,-1) were used. Figures 5.4 and 5.5 represent the $\text{Log}_{10}|f(x)|$ versus iteration number. Figures 5.6 and 5.7 depict the age of the oldest column of U_k and V_k as a function of iteration number for the same runs.

From either starting point with $\alpha = \beta = 10^{-4}$, the age of the oldest column was two after the second iteration. Also with $\alpha = \beta = 10^{-4}$, after the second iteration U_k was nonsingular and the p_k^* step was satisfactory at each stage. On Rosenbrock's function then the sequence of matrices $V_k U_k^{-1}$ generated by algorithms 5.3.1 and 5.3.2 with $\alpha = \beta = 10^{-4}$ is the same as the sequence which would be generated by the sequential secant recursion formula if it were applied to the function minimization problem. In the particular case of Rosenbrock's function no difficulty was encountered with this formula and the algorithm converged superlinearly for reasonably small α and β. On more difficult functions, difficulties will arise because of the problems

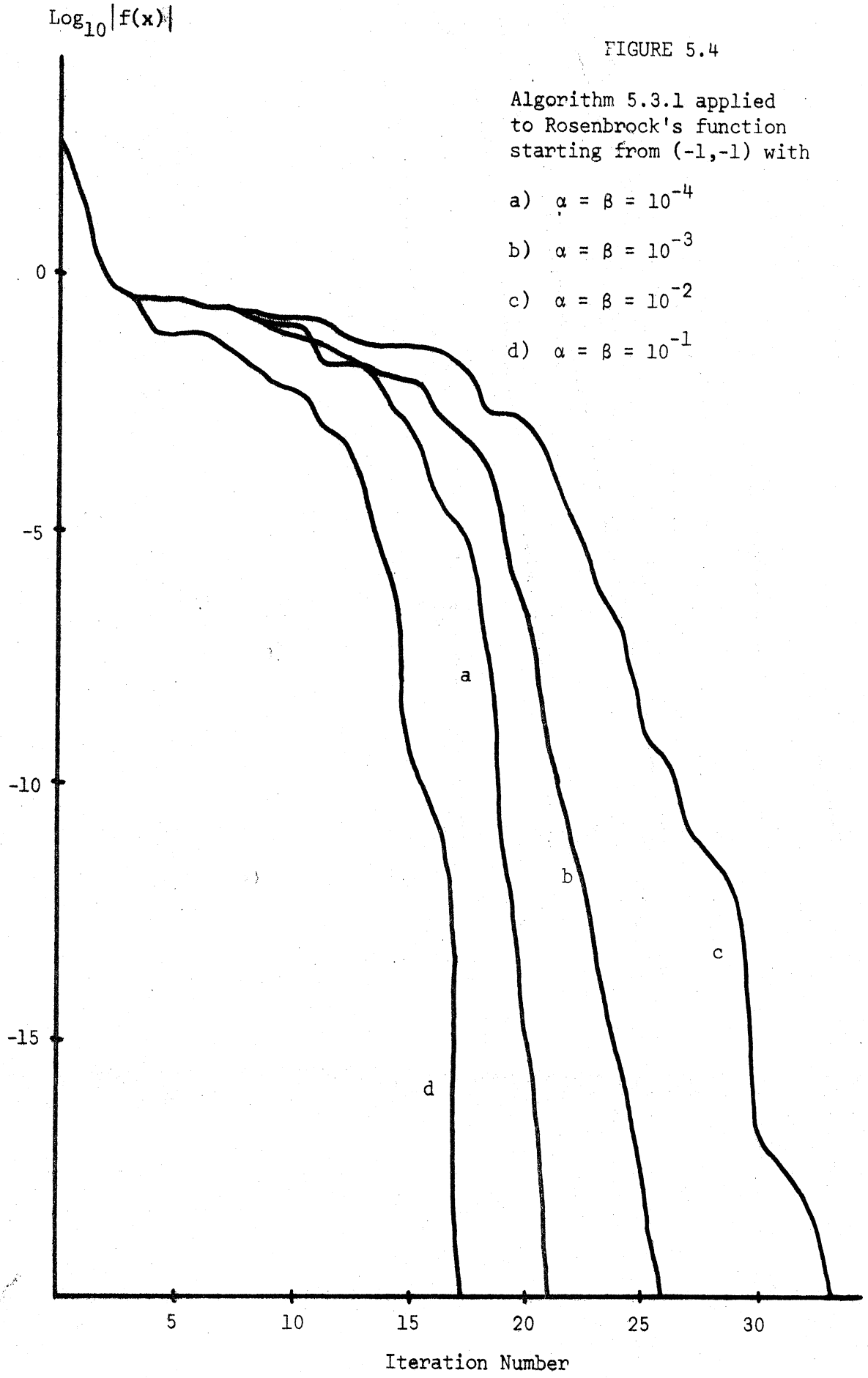


FIGURE 5.4

Algorithm 5.3.1 applied to Rosenbrock's function starting from (-1,-1) with

a) $\alpha = \beta = 10^{-4}$

b) $\alpha = \beta = 10^{-3}$

c) $\alpha = \beta = 10^{-2}$

d) $\alpha = \beta = 10^{-1}$

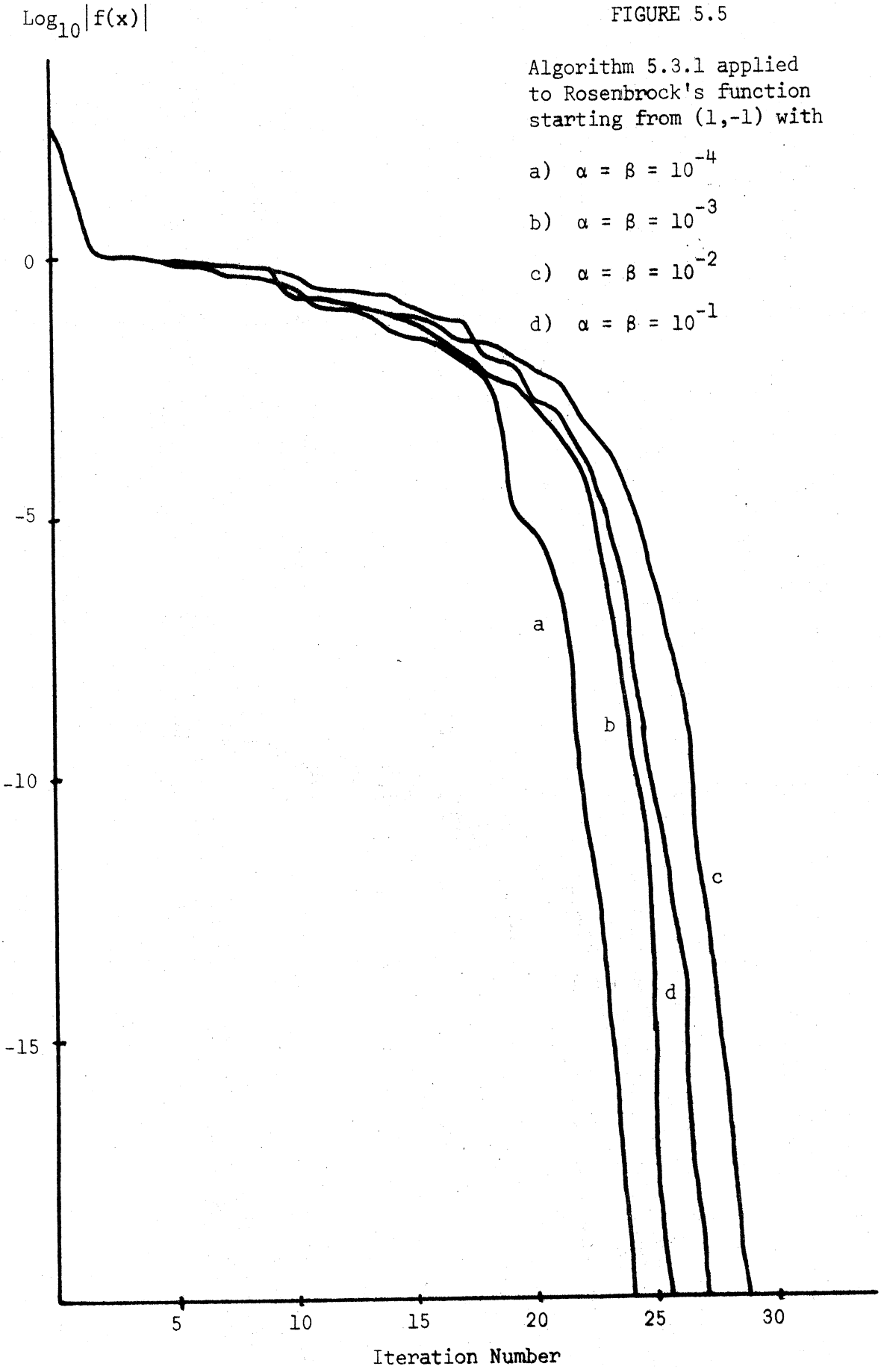


FIGURE 5.5

Algorithm 5.3.1 applied to Rosenbrock's function starting from (1,-1) with

a) $\alpha = \beta = 10^{-4}$

b) $\alpha = \beta = 10^{-3}$

c) $\alpha = \beta = 10^{-2}$

d) $\alpha = \beta = 10^{-1}$

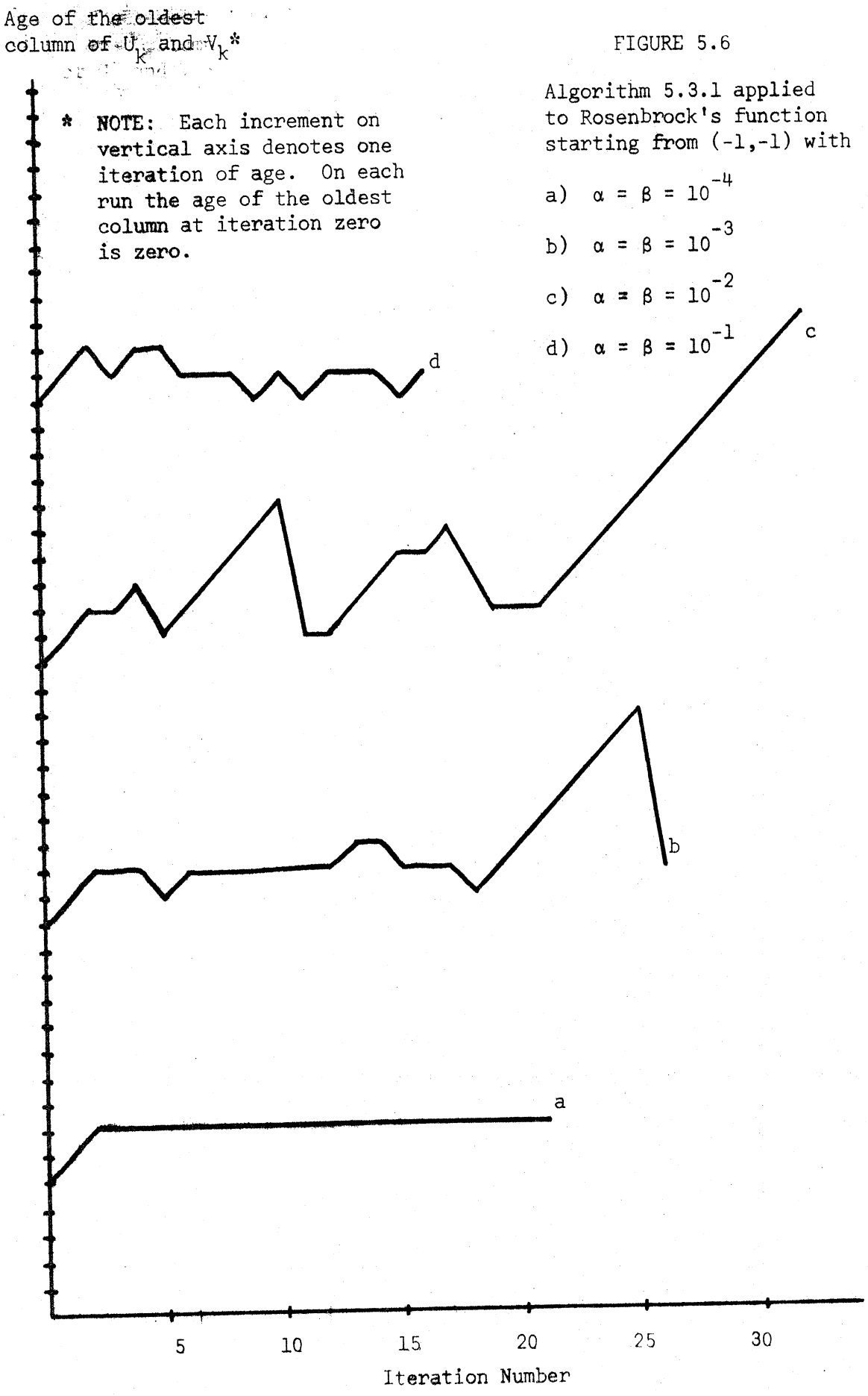


FIGURE 5.6

Algorithm 5.3.1 applied to Rosenbrock's function starting from (-1,-1) with

a) $\alpha = \beta = 10^{-4}$

b) $\alpha = \beta = 10^{-3}$

c) $\alpha = \beta = 10^{-2}$

d) $\alpha = \beta = 10^{-1}$

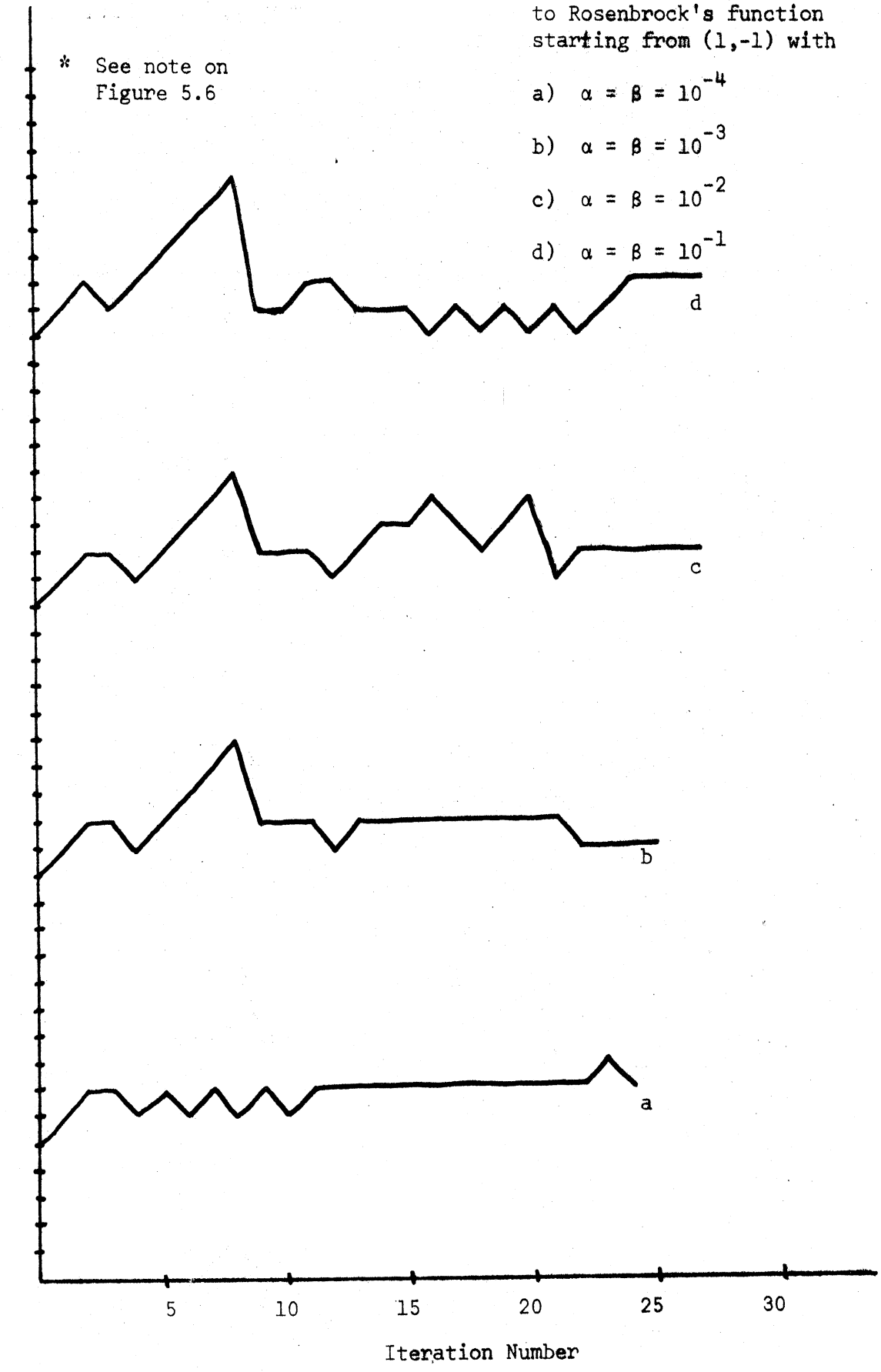


FIGURE 5.7

Algorithm 5.3.1 applied to Rosenbrock's function starting from (1,-1) with

a) $\alpha = \beta = 10^{-4}$

b) $\alpha = \beta = 10^{-3}$

c) $\alpha = \beta = 10^{-2}$

d) $\alpha = \beta = 10^{-1}$

discussed in Chapter 3 with the sequential secant algorithm.

One might expect to find a correlation between the coefficients α and β and the performance of algorithms 5.3.1 and 5.3.2. Figures 5.4 and 5.5 do not demonstrate such a correlation. Starting at $(-1,-1)$ the choices $\alpha = \beta = 10^{-1}$ and $\alpha = \beta = 10^{-4}$ yield convergence in a comparable number of iterations. Figures 5.6 and 5.7 indicate better correlation between performance and the age of the oldest column of U_k V_k.

In all cases the performance of algorithm 5.3.1 was inferior to that of Davidon on the Rosenbrock function using the starting points $(-1,-1)$ and $(1,-1)$. However when α and β are chosen small, the difference in performance is not significant. In all cases the super-linear rate of convergence of algorithms 5.3.1 and 5.3.2 is verified.

5.5 Wood's Function

The second test function investigated was Wood's function of four variables, which is given by

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 \qquad (5.5.1)$$
$$+ (1 - x_3)^2 + 10.1(x_2 - 1)^2 + 10.1(x_4 - 1)^2$$
$$+ 19.9(x_2 - 1)^2(x_4 - 1)^2.$$

The first runs made were with algorithm 5.3.2 starting at $(-3,-1,-3,-1)$ and $(-3,0,-3,-1)$. It was observed that, from both starting points with various values of α and β, algorithm 5.3.2 had a strong tendency to take repeated gradient steps. In all runs the matrix U_k became nonsingular on the 4th or 5th iteration. Following that, the direction p_k^* was uphill which caused the gradient direction to be chosen as the search direction. As there is no procedure for reducing the rank of U_k in

algorithm 5.3.2, all steps past step 5 were either p_k^* steps or gradient steps. Unfortunately, choosing a gradient step and updating the matrices U_k and V_k did not tend to make the p_k^* direction downhill. As a result the algorithm degenerated to a gradient search procedure. Although the theory of Chapter 4 predicts that (once in a small neighborhood of the minimum) the direction p_k^* will be downhill, convergence to such a neighborhood is extremely slow in algorithm 5.3.2 if the algorithm initially performs a large number of gradient direction searches. Because of this deficiency, which was quite clearly brought out on Wood's function, algorithm 5.3.2 was not investigated further. Instead, attention was restricted to algorithm 5.3.1.

In applying algorithm 5.3.1 to Wood's function a further effort was made to determine the relationship between the performance of the algorithm and the coefficients α and β. Starting from the point $(-3,0,-3,-1)$ two runs were made using algorithm 5.3.1. Coefficients α and β were chosen small in one run and relatively large in the other. In Figure 5.8, $\text{Log}_{10}|f(x)|$ is plotted versus iteration number for these two runs and for the Davidon run. Notice that in this case the larger choice of α and β caused algorithm 5.3.1 to converge poorly.

From the starting point $(-3,-1,-3,-1)$ a large number of runs were made with values of α and β varying between 10^{-4} and 10^{-1}. Figures 5.9, 5.10, and 5.11 display $\text{Log}_{10}|f(x)|$ versus iteration number for $\alpha = 10^{-1}$, 10^{-2}, and 10^{-4} respectively with several values of β in each case. The corresponding Davidon run is shown in Figure 5.11. In Figures 5.12, 5.13, and 5.14 the rank of U_k and the age of the oldest column of U_k and V_k are plotted as a function of iteration number for the same family of runs. The number of iterations required for convergence generally seemed

FIGURE 5.8

a) Davidon's algorithm applied to Wood's function starting from (-3,0,-3,-1).

b) Algorithm 5.3.1 applied to Wood's function starting from (-3,0,-3,-1) with $\alpha = 10^{-4}$ and $\beta = 10^{-3}$.

c) Same as b) except $\alpha = 10^{-1}$ and $\beta = .4 \times 10^{-1}$

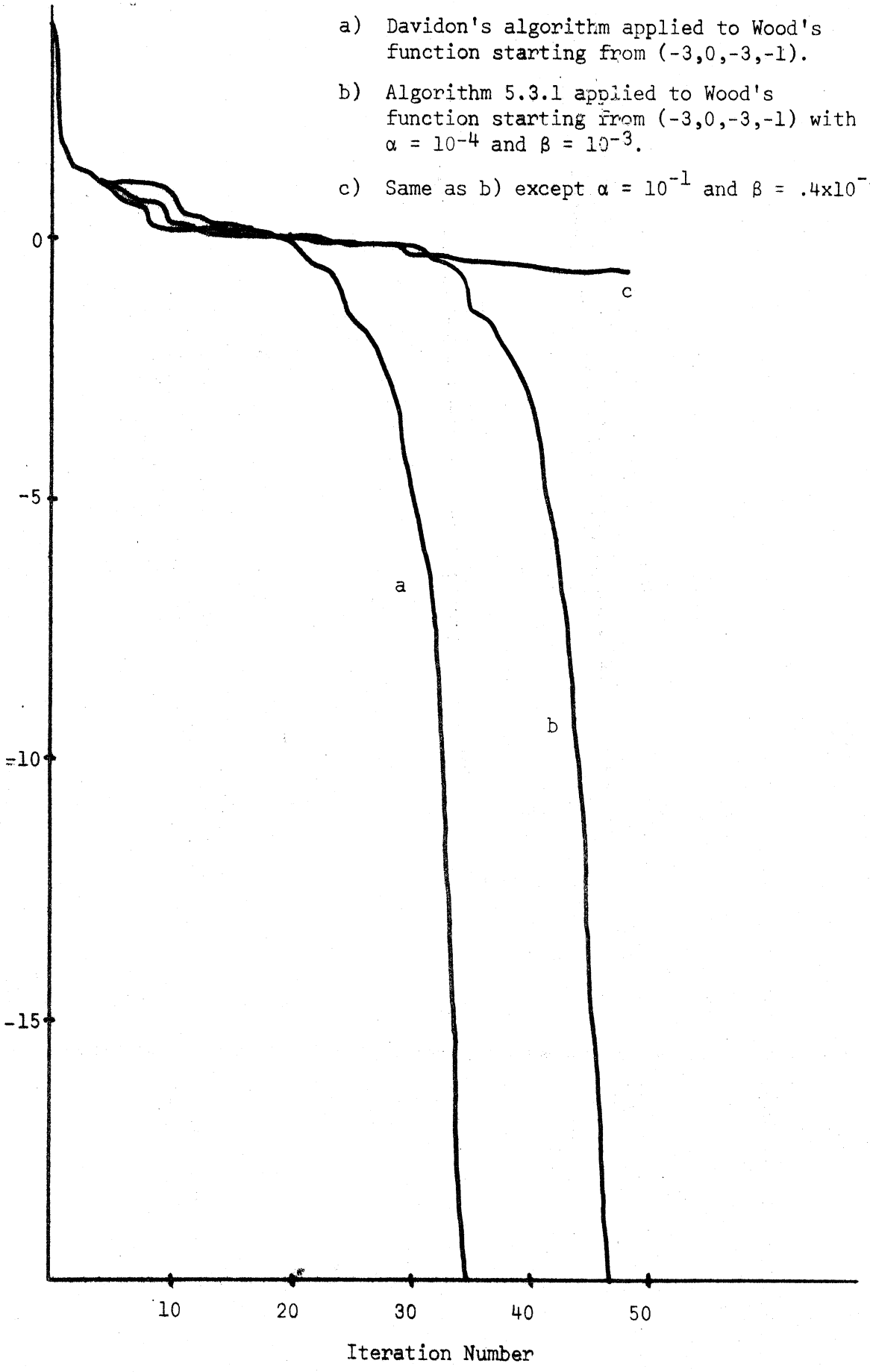


FIGURE 5.9

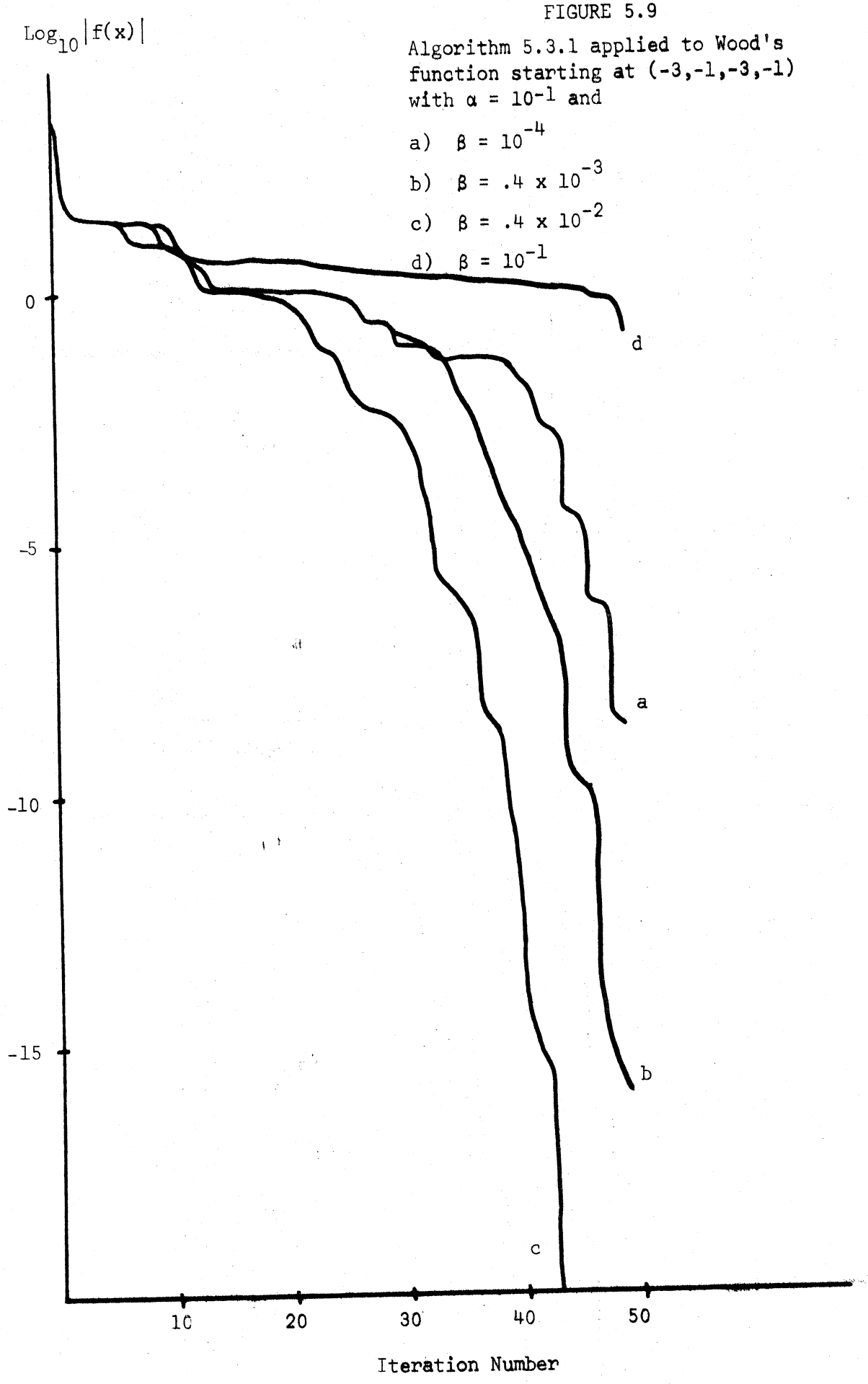
Algorithm 5.3.1 applied to Wood's function starting at (-3,-1,-3,-1) with $\alpha = 10^{-1}$ and

a) $\beta = 10^{-4}$

b) $\beta = .4 \times 10^{-3}$

c) $\beta = .4 \times 10^{-2}$

d) $\beta = 10^{-1}$



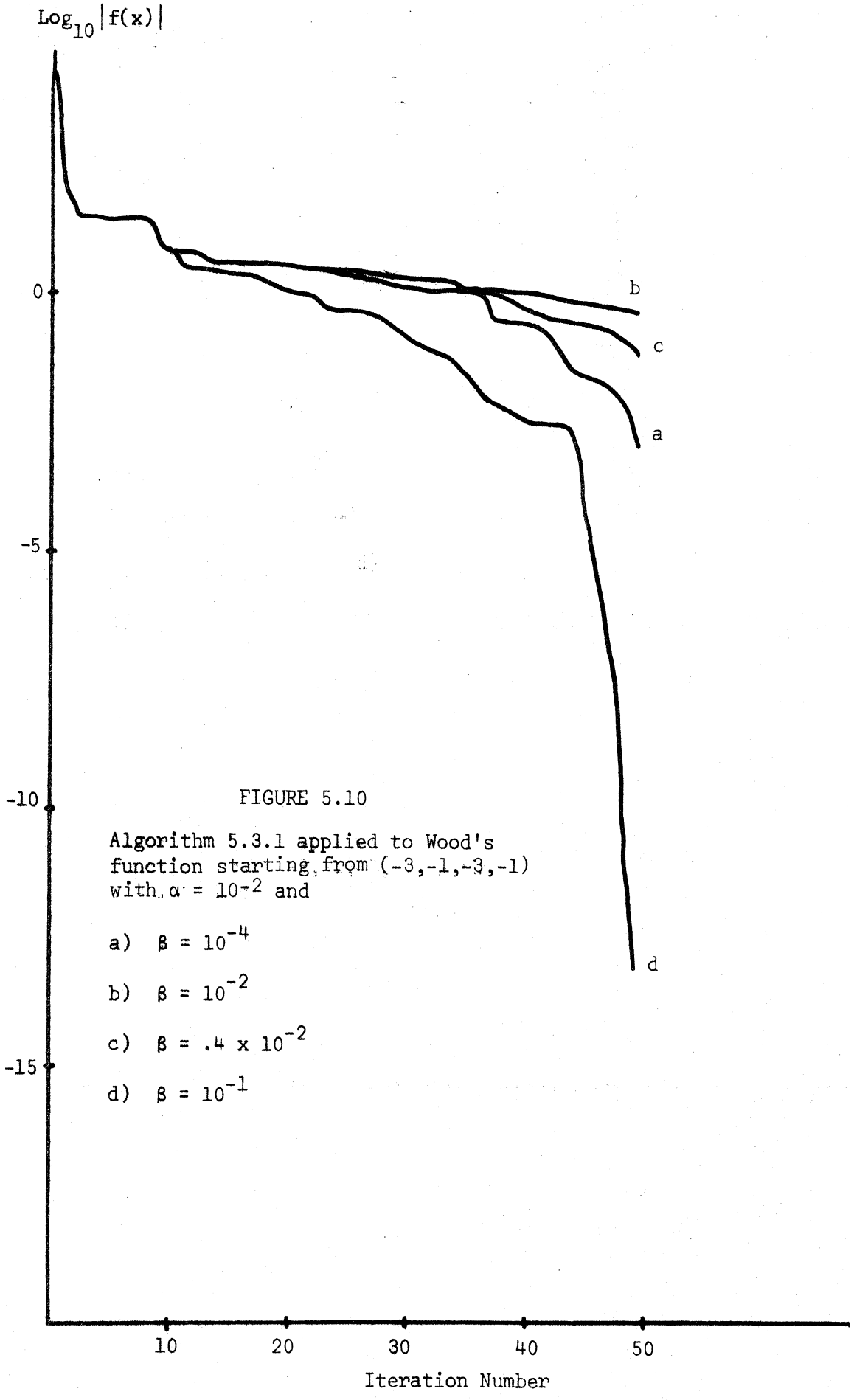


FIGURE 5.10

Algorithm 5.3.1 applied to Wood's function starting from (-3,-1,-3,-1) with $\alpha = 10^{-2}$ and

a) $\beta = 10^{-4}$

b) $\beta = 10^{-2}$

c) $\beta = .4 \times 10^{-2}$

d) $\beta = 10^{-1}$

FIGURE 5.11

Algorithm 5.3.1 applied to Wood's function starting from (-1,-3,-1,-3) with $\alpha = 10^{-4}$ and

a) $\beta = .4 \times 10^{-4}$, $\beta = 10^{-2}$

b) $\beta = .4 \times 10^{-1}$

c) $\beta = 10^{-1}$

d) Davidon's algorithm applied to Wood's function starting from (-1,-3,-1,-3).

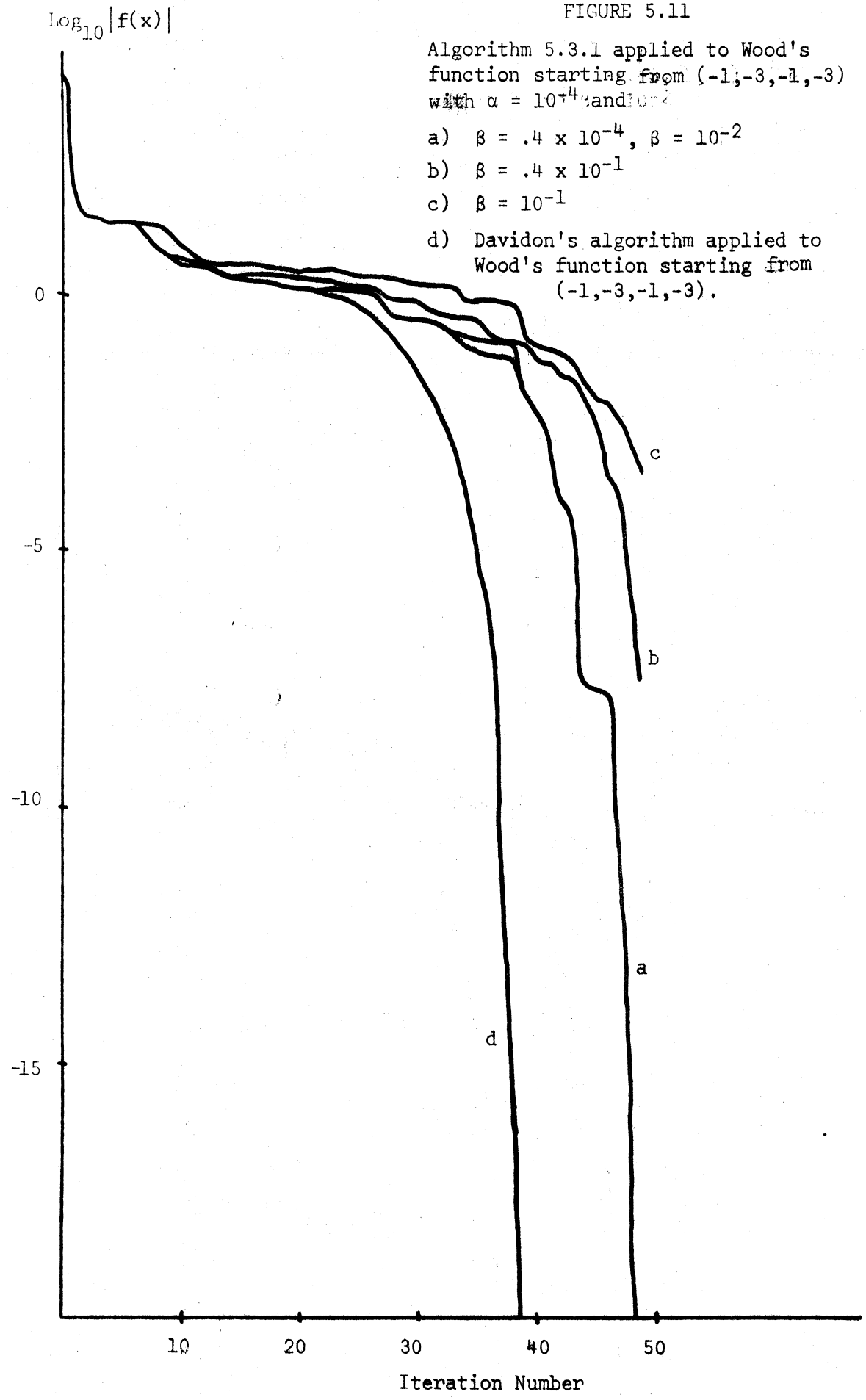


FIGURE 5.12

Age of the oldest column of U_k and V_k and rank of U_k*

* NOTE. Continuous functions are age and discontinuous functions are rank. Each vertical increment indicates one iteration of age or one rank change. Age and rank are zero at iteration zero.

Algorithm 5.3.1 applied to Wood's function starting at (-1,-3,-1,-3) with $\alpha = 10^{-1}$ and

a) $\beta = 10^{-4}$

b) $\beta = .4 \times 10^{-3}$

c) $\beta = .4 \times 10^{-2}$

d) $\beta = 10^{-1}$

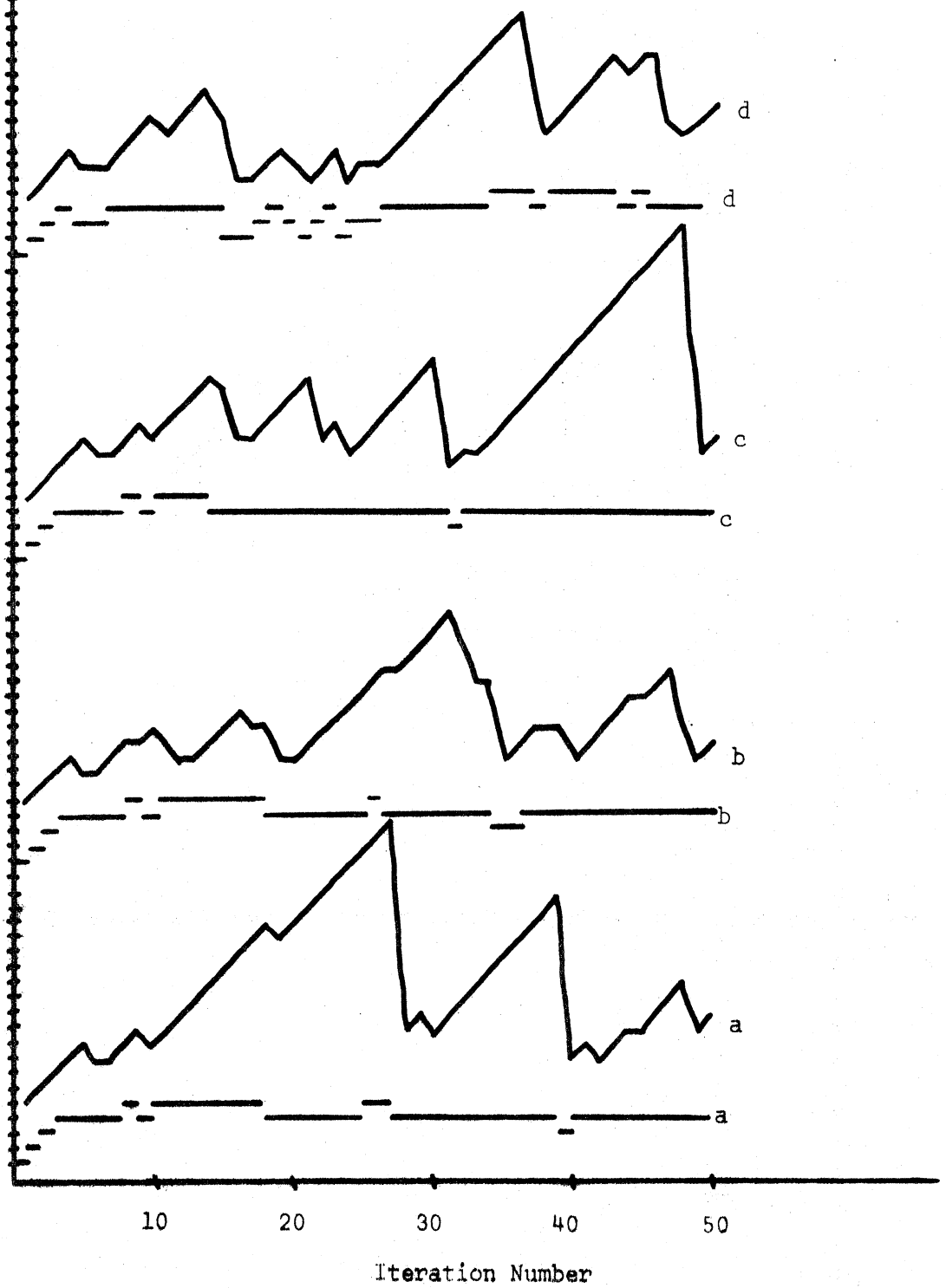


FIGURE 5.13

Algorithm 5.3.1 applied to Wood's function starting at (-1,-3,-1,-3) with $\alpha = 10^{-2}$ and

a) $\beta = 10^{-4}$

b) $\beta = 10^{-2}$

c) $\beta = .4 \times 10^{-2}$

d) $\beta = 10^{-1}$

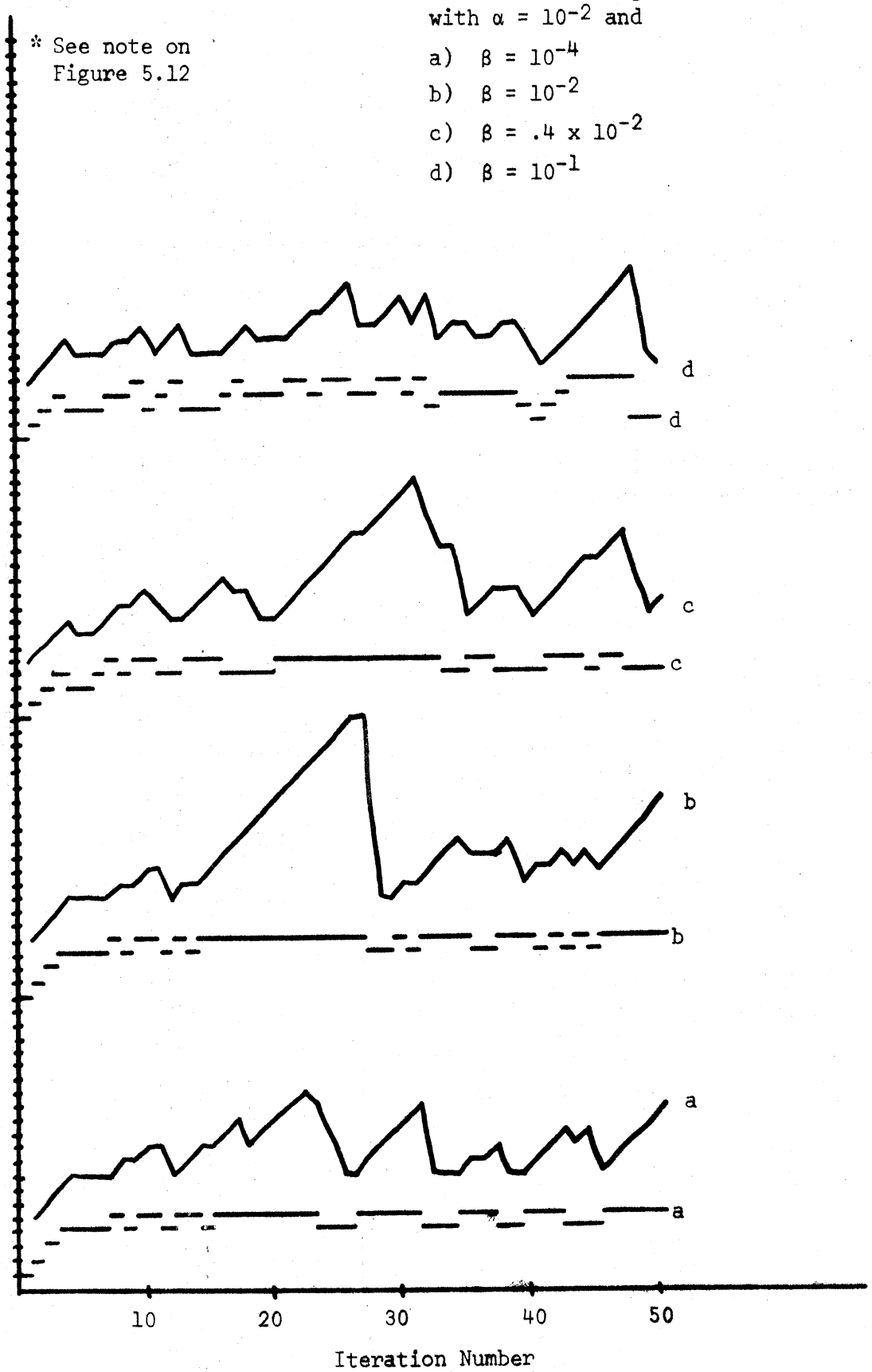


FIGURE 5.14

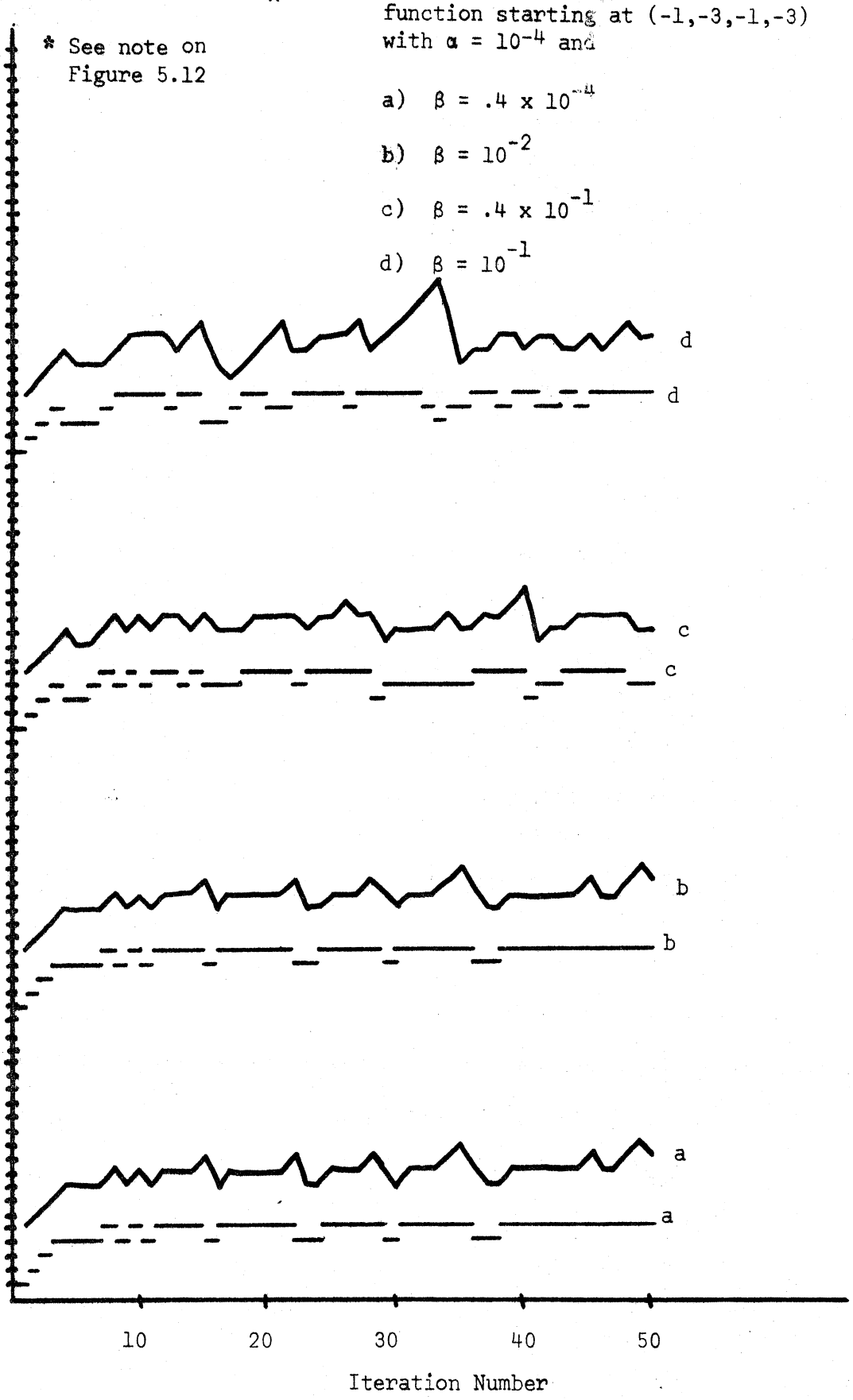
Algorithm 5.3.1 applied to Wood's function starting at (-1,-3,-1,-3) with $\alpha = 10^{-4}$ and

a) $\beta = .4 \times 10^{-4}$

b) $\beta = 10^{-2}$

c) $\beta = .4 \times 10^{-1}$

d) $\beta = 10^{-1}$



to decrease with decreasing α and β. There are, however, some notable exceptions. The run in Figure 5.9 with $\alpha = 10^{-1}$ and $\beta = .4 \times 10^{-2}$, for instance, converged in the fewest number of iterations yet $\alpha = 10^{-1}$ was the largest value of α tested. Similarly, the relationship between the age of the oldest column of U_k and V_k and performance of the algorithm is not entirely consistent. Although the average age of the oldest column does decrease with decreasing α in some cases where good performance was noted very old data was retained. Also, in some runs where the algorithm performed poorly there was no accumulation of old data. Because the constant β determines when the rank of U_k is to be decreased, one would expect that the rank of U_k would tend to increase on the average as β was chosen smaller. This does seem to be the case, although the relationship is not very pronounced.

In many applications the amount of computation required to evaluate the function and its gradient is large when compared with the computations which occur in the algorithm. If this is the case then a more valid measure of performance of the algorithm is the number of function (gradient) evaluations required to reach the minimum. In Figures 5.15 and 5.16 $\text{Log}_{10}|f(x)|$ is plotted versus the number of function (gradient) evaluations starting from $(-3,0,-3,-1)$ and $(-3,-1,-3,-1)$, respectively for algorithm 5.3.1 and for Davidon. All the curves correspond to runs presented earlier in Figures 5.9, 5.10, and 5.11. In all cases except one, algorithm 5.3.1 performs better than Davidon using this measure of performance. In particular, notice that from both starting points for small α and β ($\alpha = 10^{-4}$, $\beta = 10^{-3}$, 10^{-4}) algorithm 5.3.1 requires fewer function (gradient) evaluations. In these cases it appears as though algorithm 5.3.1 provides a slightly better

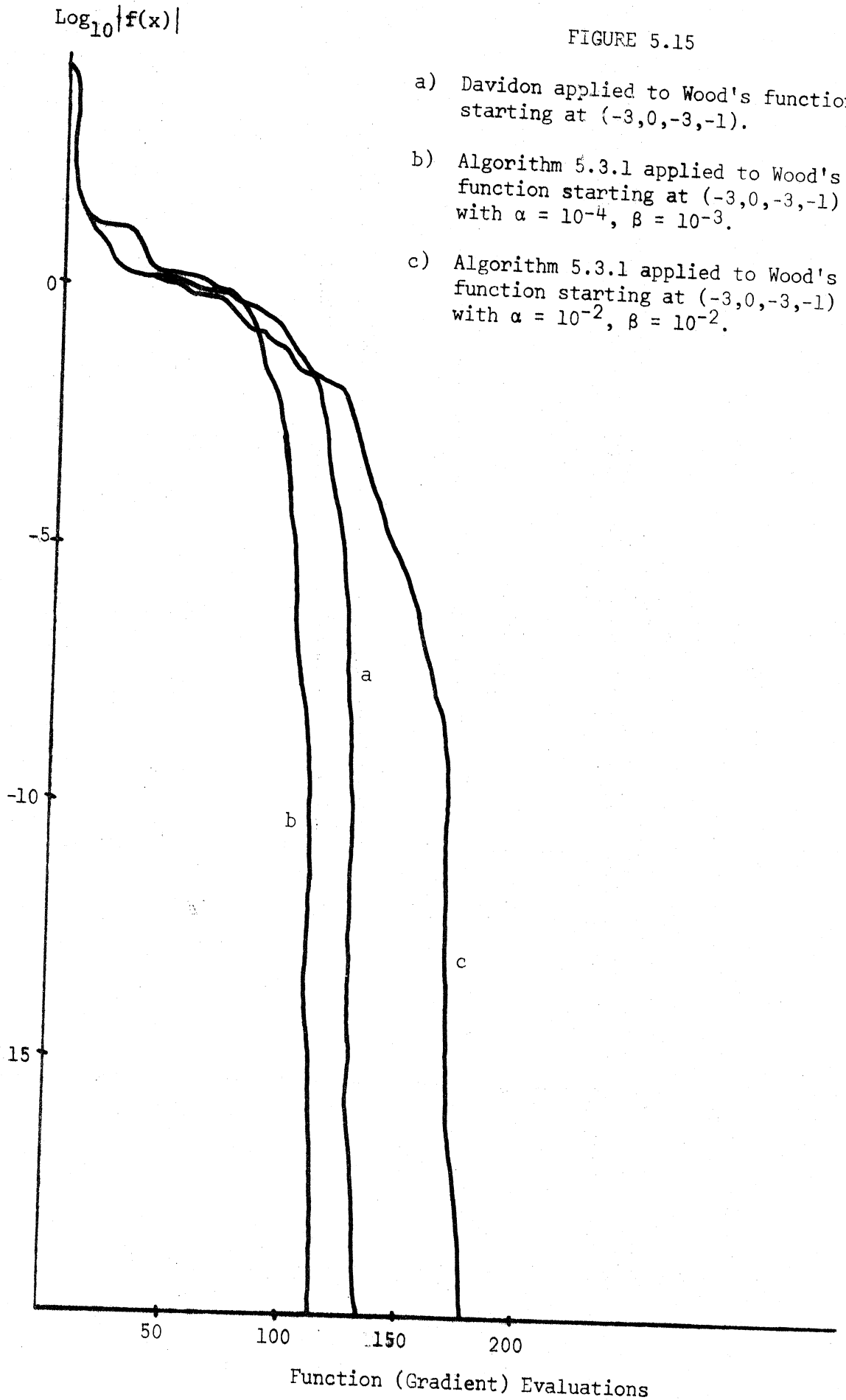


FIGURE 5.15

a) Davidon applied to Wood's function starting at (-3,0,-3,-1).

b) Algorithm 5.3.1 applied to Wood's function starting at (-3,0,-3,-1) with $\alpha = 10^{-4}$, $\beta = 10^{-3}$.

c) Algorithm 5.3.1 applied to Wood's function starting at (-3,0,-3,-1) with $\alpha = 10^{-2}$, $\beta = 10^{-2}$.

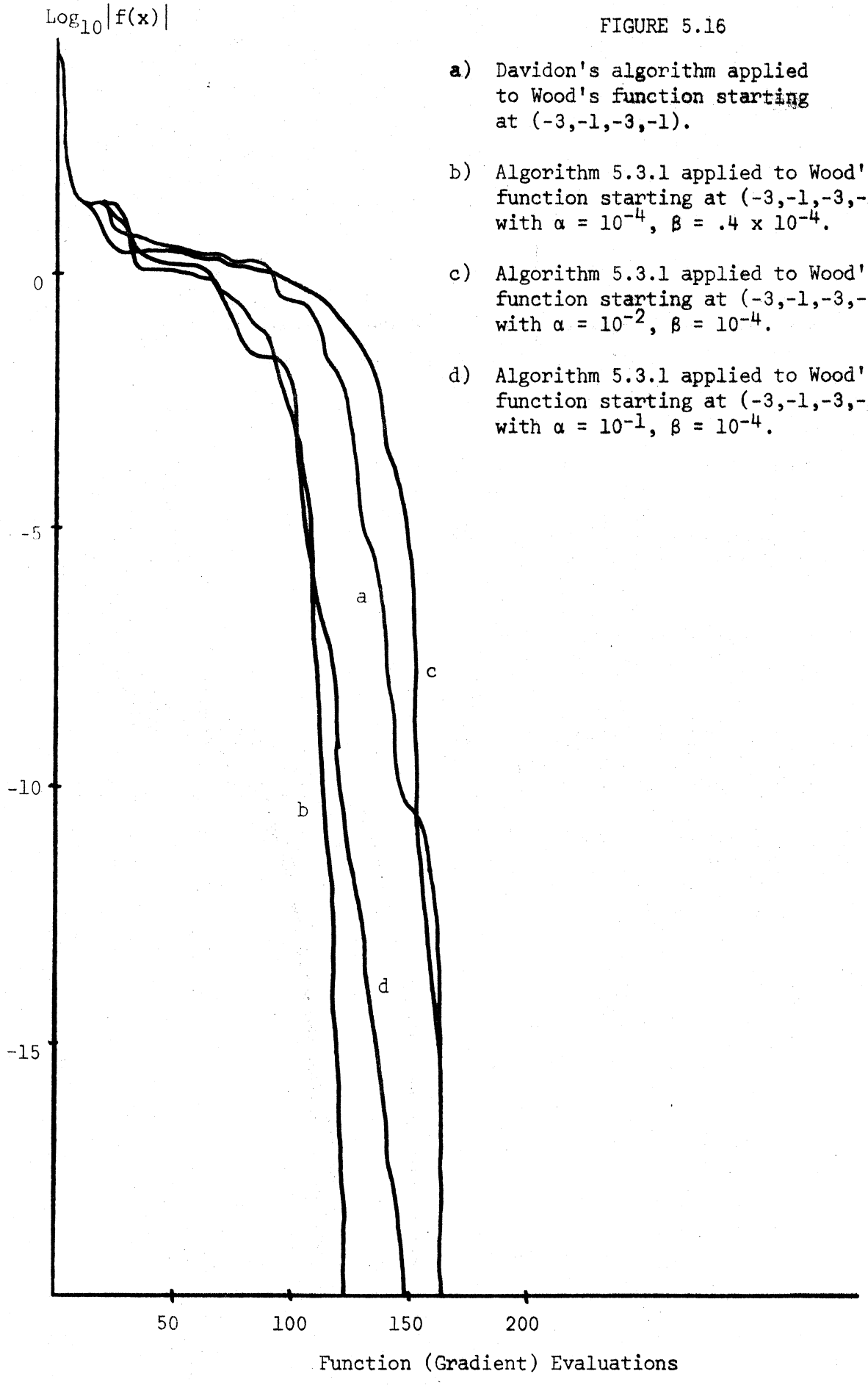


FIGURE 5.16

a) Davidon's algorithm applied to Wood's function starting at (-3,-1,-3,-1).

b) Algorithm 5.3.1 applied to Wood's function starting at (-3,-1,-3,-1) with $\alpha = 10^{-4}$, $\beta = .4 \times 10^{-4}$.

c) Algorithm 5.3.1 applied to Wood's function starting at (-3,-1,-3,-1) with $\alpha = 10^{-2}$, $\beta = 10^{-4}$.

d) Algorithm 5.3.1 applied to Wood's function starting at (-3,-1,-3,-1) with $\alpha = 10^{-1}$, $\beta = 10^{-4}$.

estimate of the distance to the minimum along the search direction so that the linear search subalgorithm makes fewer function (gradient) evaluations. This may be due to the fact that algorithm 4.3.2 does not depend upon an exact linear minimization for its theoretical properties.

5.6 Powell's Function

The four dimensional function given by

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \quad (5.6.1)$$

is particularly interesting because the Hessian matrix at the unique minimum of $f(x)$ is singular. The results of Chapter 4 do not apply to a function with a singular Hessian at the minimum. Further, no theoretical results have been obtained for the Davidon algorithm in this case. Functions with singular or ill-conditioned Hessians do arise in applications.

Runs were made from four different starting points with Davidon's algorithm and algorithm 5.3.1. In Figures 5.17 and 5.18, $\text{Log}_{10}|f(x)|$ is plotted as a function of iteration number for Davidon's algorithm and algorithm 5.3.1, respectively. Figure 5.19 is a plot of the rank of U_k and the age of the oldest column of U_k and V_k as a function of iteration number for the four runs of Figure 5.18. In these four runs of algorithm 5.3.1, $\alpha = \beta = 10^{-4}$. In all four cases Davidon's algorithm reduced the function value to 10^{-20} within 50 iterations. It is interesting that the rate of convergence seems to be linear in the Davidon run rather than superlinear. This suggests that it may be possible to obtain some convergence and rate of convergence results for Davidon's algorithm when the Hessian is singular at the minimum.

With $\alpha = \beta = 10^{-4}$, algorithm 5.3.1 behaved much like Davidon's

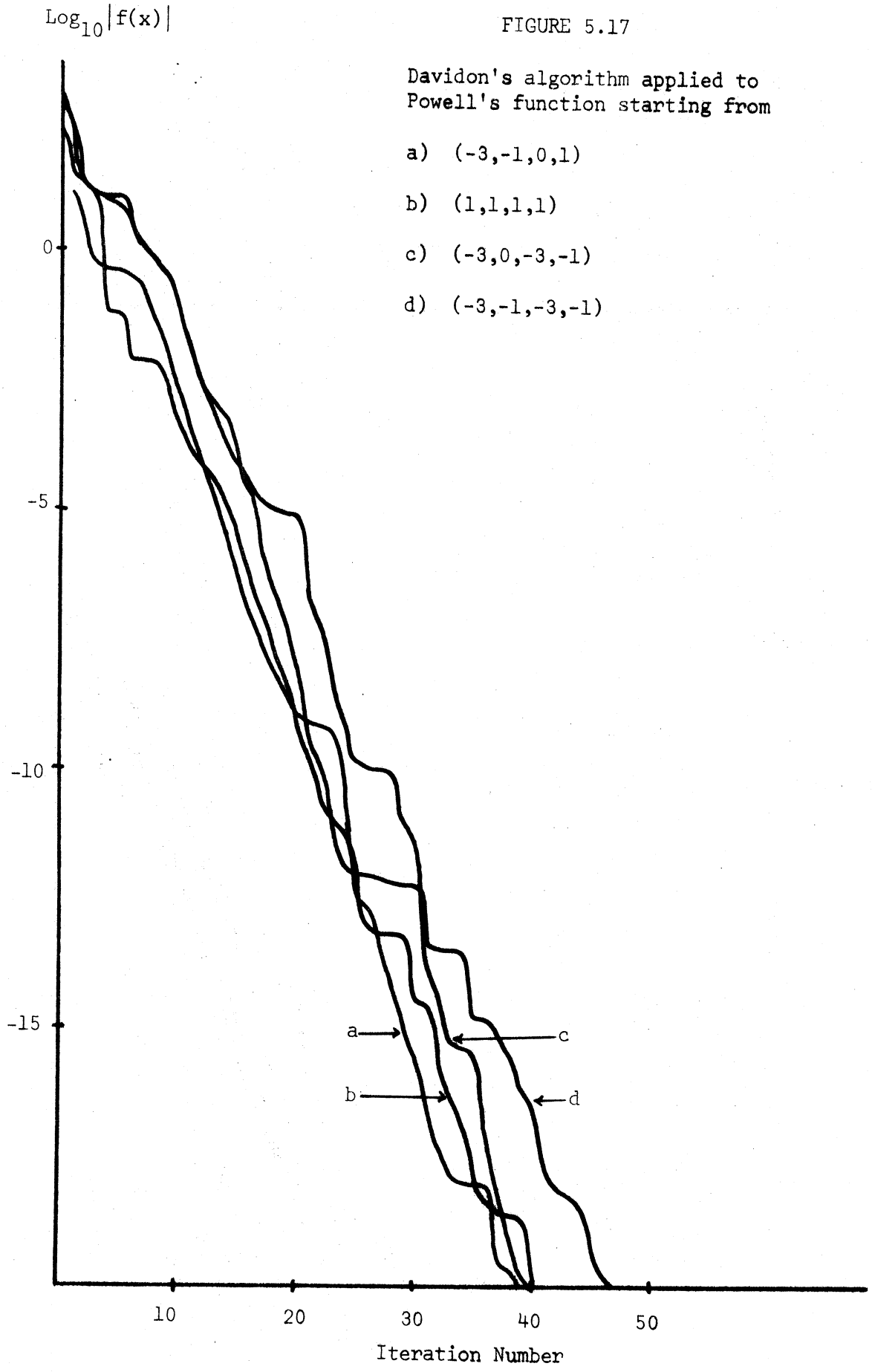


FIGURE 5.17

Davidon's algorithm applied to Powell's function starting from

a) (-3,-1,0,1)

b) (1,1,1,1)

c) (-3,0,-3,-1)

d) (-3,-1,-3,-1)

FIGURE 5.18

Algorithm 5.3.1 applied to Powell's function with $\alpha = \beta = 10^{-4}$ starting from

a) (-3,-1,0,1)

b) (1,1,1,1)

c) (-3,0,-3,-1)

d) (-3,-1,-3,-1)

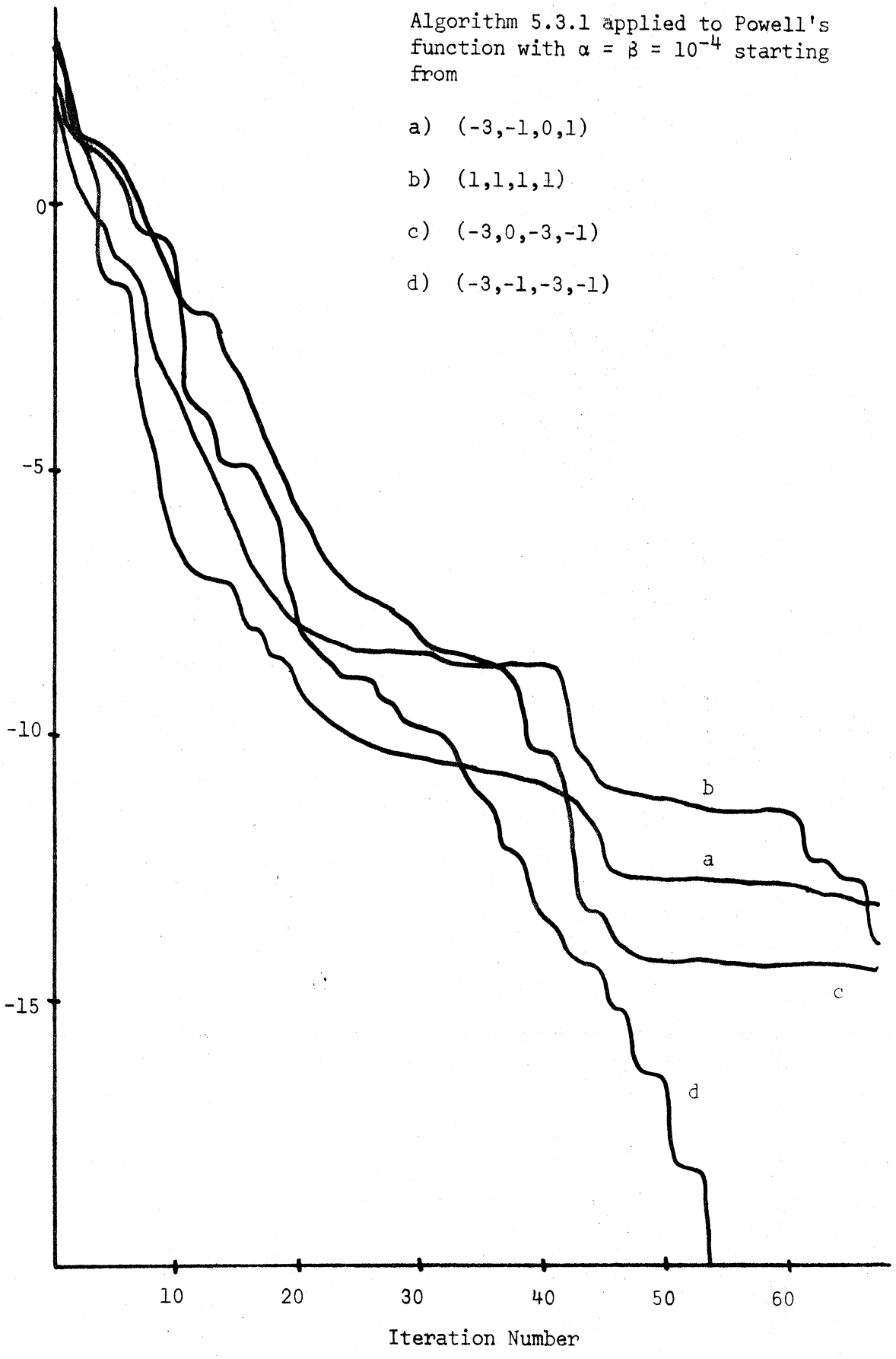


FIGURE 5.19

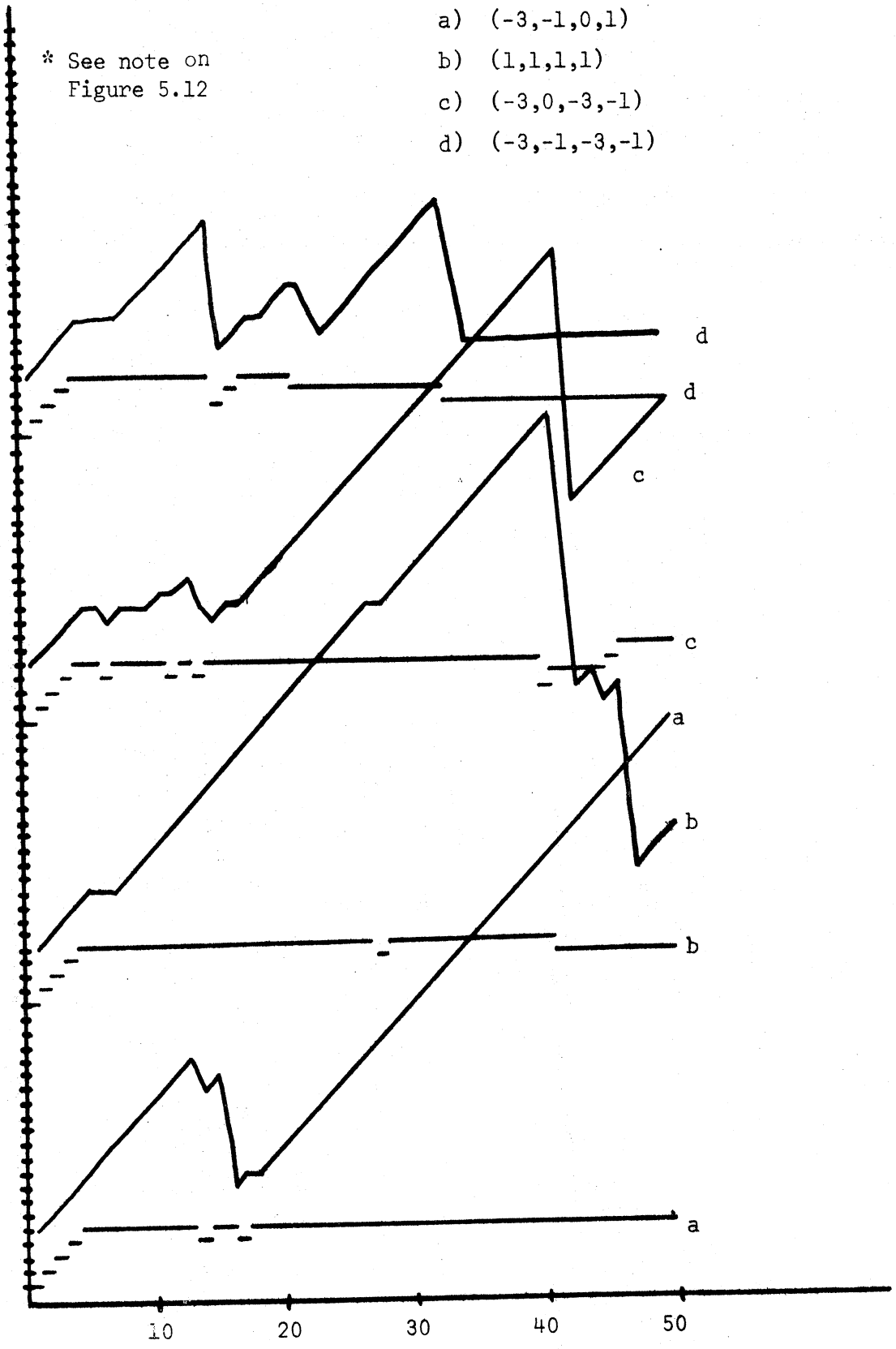
Algorithm 5.3.1 applied to Powell's function with $\alpha = \beta = 10^{-4}$ starting from

a) (-3,-1,0,1)

b) (1,1,1,1)

c) (-3,0,-3,-1)

d) (-3,-1,-3,-1)



algorithm for the first 20 iterations. After 20 iterations algorithm 5.3.1 ran into difficulty. In three of the four runs it became impossible to delete any single column of the matrices U_k and V_k and to add the new columns (u_k, v_k) while satisfying the angle condition of step 8 of the algorithm. As a result the age of the columns of U_k and V_k grew quite large. In the one case where good convergence was obtained the age of the columns did not become large. However, in the final iterations the rank of U_k remained at two.

In an attempt to improve this performance the coefficients α and β were reduced to 10^{-8} and the runs were repeated. Figure 5.20 depicts $\text{Log}_{10}|f(x)|$ versus iteration number for these runs. In Figure 5.21 the age of the oldest column and the rank of U_k are plotted versus iteration number for the same runs. Although the tendency of the algorithm to become stuck is somewhat less for these runs, in each case the age of all columns of U_k and V_k grew without bound. Again this occurred because it was impossible to satisfy the angle condition by deleting a single column of U_k and V_k.

Because of these runs it was decided to impose an explicit age bound on the columns of U_k and V_k. Figures 5.22 and 5.23 depict the runs made from the same starting points with $\alpha = \beta = 10^{-8}$ and with the age of the oldest column of U_k and V_k constrained to be less than 2n. This modification improved the performance of algorithm 5.3.1 on Powell's function considerably. From two of the starting points convergence is obtained in considerably fewer iterations than with Davidon's algorithm. From the other two, algorithm 5.3.1 requires a few more iterations to converge.

The linear search algorithm 5.2.1 did not perform well on Powell's

FIGURE 5.20

Algorithm 5.3.1 applied to Powell's function with $\alpha = \beta = 10^{-8}$ starting from

a) (-3,-1,0,1)

b) (1,1,1,1)

c) (-3,0,-3,-1)

d) (-3,-1,-3,-1)

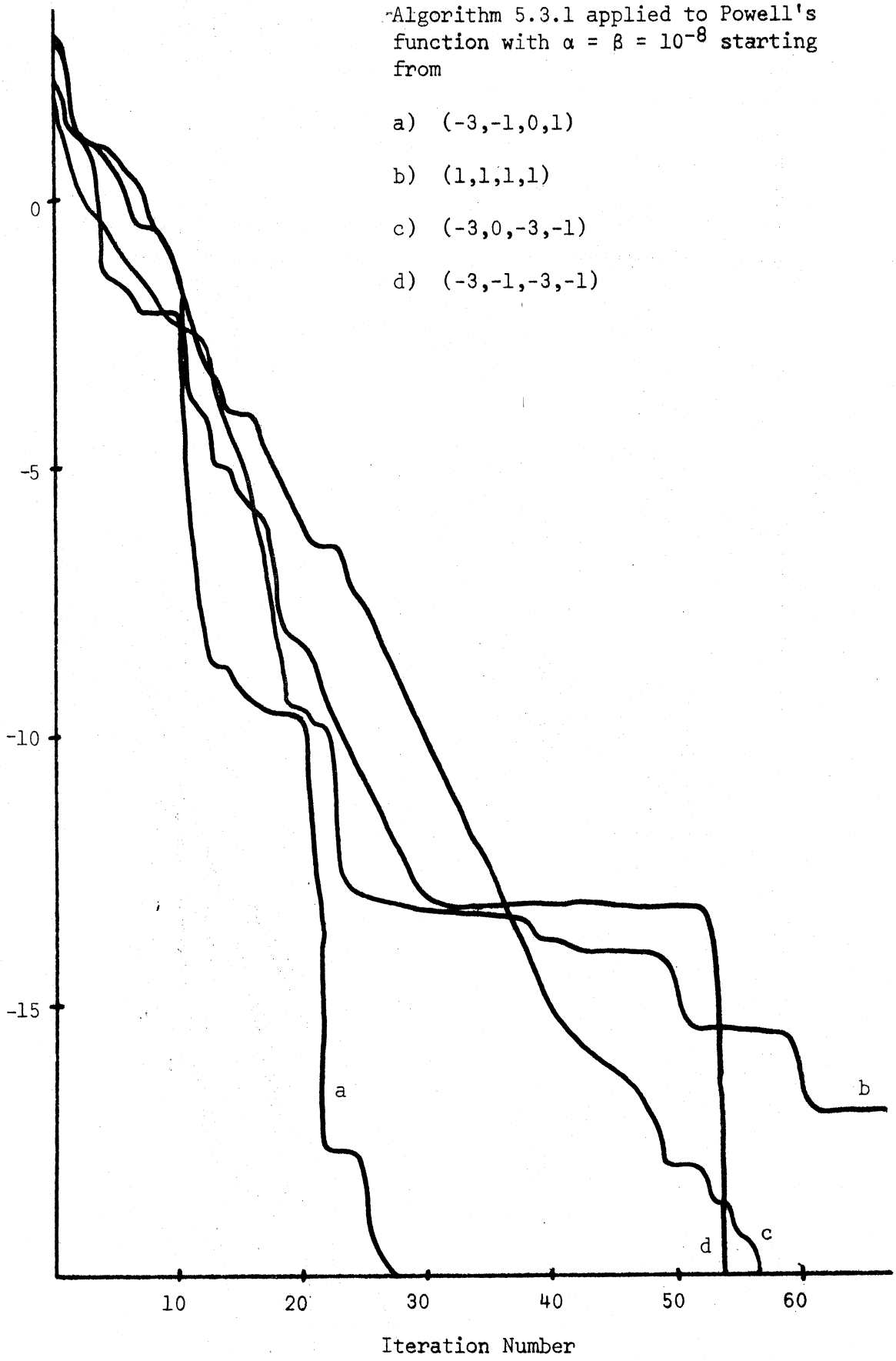


FIGURE 5.21

Age of the oldest column of U_k and V_k and rank of U_k*

* See note on Figure 5.12

Algorithm 5.3.1 applied to Powell's function with $\alpha = \beta = 10^{-8}$

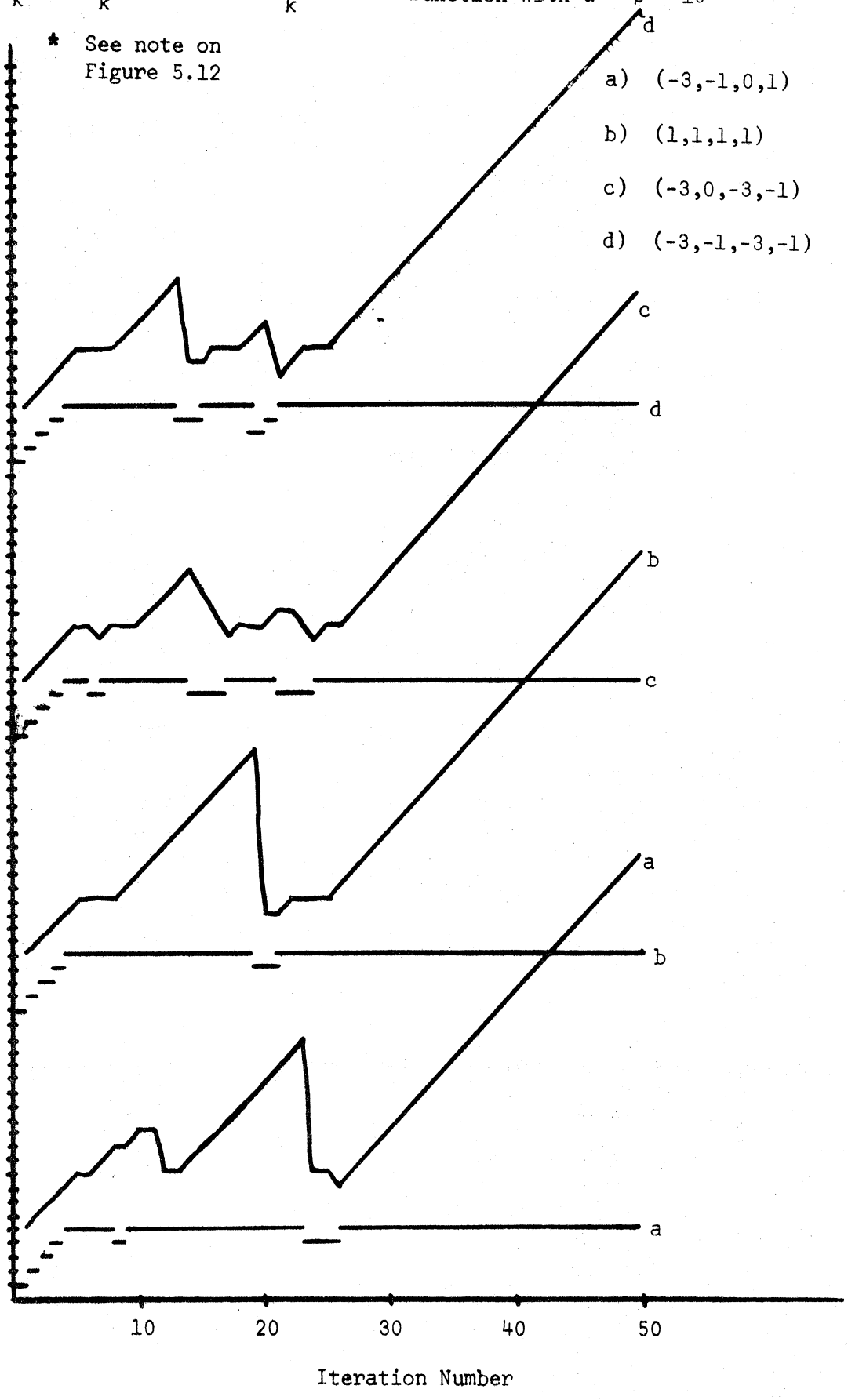


FIGURE 5.22

Algorithm 5.3.1 applied to Powell's function with $\alpha = \beta = 10^{-8}$ and with the age of the oldest column of U_k and V_k bounded less than $2n = 8$, starting from

a) (-3,-1,0,1)

b) (1,1,1,1)

c) (-3,0,-3,-1)

d) (-3,-1,-3,-1)

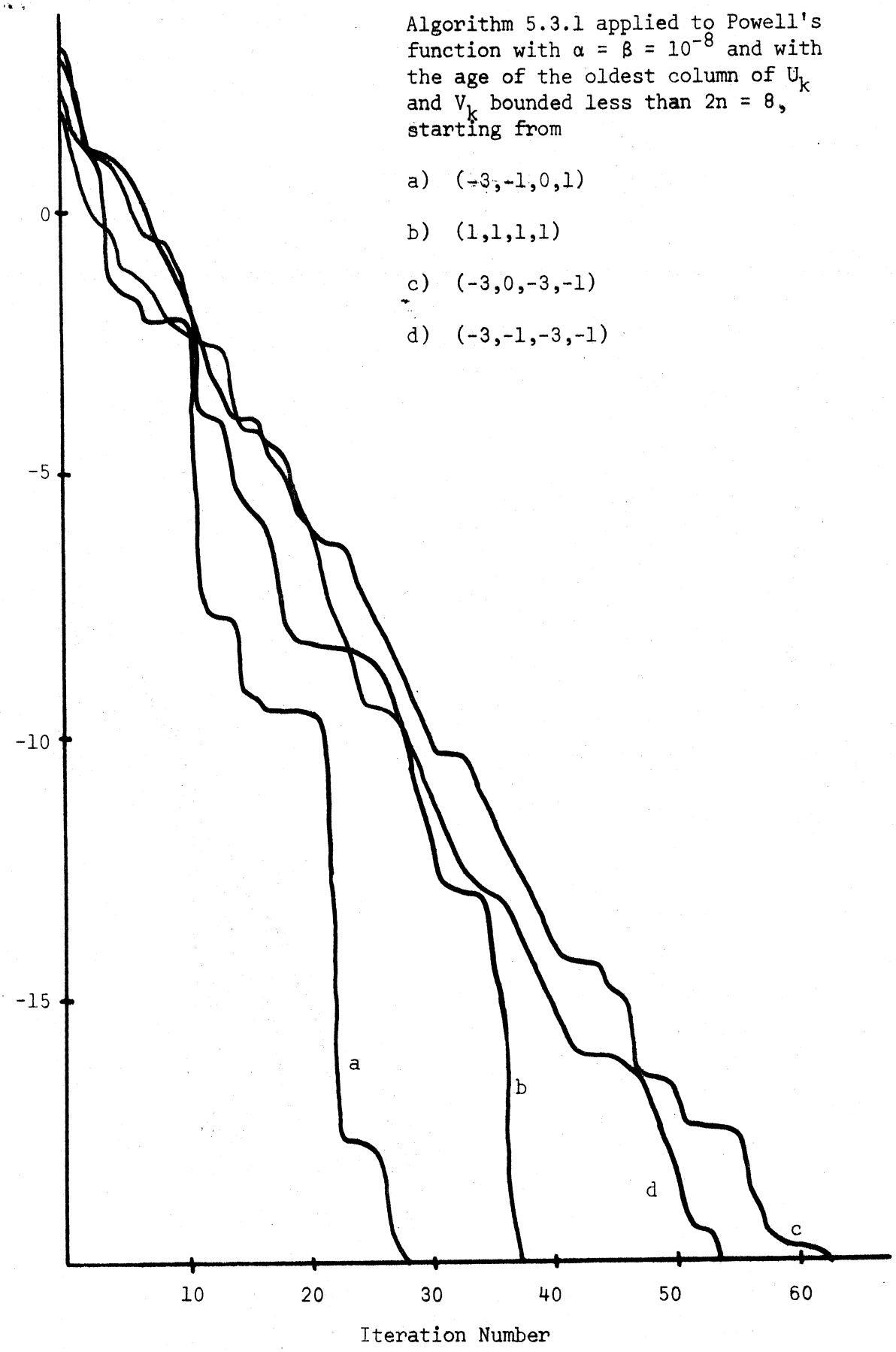


FIGURE 5.23

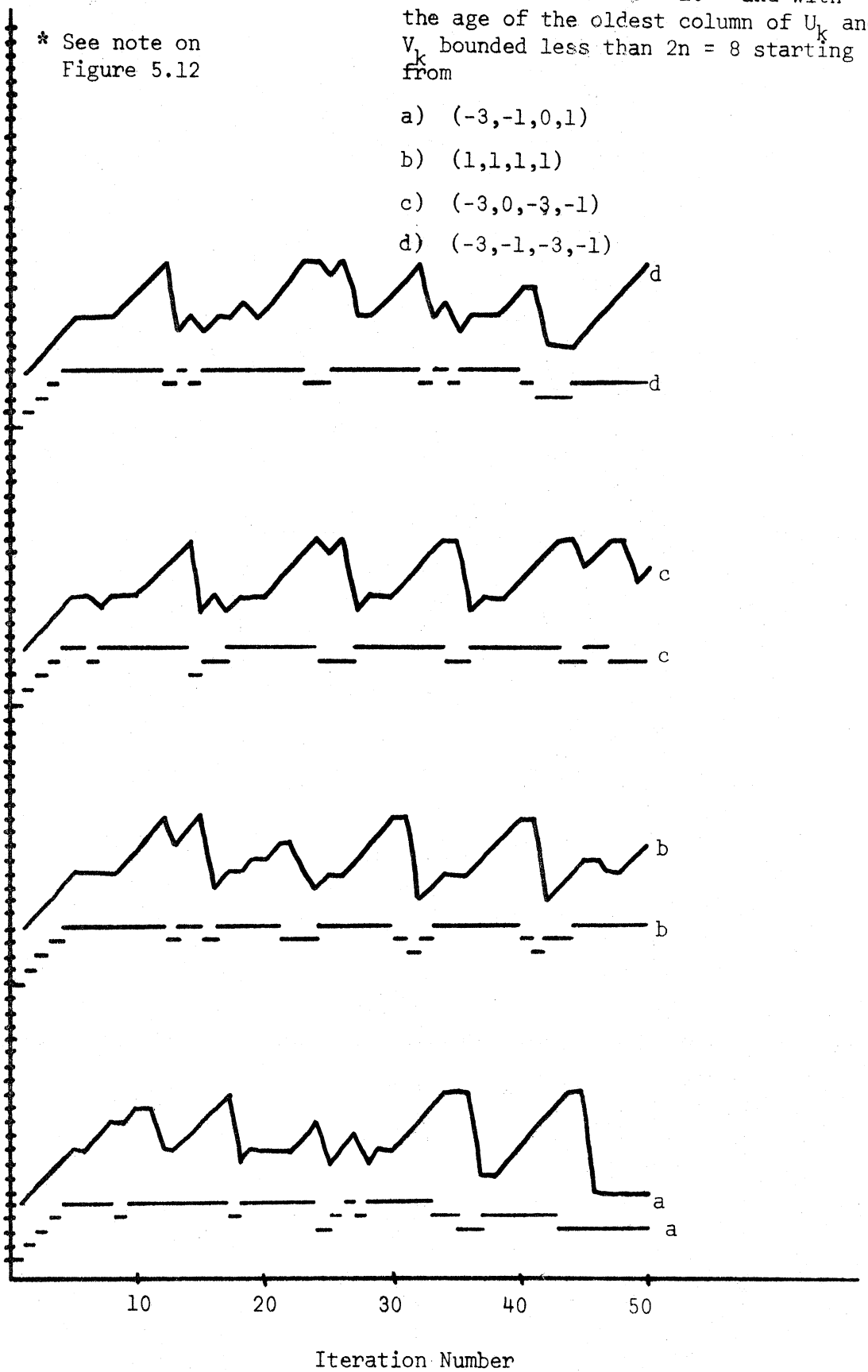
Algorithm 5.3.1 applied to Powell's function with $\alpha = \beta = 10^{-8}$ and with the age of the oldest column of U_k and V_k bounded less than $2n = 8$ starting from

a) (-3,-1,0,1)

b) (1,1,1,1)

c) (-3,0,-3,-1)

d) (-3,-1,-3,-1)



function either in Davidon's algorithm or in algorithm 5.3.1. Although this subalgorithm did not become stuck it did require a large number of function (gradient) evaluations to obtain a function decrease. A different linear search procedure would improve the performance of both algorithms when the Hessian is very poorly conditioned. The performances of algorithm 5.3.1 and Davidon's algorithm were comparable in terms of function (gradient) evaluations on Powell's function using search algorithm 5.2.1.

5.7 A Six Dimensional Function

The final test function was a six dimensional function constructed from Wood's four dimensional function and the two dimensional Rosenbrock function. This function is given by

$$f(x) = (g^2(x) + ch^2(x))^{1/2} \tag{5.7.1}$$

where

$$g(x) = 100(x_2 - x_1)^2 + (1 - x_1)^2 \tag{5.7.2}$$

and

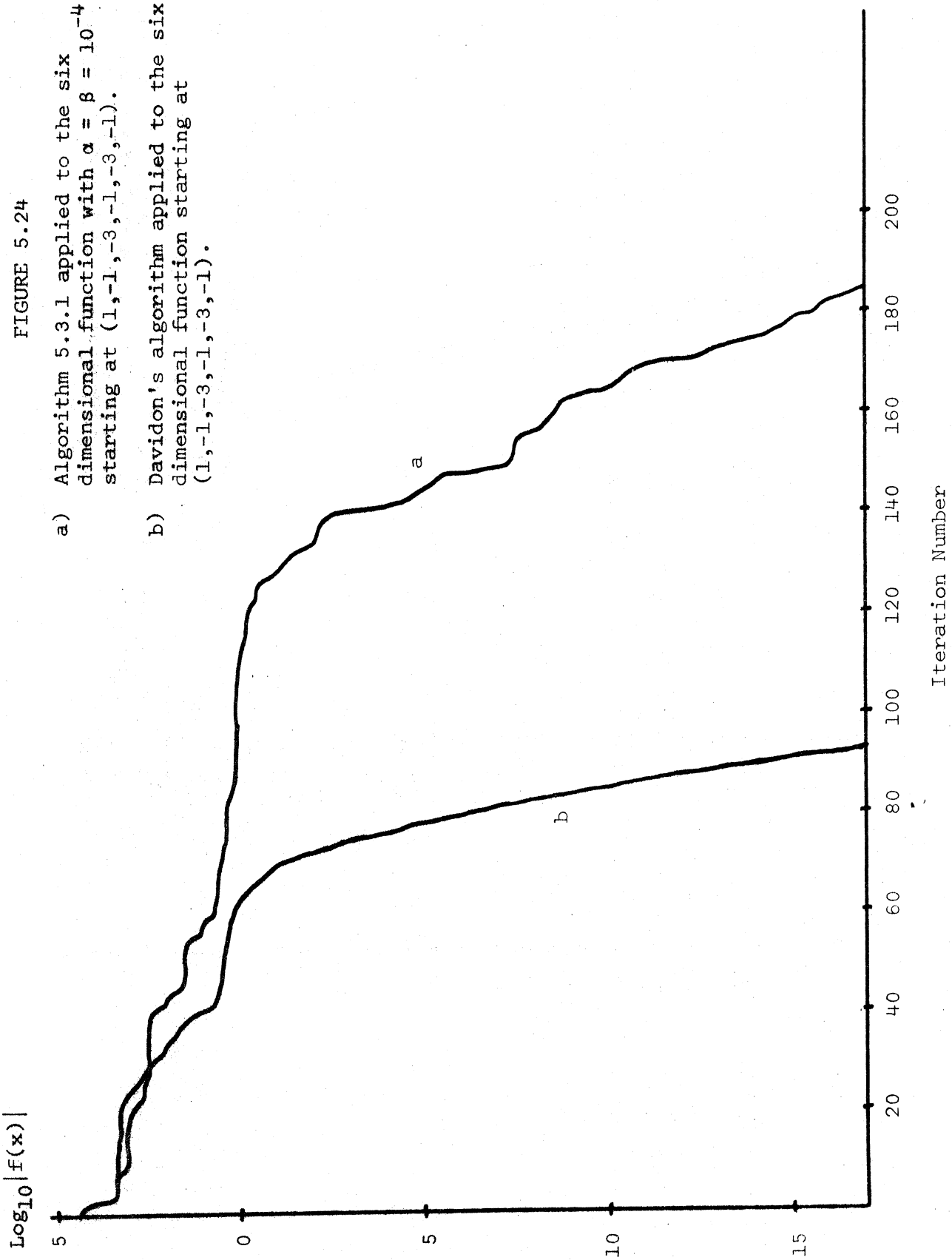
$$\begin{aligned} h(x) = {} & 100(x_3 - x_4)^2 + (1 - x_3)^2 + 90(x_6 - x_5^2)^2 \\ & + (1 - x_5)^2 + 10.1(x_4 - 1)^2 + 10.1\,(x_6 - 1)^2 \\ & + 19.9(x_4 - 1)^2(x_6 - 1)^2. \end{aligned} \tag{5.7.3}$$

The value C = 793 was chosen so that at the point (1,-1,-3,-1,-3,-1) $g^2(x) = ch^2(x)$. Algorithm 5.3.1 and the Davidon algorithm were run from the three starting points (0,0,0,0,0,0), (.9,.9,.9,.9,.9,.9) and (1,-1,-3,-1,-3,-1). In all three runs of algorithm 5.3.1, α and β were chosen to be 10^{-4}. Figures 5.26, 5.25, and 5.26 depict $\text{Log}_{10}|f(x)|$ versus iteration number for Davidon's algorithm and for algorithm 5.3.1 starting from these three starting points. In one case Davidon's

FIGURE 5.24

a) Algorithm 5.3.1 applied to the six dimensional function with $\alpha = \beta = 10^{-4}$ starting at (1,-1,-3,-1,-3,-1).

b) Davidon's algorithm applied to the six dimensional function starting at (1,-1,-3,-1,-3,-1).



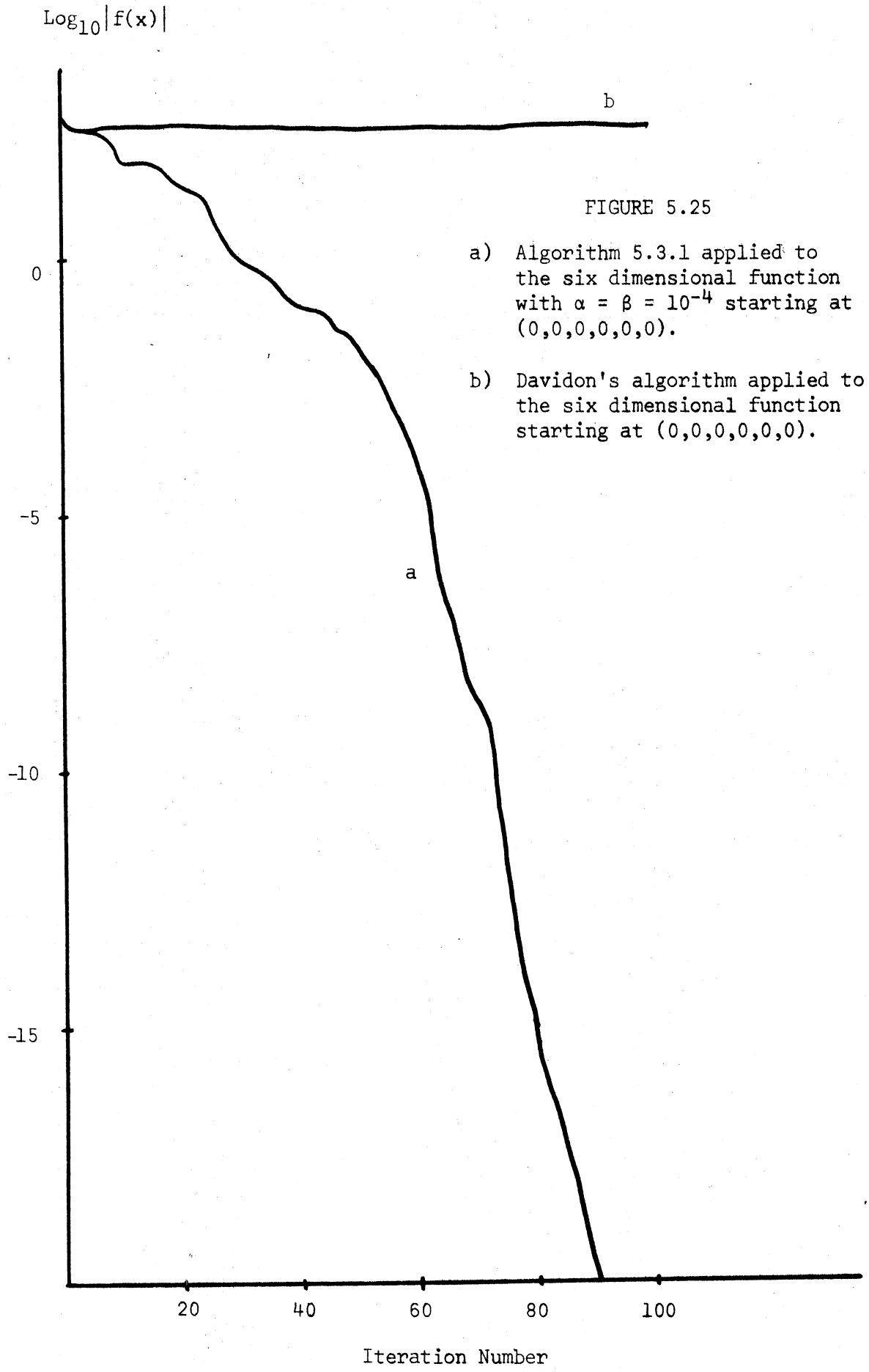


FIGURE 5.25

a) Algorithm 5.3.1 applied to the six dimensional function with $\alpha = \beta = 10^{-4}$ starting at (0,0,0,0,0,0).

b) Davidon's algorithm applied to the six dimensional function starting at (0,0,0,0,0,0).

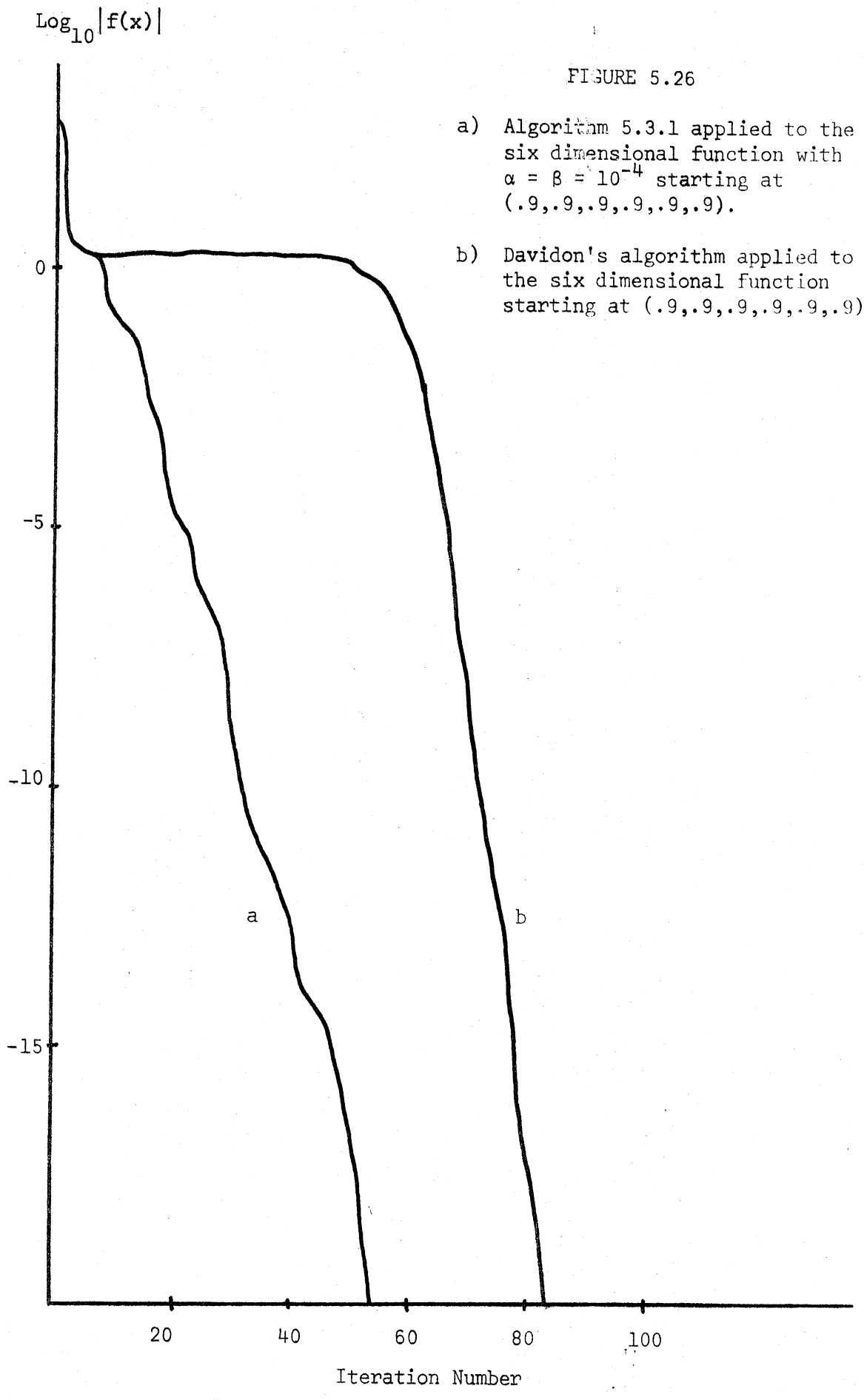


FIGURE 5.26

a) Algorithm 5.3.1 applied to the six dimensional function with $\alpha = \beta = 10^{-4}$ starting at (.9,.9,.9,.9,.9,.9).

b) Davidon's algorithm applied to the six dimensional function starting at (.9,.9,.9,.9,.9,.9).

algorithm becomes stuck while algorithm 5.3.1 converges in less than 100 iterations. In two of the three cases algorithm 5.3.1 performs better than Davidon's in terms of iterations required to reduce $f(x)$ to 10^{-20}. If the measure of performance used is the total number of function (gradient) evaluations, algorithm 5.3.1 performs significantly better than Davidon's on the six dimensional function from all these starting points.

5.8 Summary

The numerical experiments verified the convergence and rate of convergence results obtained in Chapter 4. On the two dimensional Rosenbrock function both new algorithms performed adequately although not quite as well as Davidon.

In testing the new algorithm on Wood's function it was discovered that algorithm 5.3.2 had a strong tendency to take repeated gradient steps and therefore converged slowly. As a result of this problem, algorithm 5.3.2 was not considered further. On Wood's function, algorithm 5.3.1 demonstrated that if the coefficients α and β are chosen sufficiently small, convergence is reliable and rapid. In terms of total number of function (gradient) evaluations algorithm 5.3.1 performed slightly better than Davidon's algorithm on Wood's function.

The four dimensional Powell function with a singular Hessian caused both Davidon's algorithm and algorithm 5.3.1 to converge linearly rather than superlinearly. It was found to be necessary to impose an explicit age bound on the columns of U_k to obtain good performance from algorithm 5.3.1. On Powell's function, Davidon's algorithm and algorithm 5.3.1 performed equally well.

On the six dimensional function, algorithm 5.3.1 demonstrated that

it was less prone to becoming stuck than Davidon's and further it performed better in nearly all cases. Overall, the numerical experiments do not show a sharp distinction between the performance of the new algorithm and Davidon's. However, for higher dimensional functions there seems to be a tendency for algorithm 5.3.1 to perform better. This tendency may be accentuated on functions with dimension greater than six. To date such functions have not been tested.

It is perhaps unfortunate that algorithms 5.3.1 and 5.3.2 were constructed using the linear search algorithm 5.2.1. Since the theoretical results of Chapter 4 do not require an exact linear search and since a unit step size is indicated when the search direction is p_k^* it seems reasonable to expect that a linear search algorithm tailored to the theoretical results of Chapter 4 would be more successful. The performance of the new algorithms may also be considerably improved by applying other procedures for updating the data sets and for choosing the search direction when both the p_k^* and $\bar{p}_k$ directions are unsatisfactory. The numerical experiments performed so far indicate that the new algorithms are comparable in performance to Davidon's algorithm. With futher refinement they can be expected to yield significantly superior performance.

CHAPTER 6

CONCLUDING REMARKS

The material presented in this dissertation can be divided into two major parts. In the first part, Chapters 2 and 3, a large number of established algorithms for root finding and minimization are considered in a general framework. This general framework unifies these algorithms and leads to the statement of a broad class of minimization algorithms which contains the established algorithms. Several modifications to the secant algorithms for root finding are also suggested. The research involving existing algorithms led the author to consider an entirely new class of algorithms for function minimization. The second part of the dissertation, Chapters 4 and 5, deals with this new class of algorithms. Theoretical convergence and rate of convergence properties are proved for the new algorithms and numerical experiments were conducted to verify these results and obtain further information concerning the performance of the algorithms.

The material presented in Chapter 2 deals with the problem of determining a linear transformation which approximates the relationship defined by a data set. Of particular significance is the material of section 2.4 concerning the characterization of the approximations to a data set when an approximation to a second data set, having elements in common with the first, is known. These results are important since they simplify the calculation of a sequence of approximations to a sequence of data sets when the data sets are not disjoint. One result in particular, theorem 2.4.5, yields a general formula for an emulation of a data set, when one exists, in terms of an emulation of a second data set when the data sets have common elements. This result is entirely new. It is shown in

Chapter 3 that the recursion formulas used in most of the best known minimization algorithms, along with some new recursion formulas, are a special case of the general formula of theorem 2.4.5. Although the concept of an emulation or approximation of a data set is not fundamentally new, the organization of these ideas in terms of a sequence of data sets, the notion of approximations using different norms and the restriction of the approximations to a certain set of linear transformations as presented in sections 2.1 through 2.3 is original.

Chapter 3 relates the material of Chapter 2 to the secant root finding algorithms and to a class of minimization algorithms which includes Davidon's algorithm. In section 3.3 it is suggested that generalized secant algorithms could be investigated using the pseudo inverse of the matrix U_k when the inverse does not exist or when a root of $F(x) : R^n \to R^m$, $n \neq m$, is sought. In considering the minimization algorithms in section 3.4 attention is restricted to quadratic functions. A class of minimization algorithms is presented in theorem 3.4.7 the members of which generate conjugate directions, form the inverse Hessian matrix and determine the minimum of a quadratic function in n or fewer steps. This class contains established algorithms such as Davidon's algorithm but is more general in four respects than these algorithms. First, new recursion formulas are admitted. Second, search directions, other than $-A_k g_k$ are allowed. Third, the conditions on A_k required to obtain convergence in n steps on a quadratic surface are relaxed. Fourth, it is recognized that different recursion formulas may be used at each stage. In section 3.5, new recursion formulas are developed for a sequence of approximations to a sequence of data sets generated by appending a single element to the data set at each stage. Such recursions may be of value in developing new algorithms

for root finding and minimization or in other problems where data sets occur which contain a large number of elements relative to n.

In Chapter 4 a new general minimization algorithm is presented. This algorithm, algorithm 4.3.2, is constructed so as to allow latitude in the choice of a linear search direction, a linear search procedure and the elements in the data set at each stage. In the algorithm provision is made for taking auxiliary steps to insure that a data set with the necessary properties can be chosen at each stage. It is shown in theorem 4.4.10 that if $f(x) : D \subset R^n \to R^1$ is continuously differentiable on $\underline{D}$ and if $L(f(x_0))$ is compact then algorithm 4.3.2 converges to the set of critical points of $f(x)$. It is further shown in theorem 4.4.12 that if algorithm 4.3.4 converges to a critical point $\hat{x}$, if $f(x)$ is twice differentiable at $\hat{x}$ and if the second derivative $H(x)$ is positive definite and strong at $\hat{x}$ the rate of convergence is superlinear. These results are considerably stronger than results obtained by other researchers for established minimization algorithms. For instance, Powell (POW 71) has shown that Davidon's algorithm converges superlinearly if $f(x) : D \subset R^n \to R^1$ is twice differentiable on $\underline{D}$ and if the eigenvalues of $H(x)$ are all greater than some positive constant for all $x \in \underline{D}$.

The numerical experiments in Chapter 5 verify that specific algorithms can be formulated from algorithm 4.3.2 which do not involve significantly more computation than established algorithms and which perform as well as the established algorithms. The author believes that with further refinement these new algorithms will offer distinct advantages over existing algorithms.

APPENDIX I

DERIVATIVES OF TRACE FUNCTIONS

In this section formulas are derived for the derivatives of the trace of a products of matricies. If $f(A) : R^{mn} \to R^1$ is a differentiable function of the m x n matrix A by the notation $\frac{d}{dA} f(A)$ we will mean the m x n matrix which has as its ij th component $\frac{\partial}{\partial a_{ij}} f(A)$ where a_{ij} is the element in the i th row and j th column of the matrix A. The element in the i th row and j th column of A', the transpose of A, will be denoted by a'_{ij} and the element in the i th row and j th column of a product of matricies such as AB'C will be denoted by $(AB'C)_{ij}$. By the notation $a_{i*}\, c_{*j}$ we mean the summation $\sum_k a_{ik}\, b_{kj}$ where a_{ik} and b_{ki} are components of the matricies A and B. The notation $a_{i*}\, b_{**}\, c_{*j}$ will denote the summation $\sum_l \sum_k a_{ik}\, b_{kl}\, c_{lj}$ where a_{ik}, b_{kl} and c_{lj} are components of matricies A, B and C. Other summations are similarly defined. Since the trace is only sensible for square matricies it will be assumed that all matrix products are defined and that the matrix product yields a square matrix.

1. $\frac{d}{dA}$ trace (BAC) = B'C'

Proof:

$(BAC)_{ij} = b_{i*}\, a_{**}\, c_{*j}$

trace (BAC) = $\sum_i b_{i*}\, a_{**}\, c_{*i}$

$\frac{\partial}{\partial a_{kl}}$ trace (BAC) = $\frac{\partial}{\partial a_{kl}} \sum_i b_{i*}\, a_{**}\, c_{*i}$

$$= \sum_i b_{ik}\, c_{li}$$

$$= \sum_i b'_{ki}\, c'_{il}$$

Therefore $\frac{d}{dA}$ trace (BAC) = B'C'.

2. $\frac{d}{dA}$ trace (BA'C) = CB

Proof:

$$\text{trace (BA'C)} = \sum_i b_{i*}\, a_{**}\, c_{*i}$$

$$\frac{\partial}{\partial a_{kl}} \text{trace (BA'C)} = \frac{\partial}{\partial a_{kl}} \sum_i b_{i*}\, a_{**}\, c_{*i}$$

$$= \sum_i b_{il}\, c_{ki}$$

$$= \sum_i b'_{li}\, c'_{ik}$$

Therefore $\frac{d}{dA}$ trace (BA'C) = (B'C')' = BC.

3. $\frac{d}{dA}$ trace (BADAC) = (CBAD + DACB)'

Proof:

$$\frac{\partial}{\partial a_{kl}} \text{trace (BADAC)} = \frac{\partial}{\partial a_{kl}} \sum_i b_{i*}\, a_{**}\, d_{**}\, a_{**}\, c_{*i}$$

$$= \sum_i b_{i*}\, a_{**}\, d_{*k}\, c_{li}$$

$$+ b_{ik}\, d_{l*}\, a_{**}\, c_{*i}$$

$$= \sum_i (BAD)_{ik} c_{li} + b_{ik} (DAC)_{li}$$

$$= ((BAD)'C')_{kl} + (B'(DAC)')_{kl}$$

Therefore $\frac{d}{dA}$ trace (BADAC) = (CBAD + DACB)'.

4. $\frac{d}{dA}$ trace (BA'DA'C) = CBA'D + DA'CB

Proof:

$$\frac{\partial}{\partial a_{kl}} \text{trace } (BA'DA'C) = \frac{\partial}{\partial a_{kl}} \sum_i b_{i*} a_{**} d_{**} a_{**} c_{*i}$$

$$= \sum_i b_{i*} a_{**} d_{*l} c_{ki}$$

$$+ b_{il} d_{k*} a_{**} c_{*i}$$

$$= \sum_i (BA'D)_{il} c_{ki} + b_{il} (DA'C)_{ki}$$

$$= ((BA'D)'C')_{lk} + (B'(DA'C)')_{lk}$$

$$= (CBA'D)_{kl} + (DA'CB)_{kl}$$

Therefore $\frac{d}{dA}$ trace (BA'DA'C) = CBA'D + DA'CB.

5. $\frac{d}{dA}$ trace (BA'DAC) = (CBA'D)' + DACB

Proof:

$$\frac{\partial}{\partial a_{kl}} \text{trace } (BA'DAC) = \frac{\partial}{\partial a_{kl}} \sum_i b_{i*} a_{**} d_{**} a_{**} c_{*i}$$

$$= \sum_i b_{i*} a_{**} d_{*k} c_{li}$$

$$+ b_{il} d_{k*} a_{**} c_{*i}$$

$$= \sum_i (BA'D)_{ik} c_{li} + b_{il} (DAC)_{ki}$$

$$+ ((BA'D)'C')_{kl} + (DACB)_{kl}$$

Therefore $\frac{d}{dA}$ trace $(BA'DAC) = (CBA'D)' + DACB$.

6. $\frac{d}{dA}$ trace $(BADA'C) = CBAD + (DA'CB)'$

Proof:

$$\frac{\partial}{\partial a_{kl}} \text{ trace } (BADA'C) = \frac{\partial}{\partial a_{kl}} \sum_i b_{i*} a_{**} d_{**} a_{**} c_{*i}$$

$$= \sum_i b_{i*} a_{**} d_{*l} c_{ki}$$

$$+ b_{ik} d_{l*} a_{**} c_{*i}$$

$$= \sum_i (BAD)_{il} c_{ki} + b_{ik} (DA'C)_{li}$$

$$= (CBAD)_{kl} + (DA'CB)'_{kl}$$

Therefore $\frac{d}{dA}$ trace $(BADA'C) = CBAD + (DA'CB)'$.

APPENDIX II

FORTRAN PROGRAM DAVIDON'S ALGORITHM


```
C
C     ..................................................................
C
C        SUBROUTINE DFMFP
C
C        PURPOSE
C           TO FIND A LOCAL MINIMUM OF A FUNCTION OF SEVERAL VARIABLES
C           BY THE METHOD OF FLETCHER AND POWELL
C
C        USAGE
C           CALL DFMFP(FUNCT,N,X,F,G,EST,EPS,LIMIT,IER,H)
C
C        DESCRIPTION OF PARAMETERS
C           FUNCT  - USER-WRITTEN SUBROUTINE CONCERNING THE FUNCTION TO
C                    BE MINIMIZED. IT MUST BE OF THE FORM
C                    SUBROUTINE FUNCT(N,ARG,VAL,GRAD)
C                    AND MUST SERVE THE FOLLOWING PURPOSE
C                    FOR EACH N-DIMENSIONAL ARGUMENT VECTOR  ARG,
C                    FUNCTION VALUE AND GRADIENT VECTOR MUST BE COMPUTED
C                    AND, ON RETURN, STORED IN VAL AND GRAD RESPECTIVELY
C                    ARG,VAL AND GRAD MUST BE OF DOUBLE PRECISION.
C           N      - NUMBER OF VARIABLES
C           X      - VECTOR OF DIMENSION N CONTAINING THE INITIAL
C                    ARGUMENT WHERE THE ITERATION STARTS. ON RETURN,
C                    X HOLDS THE ARGUMENT CORRESPONDING TO THE
C                    COMPUTED MINIMUM FUNCTION VALUE
C                    DOUBLE PRECISION VECTOR.
C           F      - SINGLE VARIABLE CONTAINING THE MINIMUM FUNCTION
C                    VALUE ON RETURN, I.E. F=F(X).
C                    DOUBLE PRECISION VARIABLE.
C           G      - VECTOR OF DIMENSION N CONTAINING THE GRADIENT
C                    VECTOR CORRESPONDING TO THE MINIMUM ON RETURN,
C                    I.E. G=G(X).
C                    DOUBLE PRECISION VECTOR.
C           EST    - IS AN ESTIMATE OF THE MINIMUM FUNCTION VALUE.
C                    SINGLE PRECISION VARIABLE.
C           EPS    - TESTVALUE REPRESENTING THE EXPECTED ABSOLUTE ERROR.
C                    A REASONABLE CHOICE IS 10**(-16), I.E.
C                    SOMEWHAT GREATER THAN 10**(-D), WHERE D IS THE
C                    NUMBER OF SIGNIFICANT DIGITS IN FLOATING POINT
C                    REPRESENTATION.
C                    SINGLE PRECISION VARIABLE.
C           LIMIT  - MAXIMUM NUMBER OF ITERATIONS.
C           IER    - ERROR PARAMETER
C                    IER = 0 MEANS CONVERGENCE WAS OBTAINED
C                    IER = 1 MEANS NO CONVERGENCE IN LIMIT ITERATIONS
C                    IER =-1 MEANS ERRORS IN GRADIENT CALCULATION
C                    IER = 2 MEANS LINEAR SEARCH TECHNIQUE INDICATES
C                    IT IS LIKELY THAT THERE EXISTS NO MINIMUM.
C           H      - WORKING STORAGE OF DIMENSION N*(N+7)/2.
C                    DOUBLE PRECISION ARRAY.
C
C        REMARKS
C           I) THE SUBROUTINE NAME REPLACING THE DUMMY ARGUMENT  FUNCT
C              MUST BE DECLARED AS EXTERNAL IN THE CALLING PROGRAM.
C          II) IER IS SET TO 2 IF , STEPPING IN ONE OF THE COMPUTED
```

```
C               DIRECTIONS, THE FUNCTION WILL NEVER INCREASE WITHIN
C               A TOLERABLE RANGE OF ARGUMENT.
C               IER = 2 MAY OCCUR ALSO IF THE INTERVAL WHERE F
C               INCREASES IS SMALL AND THE INITIAL ARGUMENT WAS
C               RELATIVELY FAR AWAY FROM THE MINIMUM SUCH THAT THE
C               MINIMUM WAS OVERLEAPED. THIS IS DUE TO THE SEARCH
C               TECHNIQUE WHICH DOUBLES THE STEPSIZE UNTIL A POINT
C               IS FOUND WHERE THE FUNCTION INCREASES.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           FUNCT
C
C        METHOD
C           THE METHOD IS DESCRIBED IN THE FOLLOWING ARTICLE
C           R. FLETCHER AND M.J.D. POWELL, A RAPID DESCENT METHOD FOR
C           MINIMIZATION,
C           COMPUTER JOURNAL VOL.6, ISS. 2, 1963, PP.163-168.
C
C     ..................................................................
C
      SUBROUTINE DFMFP(FUNCT,N,X,F,G,EST,EPS,LIMIT,IER,H)
C
C        DIMENSIONED DUMMY VARIABLES
      COMMON/ANS/XSTAR,FSTAR,ICNT
      DIMENSION H(1),X(1),G(1)
     A  ,XSTAR(10)
      DOUBLE PRECISION X,F,FX,FY,OLDF,HNRM,GNRM,H,G,DX,DY,ALFA,DALFA,
     1AMBDA,T,Z,W
     A                ,XMAG,XLOG,XSTAR,GMAG,GLOG,FMAG,FLOG,FSTAR
C
C        COMPUTE FUNCTION VALUE AND GRADIENT VECTOR FOR INITIAL ARGUMENT
      CALL FUNCT(N,X,F,G)
C
C        RESET ITERATION COUNTER AND GENERATE IDENTITY MATRIX
      ICNT=0
      IER=0
      KOUNT=0
      N2=N+N
      N3=N2+N
      N31=N3+1
    1 K=N31
      WRITE(6,500)
  500 FORMAT('RESET')
      ICNT=ICNT+1
      DO 4 J=1,N
      H(K)=1.D0
      NJ=N-J
      IF(NJ)5,5,2
    2 DO 3 L=1,NJ
      KL=K+L
    3 H(KL)=0.D0
    4 K=KL+1
C
C        START ITERATION LOOP
    5 KOUNT=KOUNT +1
      GMAG=0.0D0
```

```
      XMAG=0.D0
      DO120 I=1,N
      XMAG=XMAG+(X(I)-XSTAR(I))**2
  120 GMAG=GMAG+G(I)**2
      XMAG=DSQRT(XMAG)
      GMAG=DSQRT(GMAG)
      FMAG=F-FSTAR
      IF(XMAG.GT.0.D0.AND.GMAG.GT.0.D0.AND.FMAG.GT.0.D0)GO TO 121
      WRITE (6,104) KOUNT,XMAG,GMAG,FMAG
  104 FORMAT (///'L=',I3,' ^X^=',D12.5,' ^G^=',D12.5,' ^F^=',D12.5/)
      GO TO 122
  121 XLOG=DLOG10(XMAG)
      GLOG=DLOG10(GMAG)
      FLOG=DLOG10(FMAG)
      WRITE (6,101) KOUNT,XMAG,XLOG,GMAG,GLOG,FMAG,FLOG
  101 FORMAT(///'L=',I3,' ^X^=',D12.5,' LOG^X^=',D12.5,' ^G^=',D12.5,
     1    ' LOG^G^=',
     A   D12.5,' ^F^=',D12.5,' LOG^F^=',D12.5/)
  122 WRITE (6,102) (X(I),I=1,N)
  102 FORMAT ('X=',10D12.4)
      WRITE (6,103) (G(I),I=1,N)
  103 FORMAT ('G=',10D12.4)
      CC=KOUNT
C
C        SAVE FUNCTION VALUE, ARGUMENT VECTOR AND GRADIENT VECTOR
      OLDF=F
      DO 9 J=1,N
      K=N+J
      H(K)=G(J)
      K=K+N
      H(K)=X(J)
C
C        DETERMINE DIRECTION VECTOR H
      K=J+N3
      T=0.D0
      DO 8 L=1,N
      T=T-G(L)*H(K)
      IF(L-J)6,7,7
    6 K=K+N-L
      GO TO 8
    7 K=K+1
    8 CONTINUE
    9 H(J)=T
C
C        CHECK WHETHER FUNCTION WILL DECREASE STEPPING ALONG H.
      DY=0.D0
      HNRM=0.D0
      GNRM=0.D0
C
C        CALCULATE DIRECTIONAL DERIVATIVE AND TESTVALUES FOR DIRECTION
C        VECTOR H AND GRADIENT VECTOR G.
      DO 10 J=1,N
      HNRM=HNRM+DABS(H(J))
      GNRM=GNRM+DABS(G(J))
   10 DY=DY+H(J)*G(J)
C
```

```
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DIRECTIONAL
C        DERIVATIVE APPEARS TO BE POSITIVE OR ZERO.
      IF(DY)11,51,51
C
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DIRECTION
C        VECTOR H IS SMALL COMPARED TO GRADIENT VECTOR G.
   11 IF(HNRM/GNRM-EPS)51,51,12
C
C        SEARCH MINIMUM ALONG DIRECTION H
C
C        SEARCH ALONG H FOR POSITIVE DIRECTIONAL DERIVATIVE
   12 FY=F
      ALFA=2.D0*(EST-F)/DY
      AMBDA=1.D0
C
C        USE ESTIMATE FOR STEPSIZE ONLY IF IT IS POSITIVE AND LESS THAN
C        1. OTHERWISE TAKE 1. AS STEPSIZE
      IF(ALFA)15,15,13
   13 IF(ALFA-AMBDA)14,15,15
   14 AMBDA=ALFA
   15 ALFA=0.D0
C
C        SAVE FUNCTION AND DERIVATIVE VALUES FOR OLD ARGUMENT
   16 FX=FY
      DX=DY
C
C        STEP ARGUMENT ALONG H
      DO 17 I=1,N
   17 X(I)=X(I)+AMBDA*H(I)
C
C        COMPUTE FUNCTION VALUE AND GRADIENT FOR NEW ARGUMENT
      CALL FUNCT(N,X,F,G)
      FY=F
C
C        COMPUTE DIRECTIONAL DERIVATIVE DY FOR NEW ARGUMENT.  TERMINATE
C        SEARCH, IF DY IS POSITIVE. IF DY IS ZERO THE MINIMUM IS FOUND
      DY=0.D0
      DO 18 I=1,N
   18 DY=DY+G(I)*H(I)
      IF(DY)19,36,22
C
C        TERMINATE SEARCH ALSO IF THE FUNCTION VALUE INDICATES THAT
C        A MINIMUM HAS BEEN PASSED
   19 IF(FY-FX)20,22,22
C
C        REPEAT SEARCH AND DOUBLE STEPSIZE FOR FURTHER SEARCHES
   20 AMBDA=AMBDA+ALFA
      ALFA=AMBDA
C        END OF SEARCH LOOP
C
C        TERMINATE IF THE CHANGE IN ARGUMENT GETS VERY LARGE
      IF(HNRM*AMBDA-1.D10)16,16,21
C
C        LINEAR SEARCH TECHNIQUE INDICATES THAT NO MINIMUM EXISTS
   21 IER=2
      RETURN
```

```
C
C         INTERPOLATE CUBICALLY IN THE INTERVAL DEFINED BY THE SEARCH
C         ABOVE AND COMPUTE THE ARGUMENT X FOR WHICH THE INTERPOLATION
C         POLYNOMIAL IS MINIMIZED
   22 T=0.D0
   23 IF(AMBDA)24,36,24
   24 Z=3.D0*(FX-FY)/AMBDA+DX+DY
      ALFA=DMAX1(DABS(Z),DABS(DX),DABS(DY))
      DALFA=Z/ALFA
      DALFA=DALFA*DALFA-DX/ALFA*DY/ALFA
      IF(DALFA)51,25,25
   25 W=ALFA*DSQRT(DALFA)
      ALFA=DY-DX+W+W
      IF (ALFA) 250,251,250
  250 ALFA=(DY-Z+W)/ALFA
      GO TO 252
  251 ALFA=(Z+DY-W)/(Z+DX+Z+DY)
  252 ALFA=ALFA*AMBDA
      DO 26 I=1,N
   26 X(I)=X(I)+(T-ALFA)*H(I)
C
C         TERMINATE, IF THE VALUE OF THE ACTUAL FUNCTION AT X IS LESS
C         THAN THE FUNCTION VALUES AT THE INTERVAL ENDS. OTHERWISE REDUCE
C         THE INTERVAL BY CHOOSING ONE END-POINT EQUAL TO X AND REPEAT
C         THE INTERPOLATION.  WHICH END-POINT IS CHOOSEN DEPENDS ON THE
C         VALUE OF THE FUNCTION AND ITS GRADIENT AT X
C
      CALL FUNCT(N,X,F,G)
      IF(F-FX)27,27,28
   27 IF(F-FY)36,36,28
   28 DALFA=0.D0
      DO 29 I=1,N
   29 DALFA=DALFA+G(I)*H(I)
      IF(DALFA)30,33,33
   30 IF(F-FX)32,31,33
   31 IF(DX-DALFA)32,36,32
   32 FX=F
      DX=DALFA
      T=ALFA
      AMBDA=ALFA
      GO TO 23
   33 IF(FY-F)35,34,35
   34 IF(DY-DALFA)35,36,35
   35 FY=F
      DY=DALFA
      AMBDA=AMBDA-ALFA
      GO TO 22
C
C     TERMINATE, IF FUNCTION HAS NOT DECREASED DURING LAST ITERATION
   36 IF (OLDF-F+EPS) 51,38,38
C
C     COMPUTE DIFFERENCE VECTORS OF ARGUMENT AND GRADIENT FROM
C         TWO CONSECUTIVE ITERATIONS
   38 DO 37 J=1,N
      K=N+J
      H(K)=G(J)-H(K)
```

```
      K=N+K
   37 H(K)=X(J)-H(K)
C
C        TEST LENGTH OF ARGUMENT DIFFERENCE VECTOR AND DIRECTION VECTOR
C        IF AT LEAST N ITERATIONS HAVE BEEN EXECUTED, TERMINATE, IF
C        BOTH ARE LESS THAN  EPS
      IER=0
      IF(KOUNT-N)42,39,39
   39 T=0.D0
      Z=0.D0
      DO 40 J=1,N
      K=N+J
      W=H(K)
      K=K+N
      T=T+DABS(H(K))
   40 Z=Z+W*H(K)
      IF(HNRM-EPS)41,41,42
   41 IF(T-EPS)56,56,42
C
C        TERMINATE, IF NUMBER OF ITERATIONS WOULD EXCEED  LIMIT
   42 IF(KOUNT-LIMIT)43,50,50
C
C        PREPARE UPDATING OF MATRIX H
   43 ALFA=0.D0
      DO 47 J=1,N
      K=J+N3
      W=0.D0
      DO 46 L=1,N
      KL=N+L
      W=W+H(KL)*H(K)
      IF(L-J)44,45,45
   44 K=K+N-L
      GO TO 46
   45 K=K+1
   46 CONTINUE
      K=N+J
      ALFA=ALFA+W*H(K)
   47 H(J)=W
C
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF RESULTS
C        ARE NOT SATISFACTORY
      IF(Z*ALFA)48,1,48
C
C        UPDATE MATRIX H
   48 K=N31
      DO 49 L=1,N
      KL=N2+L
      DO 49 J=L,N
      NJ=N2+J
      H(K)=H(K)+H(KL)*H(NJ)/Z-H(L)*H(J)/ALFA
   49 K=K+1
      GO TO 5
C        END OF ITERATION LOOP
C
C        NO CONVERGENCE AFTER  LIMIT  ITERATIONS
   50 IER=1
```

```
      RETURN
C
C        RESTORE OLD VALUES OF FUNCTION AND ARGUMENTS
   51 DO 52 J=1,N
      K=N2+J
   52 X(J)=H(K)
      CALL FUNCT(N,X,F,G)
C
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DERIVATIVE
C        FAILS TO BE SUFFICIENTLY SMALL
      IF(GNRM-EPS)55,55,53
C
C        TEST FOR REPEATED FAILURE OF ITERATION
   53 IF(IER)56,54,54
   54 IER=-1
      GOTO 1
   55 IER=0
   56 RETURN
```

APPENDIX III

FORTRAN PROGRAM FOR ALGORITHM 5.3.1


```
      SUBROUTINE DIXON(FUNCT,N,X,F,G,EST,EPS,LIMIT)
C
CI.      DECLARATIONS
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ANS/XSTAR(10),FSTAR
      COMMON/PIN1/U(10,10),UD(10,10),UDM(10,10),U1(10),UT(10),UV(10),UM(
      DIMENSION V(10,10),VV(10),        P(10),X(10),XOLD(10),G(10),
     A   GOLD(10),EPS(4),IAGE(10)
C
CII.     CHECK DIMENSION OF PROBLEM
C
      IF (N .LE. 10) GO TO 5
      WRITE (6,100) N
 100  FORMAT ('-N=',I3,3X,'IS TOO LARGE FOR SUBROUTINE; CHANGE DIMENSION
     ANT'//)
      STOP
C
CIII.    INITIALIZE VARIABLES
C
 5    IRK=0
      L=0
      ISW2=0
      ISW3=0
      DO 10 I=1,N
      IAGE(I)=0
 10   UD(1,I)=0.000
      CALL FUNCT(N,X,F,G)
C
CIV.     STARTING ITERATION NUMBER "L"
C
   15 IF(IAGE(1).EQ.0)GO TO 16
      IF(L-IAGE(1)-2*N)16,17,17
   17 CALL PINV2(N,IRK,1)
      IRK=IRK-1
      DO 18 J=1,IRK
      DO 19 I=1,N
      U(I,J)=U(I,J+1)
      V(I,J)=V(I,J+1)
   19 UD(J,I)=UDM(J,I)
   18 IAGE(J)=IAGE(J+1)
      IAGE(IRK+1)=0
   16 L=L+1
      IF (L .GT. LIMIT) RETURN
      IRK1=IRK+1
      GMAG=0.D0
      XMAG=0.D0
      DO 20 I=1,N
      XMAG=XMAG+(X(I)-XSTAR(I))**2
20    GMAG=GMAG+G(I)**2
      XMAG=DSQRT(XMAG)
      GMAG=DSQRT(GMAG)
      FMAG=F-FSTAR
      IF(XMAG.GT.0.D0.AND.GMAG.GT.0.D0.AND.FMAG.GT.0.D0)GO TO 121
      WRITE (6,105) L,XMAG,GMAG,FMAG
105   FORMAT (///'L=',I3,' ^X^=',D12.5,' ^G^=',D12.5,' ^F^=',D12.5/)
```

```
      GO TO 122
  121 XLOG=DLOG10(XMAG)
      GLOG=DLOG10(GMAG)
      FLOG=DLOG10(FMAG)
      WRITE (6,101) L,XMAG,XLOG,GMAG,GLOG,FMAG,FLOG
  101 FORMAT(///'L=',I3,' ^X^=',D12.5,' LOG^X^=',D12.5,' ^G^=',D12.5,' L
     A   D12.5,' ^F^=',D12.5,' LOG^F^=',D12.5/)
  122 WRITE (6,102) (X(I),I=1,N)
  102 FORMAT ('X=',10D12.4)
      WRITE (6,103) (G(I),I=1,N)
  103 FORMAT ('G=',10D12.4)
      WRITE (6,104) (IAGE(I),I=1,N)
  104 FORMAT ('IAGE=',10I6)
      CC=L
C
CIV.     A.   ITERATION TERMINATED IF ^G^ < EPS4
C
   24 IF ( GMAG .GT. EPS(4)) GO TO 25
      RETURN
C
CIV.     B.   COMPUTE PSEUDO INVERSE OF NEW U FROM UD & UV
C
   25 IF (ISW2 .EQ. 0) GO TO 26
      DO 254 I=1,N
  254 UD(1,I)=0.D0
      IF (IRK .EQ. 0) GO TO 26
      DO 255 I=1,IRK
      DO 256 J=1,N
  256 UV(J)=U(J,I)
  255 CALL PINV(N,I)
C
CIV.     C.   COMPUTE PBAR
C
   26 ISW2=1
  275 PMAG=0.0
      COS=0.0D0
      DO 27 I=1,IRK
      UT(I)=0.D0
      DO 27 J=1,N
   27 UT(I)=UT(I)+UD(I,J)*G(J)
      IF (IRK .EQ. N) GO TO 30
      DO 29 I=1,N
      P(I)=G(I)
      DO 28 J=1,IRK
   28 P(I)=P(I)-U(I,J)*UT(J)
      PMAG=PMAG+P(I)**2
   29 COS=COS+G(I)*P(I)
      PMAG=DSQRT(PMAG)
      COS=COS/(GMAG*PMAG)
C
CIV.     D.   IF PBAR IS PERPENDICULAR TO G COMPUTE PSTAR
C
      IF (COS .GT. EPS(1)) GO TO 40
   30 COS=0.0D0
      PMAG=0.0D0
      DO 32 I=1,N
```

```
      P(I)=0.0D0
      DO 31 J=1,IRK
 31   P(I)=P(I)+V(I,J)*UT(J)
      PMAG=PMAG+P(I)**2
 32   COS=COS+G(I)*P(I)
      PMAG=DSQRT(PMAG)
      COS=COS/(GMAG*PMAG)
      ISW3=1
C
CIV.     E.  IF PSTAR IS PERPENDICULAR TO G; DELETE THE OLDEST COL. OF U
C
      IF (COS .GT. EPS(2)) GO TO 40
      CALL PINV2(N,IRK,1)
      L=L-1
      ISW2=0
      ISW3=0
      IRK=IRK-1
      DO 36 J=1,IRK
      DO 35 I=1,N
      U(I,J)=U(I,J+1)
      V(I,J)=V(I,J+1)
 35   UD(J,I)=UDM(J,I)
 36   IAGE(J)=IAGE(J+1)
      IAGE(IRK+1)=0
      GO TO 15
 40   DO 41 I=1,N
      GOLD(I)=G(I)
 41   XOLD(I)=X(I)
C
CIV.      F.  DETERMINE STEPLENGTH
C
 42   CALL STPLEN(FUNCT,N,X,F,G,P,EST)
C
CIV.     G.   CALCULATE DX & DG VECTORS
C
 44   UMAG=0.0D0
 45   DO 50 I=1,N
      VV(I)=X(I)-XOLD(I)
      UV(I)=G(I)-GOLD(I)
 50   UMAG=UMAG+UV(I)**2
      UMAG=DSQRT(UMAG)
C
CIV.     H.   TEST NEW DG VECTOR
C
      IF (IRK .EQ. N) GO TO 61
      DO 505 I=1,IRK
      UT(I)=0.00
      DO 505 J=1,N
 505  UT(I)=UT(I)+UD(I,J)*UV(J)
      CMAG=0.00
      DO 52 I=1,N
      COMP=0.00
      DO 51 J=1,IRK
 51   COMP=COMP+U(I,J)*UT(J)
 52   CMAG=CMAG+COMP**2
      CMAG=DSQRT(CMAG)/UMAG
```

```
      IF (CMAG .GT. (1.D0-EPS(3))) GO TO 61
C
CIV.     H1.  PASSED:  UPDATE U & V MATRICES
C
      IRK=IRK+1
      IAGE(IRK)=L
      DO 60 I=1,N
      U(I,IRK)=UV(I)
 60   V(I,IRK)=VV(I)
      GO TO 15
C
CIV.     H2.  FAILED: TRY ELIMINATING ONE OF THE COLUMNS
C
 61   DO 75 M=1,IRK
      CALL PINV2(N,IRK,M)
      CMAG=0.D0
      DO 67 II=1,N
      COMP=0.D0
      JJ=0
      DO 66 J=1,IRK
      IF (J .EQ. M) GO TO 66
      JJ=JJ+1
      COMP=COMP+U(II,J)*UT(JJ)
 66   CONTINUE
 67   CMAG=CMAG+COMP**2
      CMAG=DSQRT(CMAG)/UMAG
      IF (CMAG .GT. (1.D0-EPS(3))) GO TO 75
      IRK1=IRK-1
      DO 72 I=1,IRK1
      DO 72 J=1,N
 72   UD(I,J)=UDM(I,J)
      DO 73 J=M,IRK1
      IAGE(J)=IAGE(J+1)
      DO 73 I=1,N
      U(I,J)=U(I,J+1)
 73   V(I,J)=V(I,J+1)
      IAGE(IRK)=L
      DO 74 I=1,N
      U(I,IRK)=UV(I)
 74   V(I,IRK)=VV(I)
      GO TO 15
 75   CONTINUE
C
CIV.     H3.  TEST FAILED AGAIN U, V & UD REMAIN UNCHANGED
C
      ISW2=0
      GO TO 15
      END
```

PAGE 00001

```
      SUBROUTINE PINV(N,IRK)
C   MATRICES USED: U,UD,UDM  --  VECTORS USED: U1,UT,UV
C     U: THE DELTA-G MATRIX                                          SUPPLIED
C     UD: THE PSUEDO INVERSE OF THE "U" MATRIX   SUPPLIED,UPDATED & RETURNED
C     UDM: WORKING MATRIX
C     U1: WORKING VECTOR; U1 = (I - U * UD) * UV
C     UT: WORKING VECTOR; UT = UD * UV
C     UV: NEW COLUMN JUST ADDED TO THE "U" MATRIX
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/PIN1/U(10,10),UD(10,10),UDM(10,10),U1(10),UT(10),UV(10),UM(10)
      IF (IRK .LE. 0) RETURN
      IRK1=IRK-1
      U1MAG=0.0D0
      IF (IRK1 .EQ. 0) GO TO 211
      DO 21 I=1,IRK1
      UT(I)=0.0D0
      DO 21 J=1,N
 21   UT(I)=UT(I)+UD(I,J)*UV(J)
 211  DO 23 I=1,N
      U1(I)=UV(I)
      IF (IRK1 .EQ. 0) GO TO 23
      DO 22 J=1,IRK1
 22   U1(I)=U1(I)-U(I,J)*UT(J)
 23   U1MAG=U1MAG+U1(I)**2
      DO 25 J=1,N
 25   UD(IRK,J)=U1(J)/U1MAG
      IF (IRK1 .EQ. 0) GO TO 26
      DO 24 I=1,IRK1
      DO 24 J=1,N
 24   UDM(I,J)=UD(I,J)
      DO 241 I=1,IRK1
      DO 241 J=1,N
      UD(I,J)=UDM(I,J)
      DO 241 K=1,N
 241  UD(I,J)=UD(I,J)-UDM(I,K)*UV(K)*U1(J)/U1MAG
 26   RETURN
      END
```

```
      SUBROUTINE PINV2(N,IRK,M)
C    MATRICES USED: UD,UDM  --  VECTORS USED: UT,UV,UM
C      UD: PSUEDO INVERSE OF THE "U" MATRIX                          SUPPLIED
C      UDM: PSUEDO INVERSE OF THE U MATRIX MINUS THE MTH COL.        RETURNED
C      UT: THE PRODUCT OF "UDM" TIMES "UV"                           RETURNED
C      UV: NEW COLUMN TO BE ADDED TO THE U MATRIX
C      UM: THE MTH ROW OF THE UD MATRIX
       IMPLICIT REAL*8(A-H,O-Z)
       COMMON/PIN1/U(10,10),UD(10,10),UDM(10,10),U1(10),UT(10),UV(10),UM(10)
       UDMAG=0.D0
       DO 62 I=1,N
       UT(I)=0.D0
       UDM(1,I)=0.D0
       UM(I)=UD(M,I)
 62    UDMAG=UDMAG+UM(I)**2
       I=0
       DO 65 II=1,IRK
       IF (II .EQ. M) GO TO 65
       I=I+1
       UT(I)=0.D0
       DO 64 J=1,N
       DUM=UD(II,J)
       DO 63 K=1,N
 63    DUM=DUM-UD(II,K)*UM(J)*UM(K)/UDMAG
       UT(I)=UT(I)+DUM*UV(J)
 64    UDM(I,J)=DUM
 65    CONTINUE
       RETURN
       END
```

```
      SUBROUTINE STPLEN(FUNCT,N,X,F,G,P,EST)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(10),G(10),P(10)
      DY=0.D0
      DO 10 I=1,N
      P(I)=-P(I)
10    DY=DY+G(I)*P(I)
      IF (DY) 12,11,11
11    WRITE (6,100)
100   FORMAT ('-THE DIRECTIONAL DERIVATIVE APPREARS TO BE > OR = 0.')
      STOP
12    FY=F
      ALFA=2.D0*(EST-F)/DY
      AMBDA=1.D0
      IF (ALFA) 15,15,13
13    IF (ALFA-AMBDA) 14,15,15
14    AMBDA=ALFA
15    ALFA=0.D0
16    FX=FY
      DX=DY
      DO 17 I=1,N
17    X(I)=X(I)+AMBDA*P(I)
      CALL FUNCT(N,X,F,G)
      FY=F
      DY=0.D0
      DO 18 I=1,N
18    DY=DY+G(I)*P(I)
      IF (DY ) 19,36,22
19    IF (FY-FX) 20,22,22
20    AMBDA=AMBDA+ALFA
      ALFA=AMBDA
      GO TO 16
22    T=0.D0
23    IF (AMBDA) 24,36,24
24    Z=3.D0*(FX-FY)/AMBDA+DX+DY
      ALFA=DMAX1(DABS(Z),DABS(DX),DABS(DY))
      DALFA=Z/ALFA
      DALFA=DALFA*DALFA-DX/ALFA*DY/ALFA
      IF (DALFA) 51,25,25
51    STOP 4
25    W=ALFA*DSQRT(DALFA)
      ALFA=DY-DX+W+W
      IF (ALFA) 250,251,250
250   ALFA=(DY-Z+W)/ALFA
      GO TO 252
251   ALFA=(Z+DY-W)/(Z+DX+Z+DY)
252   ALFA=ALFA*AMBDA
      DO 26 I=1,N
26    X(I)=X(I)+(T-ALFA)*P(I)
      CALL FUNCT(N,X,F,G)
       IF (F-FX) 27,27,28
27    IF (F-FY) 36,36,28
28    DALFA=0.D0
      DO 29 I=1,N
29    DALFA=DALFA+G(I)*P(I)
      IF (DALFA) 30,33,33
```

```
30    IF (F-FX)32,31,33
31    IF (DX-DALFA) 32,36,32
32    FX=F
      DX=DALFA
      T=ALFA
      AMBDA=ALFA
      GO TO 23
33    IF (FY-F) 35,34,35
34    IF (DY-DALFA) 35,36,35
35    FY=F
      DY=DALFA
      AMBDA=AMBDA-ALFA
      GO TO 22
36    RETURN
```

BIBLIOGRAPHY

AFR 70 Afriat, S.N., The Progressive Support Method for Convex Programming, SIAM Journal of Numerical Analysis, Vol. 7, No. 3, September 1970, pp. 447-457.

AKA 59 Akaike, Hirotugu, On a Successive Transformation of Probability Distribution and Its Application to the Analysis of the Optimum Gradient Method, Annals Institute of Statistical Mathematics, Tokyo, Vol. 11, 1959, pp. 1-16.

ALB 72 Albert, Arthur, Regression and the Moore-Penrose Pseudoinverse, Academic Press, New York, 1972.

ANT 62 Antosiewicz, Henry A., and Werner C. Rheinboldt, Numerical Analysis and Functional Analysis, Survey of Numerical Analysis, J. Todd, ed. McGraw-Hill, New York, 1962, pp. 485-517.

BEN 66 Ben-Israel, Adi, A Newton-Raphson Method for the Solution of Systems of Equations, Journal of Mathematical Analysis and Applications, Vol. 15, 1966, pp. 243-252.

BRO 65 Broyden, C.G., A Class of Methods for Solving Nonlinear Simultaneous Equations, Mathematics of Computation, Vol. 19, 1965, pp. 577-593.

BRO 67 Broyden, C.G., Quasi-Newton Methods and Their Application to Function Minimisation, Mathematics of Computation, Vol. 21, 1967, 368-381.

BRO 69 Broyden, C.G., A New Method of Solving Nonlinear Simultaneous Equations, Computer Journal, Vol. 12, 1969, pp. 94-99.

BRO 70 Broyden, C.G., The Convergence of a Class of Double-Ranked Minimization Algorithms, Journal of the Institute of Mathematics and Applications, Vol. 6, 1970, pp. 76-90.

BRO 70a Broyden, C.G., The Convergence of a Class of Double-Rank Minimization Algorithms, Journal of the Institute of Mathematics and Applications, 1970, Vol. 6, pp. 222-231.

CAN 66 Canon, M., C. Cullum, and E. Polak, Constrained Minimization Problems in Finite-Dimensional Spaces, SIAM Journal of Control, Vol. 4, No. 3, 1966, pp. 528-547.

CHI 64 Chipman, John S., On Least Squares with Insufficient Observations, American Statistical Association Journal, December 1964, pp. 1078-1110.

CLI 64 Cline, Randall E., Representations for the Generalized Inverse of a Partitioned Matrix, SIAM Journal, Vol. 12, No. 3, September 1964, pp. 588-600.

CLI 65 Cline, Randall E., Representation for the Generalized Inverse of Sums of Matricies, SIAM Journal of Numerical Analysis, Ser. B, Vol. 2, No. 1, 1965, pp. 99-114.

COH 72 Cohen, Arthur I., Rate of Convergence of Several Conjugate Gradient Algorithms, SIAM Journal of Numerical Analysis, Vol. 9, No. 2, June 1972, pp. 248-259.

DAV 67 Davidon, William C., Variance Algorithm for Minimization, Computer Journal, Vol. 10, No. 4, 1967, pp. 406-411.

DAV 59 Davidon, William C., Variable Metric Methods for Minimization, A.E.C. Report ANL-5990, Argon National Laboratory.

DEI 67 Deist, F.H., and L. Sefor, Solution of Systems of Non-Linear Equations by Parameter Variation, Computer Journal, Vol. 10, 1967, pp. 78-82.

DES 63 Desoer, C.A., and B.H. Whalen, A Note on Pseudoinverses, SIAM Journal, Vol. 11, No. 2, June 1963, pp. 442-447.

FLE 63 Fletcher, R. and M.J.D. Powell, Rapidly Convergent Descent Method for Minimization, Computer Journal, Vol. 6, No. 2, 1963, pp. 163-168.

FLE 64 Fletcher, R., and C.M. Reeves, Function Minimization of Conjugate Gradients, Computer Journal, Vol. 7, 1964, pp. 149-154.

FLE 65 Fletcher, R., Function Minimization Without Evaluating Derivatives - A Review, Computer Journal, Vol. 8, 1965, pp. 33-41.

FLE 68 Fletcher, R., Generalized Inverse Methods for the Best Least Squares Solution of Systems of Non-Linear Equations, Computer Journal, Vol. 10, 1968, pp. 392-399.

FLE 69 Fletcher, R., A Review of Methods for Unconstrained Optimization, Optimization, Proceedings of an International Conference on Optimization at Keele Univ., Academic Press, 1969, pp. 1-13.

FLE 70 Fletcher, R., A New Approach to Variable Metric Algorithms, The Computer Journal, Vol. 13, No. 3, August 1970, pp. 317-322.

GRE 59 Greville, T.N.E., The Pseudoinverse of a Rectangular or Singular Matrix and Its Application to the Solution of Systems of Linear Equations, SIAM Review, Vol. 1, No. 1, January 1959, pp. 38-43.

GRE 60 Greville, T.N.E., Some Applications of the Pseudoinverse of a Matrix, SIAM Review, Vol. 2, No. 1, January 1960, pp. 15-22.

HOR 68 Horwitz, Lawrence B., and Philip E. Sarachik, Davidon's Method in Hilbert Space, SIAM Journal of Applied Mathematics, Vol. 16, No. 4, July 1968, pp. 676-695.

JON 70 Jones, A., Spiral - A New Algorithm for Non-Linear Parameter Estimation Using Least Squares, The Computer Journal, Vol. 13, No. 3, August 1970, p. 301.

KEL 66 Kelley, H.J., W.F. Denham, I.L. Johnson, and P.O. Wheatley, An Accelerated Gradient Method for Parameter Optimization with Non-Linear Constraints, Journal of the Astronautical Sciences, Vol. XIII, No. 4, 1966, pp. 166-169.

LAN 70 Lancaster, Peter, Explicit Solutions of Linear Matrix Equations, SIAM Review, Vol. 12, No. 4, October 1970, pp. 544-566.

LAS 67 Lasdon, L.S., S.K. Mitter and A.D. Waren, The Conjugate Gradient Method for Optimal Control Problems, IEEE Transactions on Automatic Control, Vol. AC-12, No. 2, April 1967, pp. 132-138.

LAS 70 Lasdon, Leon S., Conjugate Direction Methods for Optimal Control, IEEE Transactions on Automatic Control, April 1970, pp. 267-268.

LUE 70 Luenberger, David G., The Conjugate Residual Method for Constrained Minimization Problems, SIAM Journal of Numerical Analysis, Vol. 7, No. 3, September 1970, pp. 390-398.

MUR 70 Murtagh, B.A., and R.W.H. Sargent, Computational Experience with Quadratically Convergent Minimisation Methods, Computer Journal, Vol. 13, 1970, pp. 185-194.

MYE 68 Myers, Geraldine E., Properties of the Conjugate - Gradient and Davidon Methods, Journal of Optimization Theory and Applications, Vol. 2, No. 4, 1968, pp. 209-219.

NEL 65 Nelder, J.A., and R. Mead, A Simplex Method for Function Minimization, Computer Journal, Vol. 7, 1965, pp. 308-313.

ORT 70 Ortega, J.M., and W.C. Rheinboldt, Iterative Solution of Non-linear Equations in Several Variables, Academic Press, New York, 1970.

ORT 72 Ortega, James M. and Werner C. Rheinboldt, A General Convergence Result for Unconstrained Minimization Methods, SIAM Journal of Numerical Analysis, Vol. 9, No. 1, March 1972, pp. 40-42.

PEA 69 Pearson, J.D., Variable Metric Methods of Minimization, Computer Journal, Vol. 12, No. 5, 1969, pp. 171-178.

PEN 54 Penrose, R., A Generalized Inverse for Matricies, Proceedings of the Cambridge Philosophical Society, 1954, pp. 406-413.

PEN 55 Penrose, R., On Best Approximate Solutions of Linear Matrix Equations, Proceedings of the Cambridge Philosophical Society, 1955, pp. 17-19.

PET 70 Peters, G. and J.H. Wilkinson, The Least Squares Problem and Pseudo-Inverses, The Computer Journal, Vol. 13, No. 3, August 1970, pp. 309-316.

PIE 70 Pierson, B.L., and S.G. Rajtora, Computational Experience with the Davidon Method Applied to Optimal Control Problems, IEEE Transactions on System Science and Cybernetics, July 1970, pp. 240-242.

POR 72 Powers, William F., A Crude-Search Davidon-Type Technique with Application to Shuttle Optimization, AIAA Astrodynamics Conference, Palo Alto, Cal., September 1972, Paper No. 72-907.

POW 64 Powell, M.J.D., An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives, Computer Journal, Vol. 7, 1964, pp. 155-162.

POW 68 Powell, M.J.D., On the Calculation of Orthogonal Vectors, Computer Journal, Vol. 11, 1968, pp. 302-304.

POW 69 Powell, M.J.D., A Theorem on Rank One Modifications to a Matrix and Its Inverses, Computer Journal, Vol. 12, 1969, pp. 288-290.

POW 70 Powell, M.J.D., A New Algorithm for Unconstrained Minimization, Non-Linear Programming, Proceedings of a Symposium at the Univeristy of Wisconsin, Madison, Academic Press, 1970, pp. 31-65.

POW 70a Powell, M.J.D., A Survey of Numerical Methods for Unconstrained Optimization, SIAM Review, Vol. 12, 1970, pp. 79-97.

POW 71 Powell, M.J.D., On the Convergence of the Variable Metric Algorithm, Journal of the Institute of Mathematical Applications, Vol. 7, 1971, pp. 21-36.

PSH 69 Pshenichnii, Boris N., On the Acceleration of Convergence of Algorithms for Solving Optimal Control, Computing Methods in Optimization Problems 2, Proceedings of a Conference, San Remo, 1968, pp. 331-352, Academic Press, 1969.

RAO 71 Rao, C. Radhakrishna, and Sujit Kumar Mitra, Generalized Inverse of Matricies and Its Application, John Wiley & Sons, Inc., New York, 1971.

SHA 64 Shah, B.V., R.J. Buehler, and O. Kempthorne, Some Algorithms for Minimizing a Function of Several Variables, SIAM Journal, Vol. 12, No. 1, March 1964, pp. 74-92.

SHN 70 Shanno, D.F., Parameter Selection for Modified Newton Methods for Function Minimization, SIAM Journal of Numerical Analysis, Vol. 7, No. 3, September 1970, pp. 366-372.

TRI 70 Tripathi, Shiva S., and Kumpati S. Narendra, Optimization Using Conjugate Gradient Methods, IEEE Transactions on Automatic Control, April 1970, pp. 268-270.

VET 70 Vetter, W.J., Derivative Operations on Matricies, IEEE Transactions on Automatic Control, April 1970, pp. 241-243.

WAR 71 Ward, J.F., T.L. Boullion, and T.O. Lewis, A Note on the Oblique Matrix Pseudoinverse, SIAM Journal of Applied Mathematics, Vol. 20, No. 2, March 1971, pp. 173-175.

ZAN 67 Zangwill, Willard I., Minimizing a Function Without Calculating Derivatives, Computer Journal, Vol. 10, 1967, pp. 293-296.

ZEL 68 Zeleznik, Frank J., Quasi-Newton Methods for Nonlinear Equations, Journal of the Association for Computing Machinery, Vol. 15, No. 2, April 1968, pp. 265-271.

ZOU 66 Zoutendijk, G., Nonlinear Programming: A Numerical Survey, SIAM Journal of Control, Vol. 4, No. 1, 1966, pp. 194-210.