INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106
AGGARWAL, Anil Kumar, 1947-
UNIFIED APPROACH TO QUADRATICALLY CONVERGENT
ALGORITHMS FOR CONSTRAINED FUNCTION
MINIMIZATION.

Rice University, Ph.D., 1974
Engineering, aeronautical

University Microfilms, A XEROX Company, Ann Arbor, Michigan
ACKNOWLEDGEMENTS

My greatest debt of gratitude is to Dr. H.Y. Huang for suggesting the topic. His constant help and suggestions were invaluable. I would also like to thank Professor A. Miele for introducing the subject of mathematical programming to me; undoubtedly his influence is reflected in this work. In proofreading the manuscript, I received generous assistance and valuable comments from my colleague Mr. J.R. Cloutier. Much appreciation also goes to Miss E.L. Carter who unfledglingly typed several drafts of this text.

I would like to dedicate this thesis to my mother for her everlasting encouragement during the period of my gloom.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Statement of Problem</td>
<td>3</td>
</tr>
<tr>
<td>3. Review of Sequential Descent-Restoration Algorithms</td>
<td>4</td>
</tr>
<tr>
<td>4. Present Approach</td>
<td>8</td>
</tr>
<tr>
<td>5. General Solution of a System of Linear Equations</td>
<td>9</td>
</tr>
<tr>
<td>6. Quadratically Convergent Algorithms</td>
<td>20</td>
</tr>
<tr>
<td>7. Nonquadratic Function and Linear Constraint</td>
<td>31</td>
</tr>
<tr>
<td>8. Nonquadratic Function and Nonlinear Constraint</td>
<td>33</td>
</tr>
<tr>
<td>9. Numerical Experiments</td>
<td>40</td>
</tr>
<tr>
<td>10. Discussion and Conclusions</td>
<td>60</td>
</tr>
<tr>
<td>11. References</td>
<td>61</td>
</tr>
</tbody>
</table>
1. **Introduction**

In recent years, considerable attention has been given to iterative algorithms for minimizing a function \( f(x) \) subject to the constraint \( c(x) = 0 \), where \( f \) is a scalar, \( x \) an \( n \)-vector, and \( c \) an \( m \)-vector. With reference to the first-order methods, three main approaches are available, namely, the **penalty function method** (for instance, Refs. 1-2), the **augmented penalty function method** (for instance, Refs. 3-6), and the **sequential descent-restoration method**.

In the penalty function method and the augmented penalty function method, the original constrained problem is replaced by a series of mathematically simpler unconstrained problems. These approaches are attractive mainly because of their simplicity. However, the success of these methods depends critically on the choice of the **penalty constant**. Moreover, since the constraint is satisfied only at convergence, the values of the function obtained at the ends of successive unconstrained problems are not comparable.

The sequential descent-restoration method was first implied in Refs. 7-9 and was later introduced in Ref. 10. The method is composed of an alternate succession of **descent phases** and **restoration phases**. In a descent phase, a nominal point satisfying the constraint is assumed; a displacement leading to the varied point is found such that the value of the function is reduced while the constraint is satisfied to first order. If the constraint is nonlinear, it may be violated at the end of the descent phase. If this is the case, a restoration phase is performed before the start of the next descent phase. In the restoration phase, a nominal point violating the constraint is assumed; then a shortest displacement leading to the varied point satisfying the constraint is found. Since only the linearized constraint is employed in this operation, the restoration phase may
involve more than one iteration; if the constraint is nonlinear. A nice property of this method is that the constraint is satisfied at the end of each cycle so that the values of the function at the ends of restoration phases are comparable.

To accelerate the rate of convergence, a conjugate-gradient version of the sequential descent-restoration method was developed in Ref. 10 and a version using Davidon's variable metric method was given in Ref. 9. Some other versions utilizing quasi-Newton methods were later given in Refs. 11-14.

In this paper a class of quadratically convergent algorithms is developed from a unified point of view. The algorithms have the following properties: (a) the algorithms employ the functions and their gradients only, that is, they are first order methods; (b) they use one-dimensional search only, (c) they use only information at the present stage and the stage immediately prior to the present, and (d) for a quadratic function subject to a linear constraint, the algorithms converge quadratically to the solution, that is, the constrained minimal point is obtained in no more than n-m descent steps. It is hoped that this work will fully expose the relationship among the various methods previously presented in isolated ways. It is also hoped that through this work some practical new particular algorithms can be found.
2. **Statement of Problem**

We shall now consider the problem of minimizing the function

\[ f = f(x) \]  \hspace{1cm} (1)

subject to the constraint

\[ \phi(x) = 0 \]  \hspace{1cm} (2)

In the above equations, \( f \) is a scalar, \( x \) is an \( n \)-vector, and \( \phi \) is an \( m \)-vector, where \( m < n \). In this paper, all vectors are column vectors. The transpose of a vector \( x \) will be denoted by \( x^T \). It is assumed that the constrained minimum exists and that the first and second partial derivatives of the function \( f \) and \( \phi \) with respect to \( x \) exist and are continuous in a certain neighborhood of the constrained minimal point.

2.1. **Stationary Conditions.** If one introduces the augmented function

\[ F(x, \lambda) = f(x) + \lambda^T \phi(x) \]  \hspace{1cm} (3)

where the \( m \)-vector \( \lambda \) is the Lagrange multiplier, then the stationary conditions for the problem are given by

\[ F_x(x, \lambda) = f_x(x) + \phi_x(x)\lambda = 0 \]
\[ \phi(x) = 0 \]  \hspace{1cm} (4)

where the \( n \)-vector \( f_x \) is the gradient of \( f \) with respect to \( x \) and the \( n \times m \) matrix \( \phi_x \) denotes the gradient of \( \phi \) with respect to \( x \).

All the numerical methods for the solution of problem (1)-(2) must lead to the satisfaction of Eqs. (4), or their equivalent form.
3. **Review of Sequential Descent-Restoration Algorithms**

The sequential descent-restoration method is composed of an alternate succession of descent phases and restoration phases. The two phases of operations are briefly reviewed below.

**Descent Phase.** In a descent phase, a nominal point \( x \) satisfying the constraint is assumed. Then a displacement \( \Delta x \) leading to the varied point \( \tilde{x} = x + \Delta x \) is determined such that the first variation \( \delta F \) of the augmented function (3) is negative, that is,

\[
\delta F = F^T_x(x, \lambda) \Delta x < 0 \tag{5}
\]

and that the linearized constraint

\[
\varphi^T_x(x) \Delta x = 0 \tag{6}
\]

is satisfied.

A choice for the displacement satisfying (5)-(6) is given by

\[
\Delta x = -\alpha H F_x(x, \lambda) \tag{7}
\]

where \( \alpha \) is a positive scaling factor and \( H \) is a positive definite, symmetric \( n \times n \) matrix. With this choice, Ineq. (5) is satisfied. The multiplier \( \lambda \) in Eq. (7) is determined such that Eq. (6) is satisfied. Substituting (7) into (6), one obtains

\[
\varphi^T_x(x) H f_x(x) + \varphi^T_x(x) H \varphi_x(x) \lambda = 0 \tag{8}
\]

defining the multiplier \( \lambda \). If the matrix \( \varphi_x \) is of maximal rank, then \( \lambda \) is explicitly given by
\[
\lambda = -\left[ \varphi_x^T(x) H \varphi_x(x) \right]^{-1} \varphi_x^T(x) H f_x(x) \tag{9}
\]

Therefore, the displacement \( \Delta x \) satisfying (5)-(6) becomes,

\[
\Delta x = -\alpha A H f_x(x) \tag{10}
\]

where \( A \) is an \( n \times n \) matrix defined by

\[
A = I - H \varphi_x(x) \left[ \varphi_x^T(x) H \varphi_x(x) \right]^{-1} \varphi_x^T(x) \tag{11}
\]

The matrix \( A \) defined above is a projection matrix satisfying

\[
\varphi_x^T(x) A = 0 \tag{12}
\]

The matrix \( H \) in Eqs. (10)-(11) can assume various forms. When \( H = I \), an identity matrix, Eqs. (10)-(11) define the simple gradient projection method (Refs. 7, 8, 10). When \( H \) is the update matrix for quasi-Newton methods for unconstrained function minimization (Refs. 15-16), Eqs. (10)-(11) give the equivalent methods for constrained function minimization. For instance, the methods in Refs. 9, 11 are obtained by using \( H \) as the Fletcher-Powell-Davidon update matrix.

**Restoration Phase.** Restoration phase is desired only when the constraint is violated at a particular stage of the algorithm. Therefore, at the start of this phase, a nominal point \( x \) not satisfying the constraint is assumed and a displacement \( \Delta x \) leading to the varied point \( \tilde{x} = x + \Delta x \) satisfying the constraint is determined. It is noted that the displacement defined above is not unique. We shall further restrict the displacement to be the shortest of all the possible ones. Specifically, we consider the problem of minimizing the function
\[ J = \frac{1}{2} \Delta x^T \Delta x \]  

subject to the linearized constraint

\[ \beta \Delta x(x) + \Delta x^T \Delta x = 0 \]  

where \( \beta, 0 < \beta < 1 \) is a positive scaling factor. The problem (13)-(14) can be reformulated as that of minimizing the augmented function

\[ W(\Delta x; \sigma) = \frac{1}{2} \Delta x^T \Delta x + \sigma^T \left[ \beta \Delta x(x) + \Delta x^T \Delta x \right] \]  

subject to (14). In Eq.: (15), the \( m \)-vector \( \sigma \) is the Lagrange multiplier. The solution to the problem is determined by

\[ \Delta x(x) + \Delta x^T \Delta x = \sigma \]  

\[ \beta \Delta x(x) + \Delta x^T \Delta x = 0 \]  

Eliminating \( \Delta x \) from Eqs. (16), one obtains

\[ \beta \Delta x(x) - \Delta x^T \Delta x = 0 \]  

defining the multiplier. If the matrix \( \Delta x \) is of maximal rank, the multiplier and the displacement are explicitly given by

\[ \sigma = \beta \left[ \Delta x^T \Delta x \right]^{-1} \Delta x(x) \]  

\[ \Delta x = -\beta \Delta x(x) \left[ \Delta x^T \Delta x \right]^{-1} \]  

When the constraint is linear, the displacement given by Eq. (18-2) with \( \beta = 1 \) restores the constraint completely, that is, \( \Delta x(x) = 0 \). Thus, the solution is obtained in one iteration. When the constraint is nonlinear, additional iterations
may be needed to restore the constraint. That this process will be successful in restoring the constraint is demonstrated by the following descent property.

Let \( \Phi^T \Phi \) denote the error in the constraint and let \( \delta(\Phi^T \Phi) \) denote the first variation of the function \( \Phi^T \Phi \), that is,

\[
\delta(\Phi^T \Phi) = +2 \Phi^T \frac{\partial}{\partial x} \Phi^T \frac{\partial}{\partial x} \Delta x
\]  

(19)

Substituting (16-2) into (19), one obtains

\[
\delta(\Phi^T \Phi) = -2 \beta \Phi^T \Phi < 0
\]

(20)

Ineq. (20) thus guarantees the reduction of error in the constraint at each iteration if \( \beta \) is sufficiently small.
4. **Present Approach**

In this paper, we intend to solve the problem (1)-(2) using the concept of the sequential descent-restoration method. Specifically, the present approach will make use of the descent phase defined by (5)-(6) and the restoration phase defined by (13)-(14). However, the explicit solutions given by Eqs. (10)-(11) for the descent phase and by Eq. (18-2) for the restoration phase are to be bypassed.

Computation of steps in the descent phase and the restoration phase through Eqs. (10)-(11) and (18-2) has the following drawback: the inverses in Eqs. (11) and (18-2) exist only when (a) the matrix $\varphi_x$ is of maximal rank and (b) the matrix $H$ is of full rank. When either of the two conditions is violated, Eqs. (10)-(11) and (18-2) cannot be applied. The restriction (b) also excludes the use of algorithms where the update matrix $H$ is not of full rank (for example, the class of algorithms in Ref. 16 with $\rho = 0$).

By the present approach, we want to accomplish the following: (a) the new method is also applicable for the general case where the matrix $\varphi_x$ may be of maximal rank but ill-conditioned, or where the matrix $\varphi_x$ may not be of maximal rank, and (b) the class of quadratically convergent algorithms be extended to include not only the quasi-Newton type but other types as well.

This approach is made possible through a stable direct computing method for the general solution of a system of linear equations. The method is discussed in Section 5. Then, the quadratically convergent algorithms are developed for the case of quadratic function and linear constraint in Section 6. Extension of the algorithms to the general case is given in Sections 7-8.
5. **General Solution of a System of Linear Equations**

In this section, a direct method for the general solution of a system of linear equations is given. The system of equations considered is given in the form

\[ \psi(x) = Q^T x - h = 0 \]  \hspace{1cm} (21)

where \( \psi \) is an \( m \)-vector, \( Q \) is an \( n \times m \) matrix, and \( h \) is an \( m \)-vector, with \( m \leq n \). It is assumed that the matrix \( Q^T \) and the augmented matrix \([Q^T ; h]\) are of the same rank \( r \leq m \). Therefore, the system is consistent and admits solution(s). The method yields computationally a shortest distant solution \( x_r \), and an \( n \times n \) matrix \( A_r \) of rank \( n-r \) so that the expression

\[ x = x_r + A_r y \]  \hspace{1cm} (22)

satisfies the system (21) for all \( n \)-vector \( y \). The solution (22) is called the general solution of the system (21). In the limiting case when \( r = n \), the matrix \( A_r \) becomes a zero matrix and \( x_r \) represents the unique solution of the system (21).

**5.1. Basic Method.** To facilitate the discussion, we rewrite the system (21) in the form

\[ \psi_i(x) = q_i^T x - h_i, \quad i = 1, 2, \ldots, m \]  \hspace{1cm} (23)

where \( h_i \) and \( \psi_i \) are the \( i \)th scalar components of the vectors \( h \) and \( \psi \), respectively, and \( q_i \) is the \( i \)th column vector of the matrix \( Q \). For a given \( x \), \( \psi_i(x) \) represents the error or residual of the \( i \)th equation. For the time being, assume \( q_i \), \( i = 1, 2, \ldots, r \) to be linearly independent and \( q_i \), \( i = r+1, r+2, \ldots, m \) to be linearly dependent on the previous equations. The basic method which leads to
the general solution of linear equations (23) is given below:

Let $x_0 = 0$ and let $A_0$ be an identity matrix of dimension $n$, then, the following computations can be performed for $i = 1, 2, \ldots, r$:

$$p_i = A_{i-1} q_i$$

$$x_i = x_{i-1} - \frac{\psi(x_{i-1})}{p_i^T p_i} p_i$$  \hspace{1cm} (24)

$$A_i = A_{i-1} - \frac{p_i^T p_i}{p_i^T q_i}$$

where $p_i$ is an $n$-vector and $A_i$ is a symmetric $n \times n$ matrix. When $x_r$ and $A_r$ are obtained, the general solution is then given by Eq. (22). This is proved below.

For $i = 1$, we have

$$p_1 = q_1 \neq 0, \quad p_1^T q_1 = p_1^T p_1 > 0$$  \hspace{1cm} (25)

Therefore, $x_1$ and $A_1$ are well defined and can be computed. The point $x_1$ given by Eq. (24-2) satisfies the first equation, namely,

$$\psi(x_1) = 0$$  \hspace{1cm} (26)

The matrix $A_1$ given by Eq. (24-3) has the property

$$A_1 q_1 = 0$$  \hspace{1cm} (27)

Because of (25-1), Eq. (27) leads to

$$A_1 p_1 = 0$$  \hspace{1cm} (28)
Since $A_0$ is of rank $n$ and $A_1$ is obtained from $A_0$ by a correction of rank one, the rank of $A_1$ is at least $n-1$. On the other hand, since $A_1$ satisfies Eq. (27), its rank is at most $n-1$. Therefore, $A_1$ is of rank $n-1$.

Due to the properties (26)-(27), any point $x$ defined by

$$x = x_1 + A_1 y$$  \hspace{1cm} (29)

where $y$ is an arbitrary $n$-vector, satisfies the equation $\psi_1(x) = 0$

For $i = 2$, we have

$$p_2 = q_2 - \frac{p_1^T q_2}{p_1^T p_1} p_1 \neq 0$$  \hspace{1cm} (30)

since $q_1$ and $q_2$ are, by definition, linearly independent. From Eqs. (27)-(28), we can obtain the relations

$$p_2^T q_1 = p_2^T p_1 = 0$$  \hspace{1cm} (31)

In the light of Eqs. (30)-(31), we have

$$p_2^T q_2 = p_2^T p_2 > 0$$  \hspace{1cm} (32)

Therefore, $x_2$ and $A_2$ are well defined and can be computed. The point $x_2$ given by Eq. (24-2) satisfies the equations

$$\psi_1(x_2) = \psi_2(x_2) = 0$$  \hspace{1cm} (33)

and the matrix $A_2$ given by Eq. (24-3) has the properties

$$A_2 q_1 = A_2 q_2 = 0$$

$$A_2 p_1 = A_2 p_2 = 0$$  \hspace{1cm} (34)
Since $A_1$ is of rank $n-1$ and $A_2$ is obtained from $A_1$ by a correction of rank one, the rank of $A_2$ is at least $n-2$. On the other hand, since $A_2$ satisfies Eq. (34-1), its rank is at most $n-2$. Therefore, $A_2$ is of rank $n-2$.

Due to the properties (33)-(34), any point $x$ defined by

$$x = x_2 + A_2y$$  \hspace{1cm} (35)

where $y$ is an arbitrary $n$-vector, satisfies the equations $\psi_1(x) = \psi_2(x) = 0$.

The same procedure can be repeated for additional steps. It can be established that, if $q_j$, $j = 1, 2, \ldots, r$, are linearly independent, the vectors $p_j^T$, $j = 1, 2, \ldots, r$, are nonzero and the products $p_j^T q_j$, $j = 1, 2, \ldots, r$, are positive. Therefore, the method defined by Eqs. (24) is well defined and $x_j$ and $A_j$, $j = 1, 2, \ldots, r$, can be computed. The following properties can be established for $i \leq r$:

$$p_i^T q_i = p_i^T p_i \neq 0; \quad p_i^T q_j = p_i^T p_j = 0; \quad 1 \leq j \leq i-1$$  \hspace{1cm} (36-1)

$$\psi_j(x_i) = 0, \quad 1 \leq j \leq i; \quad A_i q_j = A_i p_j = 0, \quad 1 \leq j \leq i$$  \hspace{1cm} (36-2)

In addition, the matrix $A_i$ is of rank $n-i$ and any point $x$ defined by

$$x = x_i + A_i y$$  \hspace{1cm} (37)

where $y$ is an arbitrary $n$-vector, satisfies the equations $\psi_j(x) = 0$, $1 \leq j \leq i$.

When $x_r$ and $A_r$ are obtained, any point $x$ defined by Eq. (22) satisfies the equations $\psi_j(x) = 0$, $j = 1, 2, \ldots, r$. Now, for $x$ defined by Eq. (22), the functions $\psi_j(x)$, $j = r+1, r+2, \ldots, m$, can be rewritten in the form
\[ \psi_j(x) = \psi_j(x_r) + q_j^T A_x \dot{y}, \quad j = r+1, r+2, \ldots, m \]  

(38)

Since \( A_x \) has the property (35-2), that is,

\[ A_x q_j = 0, \quad j = 1, 2, \ldots, r \]  

(39)

and, by assumption, \( q_j, j = r+1, r+2, \ldots, m \), are linearly dependent on \( q_j \), \( j = 1, 2, \ldots, r \), it follows that

\[ A_x q_j = 0, \quad j = r+1, r+2, \ldots, m \]  

(40)

Therefore, Eq. (38) reduces to

\[ \psi_j(x) = \psi_j(x_r), \quad j = r+1, r+2, \ldots, m \]  

(41)

Since we assume that the system (23) is consistent, \( \psi_j(x_r), j = r+1, r+2, \ldots, m \) must be zero. This being the case, Eq. (41) reduces to

\[ \psi_j(x) = 0, \quad j = r+1, r+2, \ldots, m \]  

(42)

for all \( x \) defined by Eq. (22). Thus, any point \( x \) given by Eq. (22) satisfies the system (23).

**Modification.** In actual situations, one may not know \textit{a priori} that a certain set of \( q_j \) is linearly independent. This being the case, the previous procedure can not be followed strictly and a slight modification is needed. Two modified versions are given below.

**Version I.** In this version, one still treats the equations in the same order as they are given. At any step, one computes \( p_i \) and \( \psi_i(x_{i-1}) \). If \( p_i \neq 0 \), one proceeds to the computation of \( x_i \) and \( A_i \) according to Eqs. (24-2)-(24-3). If
$p_i = 0$ and $\psi_i(x_{i-1}) = 0$, the equation is linearly dependent on the previous equations and can be discarded completely. One bypasses the computation defined by Eqs. (24-2)-(24-3) and proceeds to the next step. If $p_i = 0$ and $\psi_i(x_{i-1}) \neq 0$, the equation is not consistent with the previous equations. For a well formulated problem, as we have assumed, this last case will never be encountered. Therefore, one can cover all the equations and, in the process, select a set of linearly independent equations and obtain the general solution.

**Version II.** In this version, one does not necessarily treat the equations in the same order as they are given. At any step, one treats, among the equations not yet treated, the particular one which provides the largest denominators in Eqs. (24-2)-(24-3). This process is generally known as **pivoting.** It has the effect of reducing computational errors for most cases. Therefore, this version is preferred for general usage. The process is elaborated below.

We assume that the exchange of indexes of equations is performed at every step, whenever necessary, so that, at the beginning of the $i$th step, the equations $\psi_j(x) = 0$, $j = 1, 2, \ldots, i-1$, represent the equations already treated and the equations $\psi_j(x) = 0$, $j = i, i+1, \ldots, m$, represent the equations yet to be treated. Since the matrix $A_{i-1}$ is available, one can compute the $n$-vectors

$$p_i^{(j)} = A_{i-1}q_j,$$  

$j = i, i+1, \ldots, m$ \hspace{1cm} (43)

Then, let

$$\left\| p_i^{(s)} \right\| = \max_{j \in J} \left\| p_i^{(j)} \right\|$$ \hspace{1cm} (44)
where $J$ is the set $(i, i+1, \ldots, m)$, one exchanges the indexes of the $i$th equation and the $s$th equation. Of course, the exchange of indexes can be bypassed when $s = i$. After the rearrangement, one has

$$p_i = p_i^{(i)}, \text{ where } \left\| p_i^{(i)} \right\| = \max_{j \in J} \left\| p_i^{(j)} \right\|$$

(45)

and the computation defined by Eqs. (24-2)-(24-3) follows.

If the matrix $Q$ is of rank $r$, the following property holds at the end of $r$ steps:

$$p_{r+1}^{(j)} = A_{r+1}^{j} = 0, \quad j = r+1, r+2, \ldots, m$$

(46)

If the system is consistent, as we have assumed, then $\psi_j(x_i) = 0$, $j = r+1, r+2, \ldots, m$, and the general solution is obtained. On the other hand, if Eqs. (46) are satisfied and any of the residuals $\psi_j(x_i)$, $j = r+1, r+2, \ldots, m$, is not zero, the system is inconsistent and there is no solution to the system.

5.2. Alternate Method. To facilitate the pivoting, an alternate computing procedure can be employed which bypasses the explicit use of Eq. (43). The procedure was established in Ref. 17. Here, for brevity, we just give the formulas used in computation.

At the beginning of the $i$th step, one computes $p_i^{(j)}$ and $d_i^{(j)}$ for $j = i$, $i+1, \ldots, m$, $i > 1$, as follows:

$$p_i^{(j)} = p_i^{(j)} - \frac{T}{p_i^{T} p_i^{(j)} p_{i-1}} p_{i-1}$$

(47)
with the understanding that \( p^{(j)}_1 = q_j \) and \( d^{(j)}_1 = h_j \), \( j = 1, 2, \ldots, m \). After exchange of indexes, one sets

\[
p_i = p^{(i)}_i, \quad d_i = d^{(i)}_i
\]

(48)

where \( p^{(i)}_i \) is defined in Eq. (45).

The procedure described by Eqs. (47)-(48) is, in effect, a transformation of a set of equations into an equivalent set at every step. Specifically, at the end of \( r \) steps, the original system is transformed into the system

\[
\begin{align*}
(p^{(j)}_{x+1})^T x - d^{(j)}_{x+1} &= 0, & j &= r+1, r+2, \ldots, m, \\
p^T_j x - d_j &= 0, & j &= 1, 2, \ldots, r
\end{align*}
\]

(49)

If the matrix \( Q \) is of rank \( r \), property (46) holds at the end of \( r \) steps. If the system is consistent, as we have assumed, then \( d^{(j)}_{x+1} = 0 \), \( j = r+1, r+2, \ldots, m \). On the other hand, if (46) is satisfied and any of the quantities \( d^{(j)}_{x+1} \), \( j = r+1, r+2, \ldots, m \), is not zero, the system is inconsistent and there is no solution to the system. For a consistent system, the particular solution \( x_r \) and the matrix \( A_r \) are given by

\[
\begin{align*}
x_r &= \sum_{i=1}^{r} \frac{d_i}{p^T_i q_i} p_i \\
A_r &= I - \sum_{i=1}^{r} \frac{p_ip^T_i}{p^T_i q_i}
\end{align*}
\]

(50)

5.3. Solution of Minimum Modulus. Here, we examine a characteristic of the methods which bears an important implication when we later apply the
methods to the case of nonlinear constraints. This characteristic can be stated as follows: The methods of Sections 5.1, 5.2 lead to, among all the solutions of the system (21), the particular solution \( x_r \) which minimizes the modulus \( |x| \), where \( x \) is given by Eq. (22).

Let \( Z \) denote the square of the modulus of \( x \). Because of Eq. (22), one has

\[
Z = (x_r + A_y y)^T (x_r + A_y y)
\]  

(51)

Since \( x_r \) and \( A_y \) are known, \( Z \) is a scalar function of the \( n \)-vector \( y \). The conditions for minimizing \( Z \) are given by

\[
\begin{align*}
Z_y &= 2A_y (x_r + A_y y) = 0 \\
Z_{yy} &= 2A_y A_y \geq 0
\end{align*}
\]  

(52)

where \( Z_y \) is the gradient of \( Z \) and \( Z_{yy} \) is the Hessian of \( Z \), regarding \( y \) as the variable.

Ineq. (52-2) represents the nonnegative definiteness of the Hessian \( Z_{yy} \) and is obviously satisfied. Eq. (52-1) represents the vanishing of the gradient \( Z_y \). Since \( A_y x_r = 0 \), \( A_y A_y = A_y \), Eq. (52-1) becomes

\[
A_y y = 0
\]  

(53)

Therefore, the vector \( y \) which minimizes \( Z \) must satisfy Eq. (53). Since \( A_y \) is of rank \( n-r \), the solution of Eq. (53) is not unique. In view of Eq. (50-2), one can easily conclude that any \( n \)-vector which is a linear combination of the vectors \( p_i = i = 1, 2, \ldots, r \), satisfies Eq. (53). With Eq. (53) satisfied, Eq. (22) leads
to \( x = x_r \), stating that the particular solution \( x_r \) minimizes the function \( Z \).

5.4. Numerical Considerations. Some numerical considerations pertaining to the method are given below:

(i) In floating-point computations, it is generally not easy to determine the rank of the matrix \( Q \). If the original matrix \( Q \) is represented exactly and we use exact arithmetic, then the rank of \( Q \) will be determined by the condition (46). On the other hand, in floating-point computations, one has to decide as to when the remaining vectors \( p^{(j)}_{r+1}, j = r+1, r+2, \ldots, m \) can be regarded as null vectors during the process of reduction. In practice, one can determine rank by replacing condition (46) with

\[
\left\| p^{(j)}_{r+1} \right\| \leq \varepsilon_1 \left\| q_j \right\|, \quad j = r+1, r+2, \ldots, m \tag{54}
\]

where \( \varepsilon_1 \) is a small positive number.

(ii) If the constraint is consistent, \( d^{(j)}_{r+1} = 0, j = r+1, r+2, \ldots, m \). In practice, this condition can be replaced by

\[
\left\| d^{(j)}_{r+1} \right\| \leq \varepsilon_2 \left\| q_j \right\| \cdot \left\| x_r \right\|, \quad j = r+1, r+2, \ldots, m \tag{55}
\]

where \( \varepsilon_2 \) is a small positive number. A reasonable choice of \( \varepsilon_1 \) and \( \varepsilon_2 \) is in the order of \( 10^{-s/2} \), where \( s \) is the significant digits in the representation of a floating point number.

(iii) Due to property (36-1), the product \( p^T_1 q_i \) in Eqs. (47) can be replaced by \( p^T_1 p_1 \). The method so obtained will generate \( p \)-vectors equivalent to the modified Gram-Schmidt orthogonalization process by columns of the matrix \( Q \). When the matrix \( Q \) is of full rank and well conditioned, quantities \( p^T_1 q_i \) and
\( p_i^T p_i \) are theoretically and computationally identical. However, when Q is ill-conditioned, the magnitude of \( p_i^T q_i \) and \( p_i^T p_i \) differ in order of magnitude, with \( \| p_i^T q_i \| >> \| p_i^T p_i \| \). In case condition (54) fails to detect the linearly dependency of equations, use of \( p_i^T p_i \) in Eqs. (47) would lead to erroneous p and d, thus resulting in bad values for \( x_r \) and \( A_r \). On the other hand, when \( r_i^T q_i \) is used, the algorithms are self-correcting in that the effects due to the inadvertent inclusion of \( p_i \) are reduced. Therefore, the algorithms are stable and very good results are obtained for \( x_r \) and \( A_r \). Since computational determination of linear dependency is always a difficult task, the use of \( p_i^T q_i \) in the above-mentioned equations is of paramount importance.
6. Quadratically Convergent Algorithms

In this section, we shall develop a class of quadratically convergent algorithms having the properties (a)-(d) as mentioned in the introduction. These algorithms are based on the model problem of minimizing a quadratic function

$$f = a + b^T x + \frac{1}{2} x^T C x$$  \hspace{1cm} (56)

subject to a linear constraint

$$c(x) = Q^T x - h = 0$$  \hspace{1cm} (57)

The scalar $a$, the $n$-vector $b$, and the $n \times n$ symmetric matrix $C$ are constant quantities. We assume that the matrix $C$ is positive definite or semidefinite, that the constraint is consistent, and that a constrained relative minimum exists.

Since the constraint (57) is linear, the general solution of the constraint can be obtained through the methods described in Section 5. Therefore, the vector $x$ satisfying the constraint (57) can be expressed by

$$x = x_x + A_x y$$  \hspace{1cm} (58)

with the understanding that the vector $x_x$ and the matrix $A_x$ are known. After substituting this general solution into (56), the function $f$ becomes

$$f = \bar{a} + \bar{b}^T y + \frac{1}{2} y^T \bar{C} y$$  \hspace{1cm} (59)

where the scalar $\bar{a}$, the $n$-vector $\bar{b}$, and the $n \times n$ symmetric matrix $\bar{C}$ are constant and are given by
\[ \bar{a} = a + b^T x_r + \frac{1}{2} x_r^T C x_r \]
\[ \bar{b} = A_r (b + C x_r) \]
\[ \bar{c} = A_r C A_r \]

Therefore, the original problem becomes that of minimizing the function (59)

with respect to the variable \( y \), \( y \) being a free variable of dimension \( n \).

Let \( g \) and \( G \) denote the gradients of \( f \) with respect to \( x \) and \( y \), respectively, that is

\[ g = f_x = b + C x \]
\[ G = f_y = b + C y \] (60)

The two gradients are related by

\[ G = A_r g \] (61)

At the minimal point of the function (59), the gradient \( G \) must vanish, namely,

\[ G = A_r g = 0 \] (62)

Condition (62) completely defines the stationary values of the transformed problem.

Since this transformed problem is not constrained, all the algorithms for unconstrained function minimization can be applied directly. Specifically, we shall consider the class of quadratically convergent algorithms given in Ref. 16. The algorithms are described below.

Let the subscript \( i \), \( i \geq r \), denote the iteration number at some stage of the algorithm. The starting iteration is defined when \( i = r \), and we take the starting point \( y_r = 0 \) corresponding to the point \( x_r \). With the starting point defined, the
the stepwise procedure to update \( y_i \) is as follows:

(i) Choose an initial \( n \times n \) matrix \( H_i \) so that the matrix \( (H_i + H_i^T) \) is positive definite.

(ii) Update the vector \( y \) by the relations

\[
\begin{align*}
    s_i &= H_i^T g_i, & \Delta y_i &= -\alpha_i s_i, & y_{i+1} &= y_i + \Delta y_i
\end{align*}
\]

(63)

where \( s_i \) is the search direction, \( \Delta y_i \) is the displacement leading from \( y_i \) to \( y_{i+1} \), and \( \alpha_i \) is the stepsize determined by a one-dimensional search so that the function \( f(y_i - \alpha_i s_i) \) is minimized.

(iii) Except for \( i = r \), the matrix \( H_i \) used in step (ii) is updated by the rule

\[
\begin{align*}
    H_i &= H_{i-1} + \rho D_{i-1} - E_{i-1} \\
    D_{i-1} &= \Delta y_{i-1} (C_1 \Delta y_{i-1} + C_2 H_{i-1}^T \Delta g_{i-1})^T / (C_1 \Delta y_{i-1} + C_2 H_{i-1}^T \Delta g_{i-1})^T \Delta g_{i-1} \\
    E_{i-1} &= H_{i-1} \Delta g_{i-1} (K_1 \Delta y_{i-1} + K_2 H_{i-1}^T \Delta g_{i-1})^T / (K_1 \Delta y_{i-1} + K_2 H_{i-1}^T \Delta g_{i-1})^T \Delta g_{i-1}
\end{align*}
\]

(64)

where \( D_{i-1} \) and \( E_{i-1} \) are \( n \times n \) matrices, and \( \rho, C_1, C_2, K_1, K_2 \) are arbitrarily prescribed real numbers subject to the only restriction that \( K_1 \) and \( K_2 \) must not vanish simultaneously. Due to the arbitrariness in the choice of the constants appearing in Eqs. (64), an infinite number of algorithms can be generated. Two important classes of algorithms are widely used, one corresponding to the case \( \rho = 1 \), and the other corresponding to the case \( \rho = 0 \). The algorithms for \( \rho = 1 \) are generally referred to as quasi-Newton methods. And the algorithms for \( \rho = 0 \) are known as projection methods.
As examples of particular algorithms which can be generated from Eqs. (64), we mention here only three which are characterized by symmetric corrections in Eqs. (64). In the class of quasi-Newton methods, the case where $C_1 = K_2 = 1$ and $C_2 = K_1 = 0$ leads to the well known Fletcher-Powell-Davidon algorithm and the case where $C_1 = K_1 = -C_2 = -K_2 = 1$ leads to the so-called rank-one algorithm. In the class of projection methods, the case where $K_1 = 0$ and $K_2 = 1$ leads to the widely known projection algorithm. For these algorithms, the $H$-matrices generated by Eqs. (64) are symmetric.

Properties. Some properties of the general algorithm pertinent to our discussions will be cited here. The details can be found in Ref. 16.

(a) The search direction $s_i$ defined by Eqs. (63)-(64) can be written in the form:

$$s_i = \gamma_i \overline{s}_i$$

(65)

where $\gamma_i$ is a scalar given by

$$\gamma_i = 1 - \frac{K_2 \Delta G_{i-1}^T H_{i-1}^T G_i}{(K_1 \Delta y_{i-1} + K_2 H_{i-1}^T \Delta G_{i-1})^T \Delta G_{i-1}}$$

(66)

and $\overline{s}_i$ is a vector given by

$$\overline{s}_i = \left[ \begin{array}{c} \Delta y_{i-1} \\ \sum_{j=r}^{i-1} \frac{\Delta y_j \Delta G_j}{\Delta y_j \Delta G_j} \\ I - \sum_{j=r}^{i-1} \frac{\Delta y_j \Delta G_j}{\Delta y_j \Delta G_j} H_{i-1}^T G_i \end{array} \right]$$

or, equivalently, by

$$\overline{s}_i = \left[ \begin{array}{c} \Delta y_{i-1} \\ \Delta y_{i-1} \Delta G_{i-1} \\ I - \frac{\Delta y_{i-1} \Delta G_{i-1}}{\Delta y_{i-1} \Delta G_{i-1}} H_{i-1}^T G_i \end{array} \right]$$

(68)
When \( i = r \), \( \gamma = 1 \) and \( \mathbf{s}_r = H_r^T G_r \). Therefore, for given \( \gamma_r \) and \( H_r \), all algorithms obtained from different sets of \( \rho, C_{1r}, C_{2r}, K_{1r}, K_{2r} \) generate same sequence of directions, differing only by a scaling factor. As a result, they generate the same sequence of points \( \gamma_{1r} \).

The scaling factor \( \gamma_{1r} \) becomes zero when \( K_{1r} \alpha_{1r-1} + K_{2r} = 0 \). This can happen only when algorithms with nonzero \( K_{1r} \) and \( K_{2r} \) are used. The rank-one algorithm is an example. However, the likelihood of this occurring in an actual computation is rather slim. Should \( \gamma_{1r} \) become zero, \( \mathbf{s}_r \) cannot be used as a search direction. This being the case \( \mathbf{s}_r \) given by Eq. (67) or (68) can be used instead of \( \mathbf{s}_r \) as the search direction.

(b) The sequence of vectors \( \mathbf{s}_i \), \( i \geq r \), which is same for all algorithms, has the properties:

\[
G_i^T \mathbf{s}_i = G_i^T H_i^T G_i \quad , \quad i \geq r
\]

\[
\mathbf{s}_i^T C_{ij} \mathbf{s}_j = 0 \quad , \quad i > j \geq r \quad (69)
\]

\[
G_i^T \mathbf{s}_j = 0 \quad , \quad i > j \geq r
\]

Eq. (69-1) implies that the product \( G_i^T \mathbf{s}_1 \) vanishes only when the gradient \( G_i \) is zero. When \( G_i = 0 \), the condition (62) is satisfied and \( \gamma_1 \) is the minimal point. When \( G_i \neq 0 \), \( G_i^T \mathbf{s}_1 \neq 0 \) implying that \( \mathbf{s}_1 \neq 0 \). Therefore, a step can be taken and the next point is obtained. Another iteration follows. The process can be repeated until the point with zero gradient is reached. Eq. (69-2) implies that the nonzero vectors \( \mathbf{s}_i \) are linearly independent and Eq. (69-3) states that the gradient at a given point is orthogonal to all the previous search directions.
(c) In view of Eq. (65), an algorithm can be obtained by using the directions given by Eq. (67) or (68). It is equivalent to using the updating formula (64) with \( \rho = 0 \), \( K_2 = 0 \) and \( K_1 \neq 0 \). However, if \( H_r = I \), Eq. (67) can be reduced to

\[
s_i = G_i + \frac{G_i^T G_i}{G_{i-1}^T G_{i-1}} s_{i-1}
\]

with \( s_r = G_r \). Here, we use \( s_i \) for \( \overline{s}_i \). Thus, Eqs. (63-2), (63-3) and (70) form a complete algorithm, known as conjugate-gradient or Fletcher-Reeves algorithm.

**Convergence.** In the light of Eq. (61), we can write

\[
G_i^T \overline{s}_i = g_i^T A_{r_i} \overline{s}_i
\]

When \( G_i \neq 0 \), \( G_i^T \overline{s}_i \neq 0 \) as previously discussed. From Eq. (71), it follows that

\[
\hat{A}_{r_i} \overline{s}_i \neq 0
\]

implying that the vectors \( \overline{s}_i \), \( i \geq r \), are linearly independent of the vectors \( p_j \), \( j = 1, 2, \ldots, r \), defined in Section 5. Since both sets of vectors are linearly independent, it follows that the set of vectors \( p_j \), \( j = 1, 2, \ldots, r \), and \( \overline{s}_i \), \( i \geq r \), are linearly independent. Furthermore, one can have a maximum of \( n-r \) nonzero \( \overline{s}_i \) vectors.

From Eqs. (36-2) and (61), we can obtain the relation

\[
G_i^T p_j = 0, \quad j = 1, 2, \ldots, r, \quad i \geq r
\]

meaning that the gradient \( G_i \) is orthogonal to all \( p_j \), \( j = 1, 2, \ldots, r \). Combining Eqs. (69-3) and (73), we see that \( G_i \) is orthogonal to all the vectors \( p_j \), \( j = 1, 2, \ldots, r \).
..., r, and $\bar{s}_j, r \leq j < i$. Since $G_i$ is an n-vector, it must become zero with $i$ at most equal $n$. This means that the constrained minimal point is obtained in at most $n-r$ iterations of the quadratically convergent algorithms.

6.1. Algorithms Using $x$. Equations (63)-(64) constitute a class of algorithms which converge quadratically in a maximum of $n-r$ iterations. The algorithms were described in terms of the free variable $y$. They can be transformed back to the original variable $x$ by using the relation (58). Under this transformation, the use of $y$ can be bypassed completely.

Due to Eq. (58), the displacement in $x$ can be written in terms of the displacement in $y$, that is,

$$\Delta x_i = A_r \Delta y_i$$  (74)

Because of Eqs. (63-1), (63-2), and (61), $\Delta x_i$ becomes

$$\Delta x_i = -\alpha_i A_r H_r A_r g_i$$  (75)

This can be rewritten as

$$\Delta x_i = -\alpha_i u_i, \quad \text{where} \quad u_i = B_r T g_i, \quad B_i = A_r H_r A_r$$  (76)

$B_i$ is the new update matrix and $u_i$ is the new search direction.

Computation of the matrix $B_i$ through Eqs. (64) and (76-3) is rather cumbersome. We shall develop a direct updating formula for the B-matrix from the initial matrix $B_r = A_r H_r A_r$. To do this, we first premultiply and postmultiply Eqs. (64) with $A_r$. Then the resulting expression is simplified by using the following relations:
\[ A_r A_r = A_r \], \[ A_r H_{i-1} \Delta G_{i-1} = B_{i-1} \Delta g_{i-1} \], \[ A_r H_{i-1} \Delta G_{i-1} = B_{i-1} \Delta g_{i-1} \]

(77)

\[ \Delta y_{i-1}^T \Delta G_{i-1} = \Delta x_{i-1}^T \Delta g_{i-1} \], \[ \Delta G_{i-1}^T H_{i-1} \Delta G_{i-1} = \Delta g_{i-1}^T B_{i-1}^T \Delta g_{i-1} \]

The simplified updating formula for the B-matrix is identical in form with that of the H-matrix. Replacing H, y, G in Eqs. (64) with B, x, g, respectively, one obtains the updating formula for the B-matrix.

Finally, we can use an identity matrix for \( H_r \), that is,

\[ H_r = I \]

(78)

Then, the initial matrix \( B_r = A_r H_r A_r \) reduces to

\[ B_r = A_r \]

(79)

because of property (77-1). Eq. (79) states that one can use the projection matrix \( A_r \) obtained from the solution of the linear constraint as the initial matrix in the quadratically convergent algorithms for minimization.

Summarizing, the general quadratically convergent algorithm for constrained function minimization of (56)-(57) can be stated as follows:

(i) Use the initial point as \( x_r \) and the initial matrix \( B_r = A_r \), where \( x_r \) is the particular solution and \( A_r \) is the projection matrix obtained from the linear constraint.

(ii) Update the vector \( x \) through the relations

\[ u_i = B_i g_i \], \[ \Delta x_i = -\alpha_i u_i \], \[ x_{i+1} = x_i + \Delta x_i \]

(80)

where the stepsize \( \alpha_i \) is determined by a one-dimensional search so that the func-
tion \( f(x_i - \alpha u_i) \) is minimized.

(iii) Update the B-matrix according to the relation

\[
B_i = B_{i-1} + \rho M_{i-1} - N_{i-1}
\]

\[
M_{i-1} = \Delta x_{i-1} (C_1 \Delta x_{i-1} + C_2 B_{i-1}^T \Delta g_{i-1})^T / (C_1 \Delta x_{i-1} + C_2 B_{i-1}^T \Delta g_{i-1})^T \Delta g_{i-1}
\]

\[
N_{i-1} = B_{i-1} \Delta g_{i-1} (K_1 \Delta x_{i-1} + K_2 B_{i-1}^T \Delta g_{i-1})^T / (K_1 \Delta x_{i-1} + K_2 B_{i-1}^T \Delta g_{i-1})^T \Delta g_{i-1}
\]

(81)

where \( M_{i-1} \) and \( N_{i-1} \) are \( n \times n \) matrices, and \( \rho, C_1, C_2, K_1, K_2 \) are given constants for an algorithm. A particular algorithm is obtained by assigning values to the constants \( \rho, C_1, C_2, K_1, K_2 \) with the only restriction that \( K_1 \) and \( K_2 \) must not vanish simultaneously.

From Eqs. (65)-(68) and (70), one can obtain the following equivalent property for the present algorithm: The search direction \( u_i \) defined by Eqs. (80)-(81) can be written in the form:

\[
u_i = \gamma_i \bar{u}_i
\]

(82)

where \( \gamma_i \) is a scalar given by

\[
\gamma_i = 1 - \frac{K_2 \Delta g_{i-1} B_{i-1}^T g_i}{(K_1 \Delta x_{i-1} + K_2 B_{i-1}^T \Delta g_{i-1})^T \Delta g_{i-1}}
\]

(83)

and \( \bar{u}_i \) is a vector given by one of the three equivalent forms:

\[
\bar{u}_i = \begin{bmatrix}
A_x - \frac{\Delta x_{i-1} \Delta g_{i-1}}{\Delta x_{i-1} \Delta g_{i-1}}
\end{bmatrix} B_{i-1}^T g_i
\]

(84-1)

or

\[
\bar{u}_i = \begin{bmatrix}
A_x - \frac{i-1}{j=r} \frac{\Delta x_j \Delta g_j}{\Delta x_j \Delta g_j}
\end{bmatrix} A_x g_i
\]

(84-2)
or

\[ \bar{u}_i = G_i + \frac{G_i^T G_i}{G_{i-1}^T G_{i-1}} \bar{u}_{i-1} \]  \hspace{1cm} (84-3)

When \( i = r \), \( \gamma_r = 1 \) and \( \bar{u}_r = G_r = A_r g_r \). Therefore, all particular algorithms obtained from different sets of \( p, C_1, C_2, K_1, K_2 \), generate same sequence of directions, differing only by a scaling factor. As a result, all particular algorithms generate the same sequence of points \( x_i \).

For particular algorithms with both \( K_1 \) and \( K_2 \) nonzero, it is possible, though highly improbable in an actual computation, that \( \gamma_i \) becomes zero before the minimal point is reached. If this happens, Eq. (84-1) can be used to find the search direction for that iteration.

Eqs. (84-2)-(84-3) with \( u_i \) replaced by \( \bar{u}_i \) define the sequence of search directions. These equations, used in combination with Eqs. (80-2)-(80-3), form two algorithms which can be called simplified algorithms.

**Convergence.** The condition (62) for the constrained minimal point is achieved in at most \( n-r \) iterations of the quadratically convergent algorithms barring the loss of accuracy in computation.

In actual computation, the minimal point is considered achieved if the inequality

\[ \|G\| \leq \varepsilon_3 \]  \hspace{1cm} (85)

is satisfied. Here, \( \varepsilon_3 \) is a small positive number.

### 6.2. Alternate Forms of the General Algorithm

Due to the property (77-1) of the projection matrix \( A_r \), the gradient \( g_i \) in Eq. (80-1) and the gradient difference \( \Delta g_i \) in Eqs. (81) can be replaced by the gradient \( G_i \) and the gradient dif-
ference $\Delta G_i$, respectively. The replacement can be done for $g_i$ in Eq. (80-1) only, or for $\Delta g_i$ in Eqs. (81) only, or for both $g_i$ in Eq. (80-1) and $\Delta g_i$ in Eqs. (81).

As a consequence, there are four different forms of the general algorithm, all of which lead to the same result. Since $G_i$, as well as $g_i$, is computed at every iteration so that the convergence condition (85) can be checked, the four forms of the algorithm involve practically the same amount of computation.
7. **Nonquadratic Function and Linear Constraint**

The algorithms described in Section 6 can be applied to the problem of minimizing a nonquadratic function subject to a linear constraint with a slight modification. The process can be summarized as follows:

(i) Solve the linear constraint using the methods described in Section 5 to find the particular solution \( x_r \) and the projection matrix \( A_r \). The matrix \( A_r \) is stored for possible later usage.

(ii) Use \( x_r \) as the initial point and \( B_r = A_r \) as the initial matrix for the algorithms described in Section 6.2. Since the constrained minimal point for the nonquadratic function may not be obtained in \( n-r \) iterations, further iterations may be required. This being the case, restarting of the algorithms is needed. In this connection, we suggest the following:

(a) The algorithms characterized by \( \rho = 0 \) are restarted after every \( n-r \) iterations by setting

\[
B_i = A_r
\]

at that iteration instead of using the updating formula (81). The same rule can be followed for the simplified algorithms using the search directions given by Eqs. (84-2)- (84-3).

(b) The algorithms characterized by \( \rho = 1 \) can be restarted as (a) or can be continued without a restart.

(c) Besides the aforementioned restarting conditions, the algorithms are also restarted when
\[ \| g_1^T u_i \| \leq \varepsilon_4 \] (86)

is satisfied, where \( \varepsilon_4 \) is some small positive number.
8. **Nonquadratic Function and Nonlinear Constraint**

In this section, we shall apply the algorithms described in Section 6 to the problem of minimizing a nonquadratic function subject to a nonlinear constraint. However, due to the nonlinearity of the constraint, a descent step taken from a point on the constraint according to Eqs. (80)-(81) may lead to a point violating the constraint. Therefore, some procedure must be employed to restore the constraint when violation occurs.

One strategy is to restore the constraint only at the end of a cycle of the algorithms given in Section 6; here a cycle is made up of \( n-r \) descent steps. In this way, one essentially is minimizing the function subject to the constraint which is a hyperplane tangential to the constraining surface at the beginning point of the cycle. This strategy is rather unstable for highly nonlinear constraint.

A more stable strategy is to restore the constraint at the end of every descent step. We shall adopt this strategy and modify the algorithms of Section 6 accordingly. The resulting modified method will be made up of alternate succession of restoration phases and descent phases. In each descent phase, only one descent step is taken; in each restoration phase, one or more restoration steps are taken.

In order to describe the approach, we shall use the following notations:

Let \( x_i, \ i = 1, 2, \ldots, \) denote the nominal points for the successive descent phases. Then at \( x_i, \ \phi(x_i) = 0 \) holds for all \( i = 1, 2, \ldots \). Let \( \tilde{x}_i, \ i = 1, 2, \ldots, \) denote the nominal points for the restoration phases which precede the corresponding descent phases. Then, by definition \( \phi(\tilde{x}_i) \neq 0, \ i = 1, 2, \ldots \). With the nomenclature defined, a combination of a restoration phase and a descent phase can be
stated as follows: From a point $\tilde{x}_1$, a restoration step $\Delta \tilde{x}_1$ leading to the next point $x_1 = \tilde{x}_1 + \Delta \tilde{x}_1$ is found; this may involve more than one iteration. Then a descent step $\Delta x_1$ leading from a point $x_1$ to $\tilde{x}_{1+1} = x_1 + \Delta x_1$ is taken. From $\tilde{x}_{1+1}$, the next combination follows. A detailed description of the restoration phase and the descent phase is given below:

8.1. Restoration Phase. In a restoration phase, a nominal point $\tilde{x}_1$ not satisfying the constraint is assumed and the shortest displacement $\Delta \tilde{x}_1$ leading to the varied point $x_1 = \tilde{x}_1 + \Delta \tilde{x}_1$ satisfying the constraint is determined. As noted earlier, the displacement $\Delta \tilde{x}_1$ is uniquely given by Eqs. (13)-(14). This displacement can be obtained directly by solving Eq. (14) with the method given in Section 5. This is because the particular solution given by the method of Section 5 is the minimum modulus solution.

In order to eliminate the need for solving Eq. (14) once for every given value of the scaling factor $\beta_i$, we rewrite the equation as follows:

$$ \phi(\tilde{x}_1) + \phi_x^T(\tilde{x}_1)z_i = 0, \quad \Delta \tilde{x}_1 = \beta_i z_i, \quad x_1 = \tilde{x}_1 + \Delta \tilde{x}_1 \quad (87) $$

where $z_i$ is an $n$-vector and $\beta_i$ is a scaling factor in the range $0 < \beta_i \leq 1$. In this way Eq. (87-1) is solved only once with the method of Section 5. When this is done, one obtains, in addition to the minimum modulus solution $z_i$, the projection matrix $A_i$ of rank $n-r$ which satisfies

$$ \phi_x^T(\tilde{x}_1) A_i = 0 \quad (88) $$

where $r$ is the rank of the matrix $\phi_x(\tilde{x}_1)$. With $z_i$ known, the next point $x_i$ can be obtained from Eqs. (87-2)-(87-3) by first assigning $\beta_i = 1$. If the inequality
\[ \varphi^T(x_i) \varphi(x_i) < \varphi^T(x_i) \varphi(\tilde{x}_i) \] (89)

is satisfied, the point \( x_i \) is accepted. On the other hand, if Ineq. (89) is violated, the scaling factor is reduced successively, say by a bisection process, until the new point \( x_i \) so obtained satisfies Ineq. (89). The eventual satisfaction of Ineq. (89) is guaranteed by the descent property (20).

Once the new point satisfying Ineq. (89) is obtained, one further checks whether the condition

\[ \varphi^T(x_i) \varphi(x_i) \leq \varepsilon_5 \] (90)

is satisfied. In Ineq. (90), \( \varepsilon_5 \) is a small prescribed number. If Ineq. (90) is violated, the point \( x_i \) is taken as the starting point of the next restorative iteration. This process is repeated until Ineq. (90) is satisfied. The final point \( x_i \) satisfying Ineq. (90) is considered to be on the constraining surface and the final projection matrix \( A_i \) obtained is retained and used as an approximation for the projection matrix pertaining to the final point \( x_i \), that is, we assume

\[ \varphi^T(x_i) A_i = 0 \] (91)

is approximately true. Both \( x_i \) and \( A_i \) are then used in the descent phase that follows.

8.2. Descent Phase. In a descent phase, a nominal point \( x_i \) satisfying the constraint is assumed and a displacement \( \Delta x_i \) leading to the varied point \( \tilde{x}_{i+1} = x_i + \Delta x_i \) satisfying the constraint to first order is found so that the value of the function is reduced. Such a displacement must satisfy the linearized constraint equation (6), which, in the present notation, becomes
\[ \sigma_1^T(x_1) \Delta x_1 = 0 \]  

(92)

Since Eq. (92) is a homogeneous linear equation in \( \Delta x_1 \), the minimum modulus solution is zero. As a result the general solution of (92) is given by

\[ \Delta x_1 = -\alpha_i A_i u_i \]  

(93)

where \( u_i \) is an arbitrary \( n \)-vector and \( -\alpha_i \) is a scaling factor inserted for convenience.

If we define the arbitrary \( u_i \) in Eq. (93) as that given by Eq. (80-1), Eq. (93) becomes

\[ \Delta x_1 = -\alpha_i A_i B_i^T g_i \]  

(94)

Thus, the operation leading from \( x_i \) to \( \tilde{x}_{i+1} \) can be defined as follows:

\[ u_i = A_i B_i^T g_i, \quad \Delta x_i = -\alpha_i u_i, \quad \tilde{x}_{i+1} = x_i + \Delta x_i \]  

(95)

where \( u_i \) is the search direction and \( \alpha_i \) is the stepsize determined so that the function \( f(x_i - \alpha_i u_i) \) is minimized.

8.3. Complete Algorithm. Summarizing, a complete algorithm for problems involving nonlinear constraints can be stated as follows:

(i) For any given initial point \( \tilde{x}_1 \), the restoration phase described in Section 8.1 is used so that \( x_1 \) and \( A_1 \) are obtained. This is done regardless of whether \( \tilde{x}_1 \) satisfies the constraint or not.

(ii) Once \( x_1 \) and \( A_1 \) are known, a sequence of points \( x_2, x_3, \ldots \) are determined by the alternate use of restoration phases and descent phases.

The descent steps in general are given by Eqs. (95) and (81). At the
starting step, one takes the initial matrix \( B_1 \) to be

\[ B_1 = A_1 \]  \hspace{1cm} (96) 

Thereafter, the B-matrices are updated according to Eqs. (81).

At the end of every descent step, a restoration phase is inserted, if necessary, before taking the next descent step. The restoration is done according to the procedure described in Section 8.1. If at the end of a descent step, the point \( \tilde{x}_{i+1} \) satisfies the constraint, the restoration phase is bypassed and one takes \( x_{i+1} = \tilde{x}_{i+1} \) and

\[ A_{i+1} = A_i \]  \hspace{1cm} (97) 

and starts the next descent step.

(iii) A cycle of the algorithm is made up of \( n-r \) descent steps. When convergence is not achieved at the end of a cycle, the algorithm is restarted by using

\[ B_1 = A_1 \]  \hspace{1cm} (98) 

instead of that obtained from Eqs. (81). With the resetting of B-matrix according to Eq. (98), the next cycle ensues.

**Simplification.** It is noted that at the starting point or the restarting point of the algorithm, one takes the B-matrix according to (96) or (98). Since the projection matrix \( A_i \) has the property \( A_i A_i = A_i \), it follows that the search direction given by Eq. (95-1) reduces to

\[ u_i = \frac{B_i^T}{A_i} g_i \]  \hspace{1cm} (99)
where \( B_i = A_i \). Therefore, (99) can be used instead of Eq. (95-1).

Also, if the constraint is still satisfied at the end of a descent step, the new projection matrix \( A_{i+1} \) is given by (97). This being the case, one has

\[
A_{i+1} B_{i+1}^T = A_i B_i^T = B_i^T
\]

Therefore, the search direction is given by Eq. (99) with \( i \) replaced by \( i+1 \).

Consequently, when the constraint is not violated at the end of every descent step, Eq. (99) is used at every step instead of Eq. (95-1). The algorithm described here then reduces to that given by Eqs. (80)-(81).

8.4. Different Forms of the Algorithms. As mentioned in Section 6.2, four possible mathematical identical formulas to each of the algorithms described in Section 6 can be obtained. This is of course not true for the simplified algorithms since they do not involve any updating matrix \( B \). These different forms of the algorithms are summarized below:

Version A: \( u_i = A_i B_i g_i \) and \( B_i \) is updated using \( \Delta g_i \)

Version B: \( u_i = A_i B_i^T g_i \) and \( B_i \) is updated using \( \Delta G_i \)

Version C: \( u_i = A_i B_i^T G_i \) and \( B_i \) is updated using \( \Delta g_i \)

Version D: \( u_i = A_i B_i^T G_i \) and \( B_i \) is updated using \( \Delta G_i \)

These four different forms of the algorithms are mathematical consequences of one another and give identical results for the model problem of Section 6. Moreover, this is also true for any linear constraint problem regardless of the form of the objective function.

8.5. Different Modes of the Algorithms. The algorithms described in Section 8.1-8.3 state that the descent phase starts from a point satisfying the constraint. Therefore, after each descent step, a complete restoration of the
constraint is performed before taking the next descent step. However, a descent step can be taken from a point not satisfying the constraint. Such an algorithm with an incomplete restoration is computationally very attractive, unfortunately nothing can be said about its stability.

Similar computational simplification can be made if the displacement in the descent step is determined with incomplete search on the stepsize $\alpha$. In general, evaluation of the exact stepsize involves much computational work due to the excessive number of function evaluations needed. So for a nonquadratic function complete search may not be a desirable idea. Keeping this in mind, one can classify the different modes of the algorithms as follows:

- **Mode 1:** Complete restoration and complete search
- **Mode 2:** Complete restoration and incomplete search
- **Mode 3:** Incomplete restoration and complete search
- **Mode 4:** Incomplete restoration and incomplete search
9. **Numerical Experiments**

In this section, several numerical examples are solved using some of the particular algorithms obtained from the general class of quadratically convergent algorithms. The particular algorithms taken from three different classes of methods are: (i) the quasi-Newton algorithm (ii) the projection algorithm and (iii) the simplified algorithm. They are briefly stated below:

**Algorithm I:**

\[
B_i = B_{i-1} + \left(\Delta x_{i-1} - B_{i-1} \Delta g_{i-1}\right) (\Delta x_{i-1} - B_{i-1} \Delta g_{i-1})^T \cdot (\Delta x_{i-1} - B_{i-1} \Delta g_{i-1})^T \Delta g_{i-1}
\]

which is obtained by setting \( \rho = 1 \), \( C_1 = k_1 = -C_2 = -k_2 = 1 \). This algorithm is of the quasi-Newton type.

**Algorithm II:**

\[
B_i = B_{i-1} - \frac{B_{i-1} \Delta g_{i-1} \Delta g_{i-1}^T B_{i-1}^T}{\Delta g_{i-1}^T B_{i-1} \Delta g_{i-1}}
\]

which is obtained by setting \( \rho = 0 \), \( k_1 = 0 \), \( k_2 \neq 0 \). This algorithm is of the projection type.

**Algorithm III:** This algorithm directly defines the search vector at each iteration and thus does not need any updating of the B-matrix. It is stated as

\[
u_i = G_i + \frac{G_i^T G_i}{G_i^T G_i} A_i u_{i-1}
\]

These algorithms, along with the four versions A-D, are used to test the examples. Experiments are also made using incomplete restoration and incomplete search in order to check the relative performance of these algorithms.
9.1. Experimental Conditions. The three algorithms along with their four versions and four modes are studied and compared through several numerical examples. Computations were performed using the Rice University IBM 370/155 digital computer. Double precision arithmetic was used and the quadratically convergent algorithms were programmed in FORTRAN IV.

Rank Determination of Linear System. For the solution of linear equations, the matrix involved will be of rank \( r \) if condition (54) is satisfied. Also for the linear system to be consistent, condition (55) must hold. Thus,

\[
\| p^{(j)}_{r+1} \| \leq \varepsilon_1 \| q_j \| , \quad \| q^{(j)}_{r+1} \| \leq \varepsilon_2 \| q_j \| \cdot \| x_r \| , \quad j = r+1, r+2, \ldots, m \tag{100}
\]

where

\[
\varepsilon_1 = 10^{-8} , \quad \varepsilon_2 = 10^{-8}
\]

Search Criterion. In order to find the gradient stepsize \( \alpha \), we restrict ourselves to the use of the cubic interpolation mentioned in Ref. 18. The method is used iteratively until a rather precise stopping condition is satisfied. In this connection, the search is stopped when the inequality

\[
\| g_{i+1}^T u_i \| \leq \varepsilon_4 \tag{101}
\]

is satisfied, where

\[
\varepsilon_4 = 10^{-12}
\]

In an incomplete search, the cubic interpolation is used only once and the next phase of the algorithm is carried out, regardless of whether the Ineq. (101) is
satisfied or not.

**Restoration Criterion.** For the complete restoration of the constraint, the restoration phase is stopped only when the condition (90) is satisfied, that is,

$$ P = \phi_1^T \phi_1 \leq \epsilon_5 $$  \hspace{1cm} (102)

where

$$ \epsilon_5 = 10^{-6} $$

For incomplete restoration, only one restoration step is taken, regardless of whether the above inequality is satisfied or not at the end of the step.

**Convergence Conditions.** The algorithm for the constrained function minimization is stopped when the following inequalities are satisfied simultaneously, that is,

$$ Q = G_i^T G_i \leq \epsilon_3, \quad P = \phi_1^T \phi_1 \leq \epsilon_5 $$  \hspace{1cm} (103)

where

$$ \epsilon_3 = 10^{-6}, \quad \epsilon_5 = 10^{-6} $$

**9.2. Examples: Quadratic Function, Linear Constraint.**

**Example 1.** The problem is to minimize the function

$$ f = (x + y)^2 + (y + z)^2 $$  \hspace{1cm} (104)

subject to the constraint

$$ x + 2y + 3z - 1 = 0 $$  \hspace{1cm} (105)
This function admits the relative minimum \( f = 0 \) at the point defined by

\[
x = \frac{1}{2}, \quad y = -\frac{1}{2}, \quad z = \frac{1}{2}
\]  

(106)

The nominal point chosen for starting the algorithm is the point of coordinates

\[
x = -4, \quad y = 1, \quad z = 1
\]  

(107)

consistent with (105).

**Example 2.** The problem is to minimize the function

\[
f = (x-1)^2 + (y-z)^2 + (u-w)^2
\]  

(108)

subject to the constraints

\[
x + y + z + u + w - 5 = 0 \\
z - 2(u + w) + 3 = 0
\]  

(109)

The function admits the relative minimum \( f = 0 \) at the point defined by

\[
x = y = z = u = w = 1
\]  

(110)

The nominal point chosen for starting the algorithm is the point of coordinates

\[
x = 3, \quad y = 5, \quad z = -3, \quad y = 2, \quad w = -2
\]  

(111)

consistent with (109).

---

\(^1\)All of the symbols used above denote scalar quantities.
Example 3. The problem is to minimize the function

\[ f = (x-y)^2 + (y-z-2)^2 + (u-1)^2 + (w-1)^2 \]  \hspace{1cm} (112)

subject to the constraints

\[ x + 3y - 4 = 0 \]  \\
\[ z + u - 3w = 0 \]  \hspace{1cm} (113)
\[ y - w = 0 \]

The function admits the relative minimum \( f = 0 \) at the point defined by

\[ x = y = z = u = w = 1 \]  \hspace{1cm} (114)

The nominal point chosen for starting the algorithm is the point of coordinates

\[ x = 5/2, \quad y = 1/2, \quad z = 2, \quad u = -1, \quad w = 1/2 \]  \hspace{1cm} (115)

consistent with (113).

Since the examples 1-3 have linear constraints, the restoration phase requires only one iteration (\( N_r = 1 \)) at the start of the algorithm and needs no more operations to satisfy the constraint during the rest of the computations. Also, because of this, the four versions of the different algorithms behave alike. Further, due to the quadratic nature of the function, the cubic interpolation used will determine the search completely in one cycle (\( N_s = 1 \)). Thus the modes of the algorithms are reduced into one.

The results for the problems 1-3 are shown in Table 1. Convergence history is shown versus the gradient iteration number \( N_g \). Results show that
these problems converge in $n-r$ iterations and all algorithms produce the same sequence of points for convergence. In view of this, Algorithm III should be favored for the minimization of a quadratic function subject to a linear constraint. This is because the simplified algorithm involves the least amount of computational work per iteration.
Table 1. Example 1-3, Algorithm I-III.

<table>
<thead>
<tr>
<th>Example</th>
<th>$N_g$</th>
<th>$f$</th>
<th>$P$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.13 x 10²</td>
<td>0.0</td>
<td>0.55 x 10²</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.23 x 10¹</td>
<td>0.44 x 10⁻³₀</td>
<td>0.20 x 10¹</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>*0.61 x 10⁻³₀</td>
<td>*0.12 x 10⁻²⁹</td>
<td>*0.33 x 10⁻²⁹</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.84 x 10²</td>
<td>0.0</td>
<td>0.62 x 10³</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.78 x 10⁰</td>
<td>0.22 x 10⁻²⁹</td>
<td>0.26 x 10¹</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.10 x 10⁰</td>
<td>0.25 x 10⁻²⁹</td>
<td>0.63 x 10⁰</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>*0.43 x 10⁻³₀</td>
<td>*0.55 x 10⁻²⁹</td>
<td>*0.32 x 10⁻²⁹</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.85 x 10¹</td>
<td>0.0</td>
<td>0.40 x 10²</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.73 x 10⁰</td>
<td>0.13 x 10⁻²⁹</td>
<td>0.37 x 10¹</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>*0.81 x 10⁻³₀</td>
<td>*0.30 x 10⁻²⁹</td>
<td>*0.32 x 10⁻²⁹</td>
<td></td>
</tr>
</tbody>
</table>

²The asterisks in Table 1 mark those numbers which should be zero in an exact solution. Because of computer limitations, the value zero cannot be achieved. Therefore, only the highest error in the solution is indicated.
### 9.3. Examples: Nonquadratic Function, Linear Constraint

**Example 4.** The problem is to minimize the function

\[ f = (x - y)^2 + (y - z)^4 \]  
(116)

subject to the constraint

\[ 2x - y + 3z - 6 = 0 \]  
(117)

This function admits the relative minimum \( f = 0 \) at the point defined by

\[ x = y = z = 1 \]  
(118)

The nominal point chosen for starting the algorithm is the point of coordinates

\[ x = 5, \quad y = 5, \quad z = 3 \]  
(119)

consistent with (117).

**Example 5.** The problem is to minimize the function

\[ f = (x-y)^2 + (z-1)^2 + (u-1)^4 + (w-1)^6 \]  
(120)

subject to the constraints

\[ x + y + z + 4u - 7 = 0 \]  
(121)

\[ z + 5w - 6 = 0 \]

This function admits the relative minimum \( f = 0 \) at the point defined by

\[ x = y = z = u = w = 1 \]  
(122)
The nominal point chosen for starting the algorithm is the point of coordinates

\[ x = 10, \quad y = 7, \quad z = 2, \quad u = 3, \quad w = 3/5 \]  \hspace{1cm} (123)

consistent with (121).

**Example 6.** The problem is to minimize the function

\[ f = (x-y)^2 + (y-z)^2 + (z-u)^4 + (u-w)^4 \]  \hspace{1cm} (124)

subject to the constraints

\begin{align*}
  x + 2y + 3z - 6 &= 0 \\
y + 2z + 3u - 6 &= 0 \\
z + 2u + 3w - 6 &= 0
\end{align*}  \hspace{1cm} (125)

This function admits the relative minimum \( f = 0 \) at the point defined by

\[ x = y = z = u = w = 1 \]  \hspace{1cm} (126).

The nominal point chosen for starting the algorithm is the point of coordinates

\[ x = 35, \quad y = -31, \quad z = 11, \quad u = 5, \quad w = -5 \]  \hspace{1cm} (127)

consistent with (125).

These examples 4-6 have linear constraints and therefore require only one restorative iteration \( (N = 1) \) at the start of the algorithm. During the rest of the computations, no more restorations are needed. This being the case, the four versions of the different algorithms behave alike. Also, linearity in the constraint reduces the modes of the algorithms to only two, that is, (i) restoration
with complete search and (ii) restoration with incomplete search.

The number of gradient steps $N_g$ and number of search steps $N_s$ required for convergence are shown in Table 2. Results indicate that the number of gradient iterations needed for convergence are approximately the same for precise and nonprecise search. On the otherhand, the number of search cycles for complete search are much larger than for incomplete search. Therefore, incomplete search involves less computational work than complete search and may be preferred for practical purposes.
Table 2. Example 4-6.

<table>
<thead>
<tr>
<th>Example</th>
<th>Complete Search</th>
<th>Incomplete Search</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Ng Ng Ng Ng Ng Ng Ng Ng Ng</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>12 32 18 45 7 20 13 13 19 19 7 7</td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>12 32 19 47 7 20 13 13 19 19 7 7</td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>12 32 17 44 7 20 11 11 18 18 7 7</td>
<td></td>
</tr>
</tbody>
</table>
9.4. Examples: Nonquadratic Function, Nonlinear Constraint

Example 7. The problem is to minimize the function

\[ f = (x-y)^2 + (y-z)^4 \]  (128)

subject to the constraint

\[ x(1+y^2) + z^4 - 3 = 0 \]  (129)

This function admits the relative minimum \( f = 0 \) at the point defined by

\[ x = y = z = 1 \]  (130)

The nominal point chosen for starting the algorithm is the point of coordinates

\[ x = -13/4, \quad y = 2, \quad z = 2 \]  (131)

consistent with (129).

Example 8. The problem is to minimize the function

\[ f = (x-y)^2 + (z-1)^2 + (u-1)^4 + (w-1)^6 \]  (132)

subject to the constraints

\[ ux^2 + \sin(u-w) - 1 = 0 \]  (133)

\[ y + z^4 u^2 - 2 = 0 \]

This function admits the relative minimum \( f = 0 \) at the point defined by

\[ x = y = z = u = w = 1 \]  (134)
The nominal point chosen for starting the algorithm is the point of coordinates

\[ x = 1/\sqrt{2}, \quad y = 7/4, \quad z = 1/2, \quad u = 2, \quad w = 2 \]  \hspace{1cm} (135)

consistent with (133).

**Example 9.** The problem is to minimize the function

\[ f = (x-y)^2 + (y-z)^2 + (z-u)^4 + (u-w)^4 \]  \hspace{1cm} (136)

subject to the constraints

\[ x + y^2 + z^3 - 3 = 0 \]
\[ y - z^2 + u - 1 = 0 \]  \hspace{1cm} (137)
\[ xw - 1 = 0 \]

This function admits the relative minimum \( f = 0 \) at the point defined by

\[ x = y = z = u = w = 1 \]  \hspace{1cm} (138)

The nominal point chosen for starting the algorithm is the point of coordinates

\[ x = 2, \quad y = \sqrt{2}, \quad z = -1, \quad u = 2 - \sqrt{2}, \quad w = 1/2 \]  \hspace{1cm} (139)

consistent with (137).

The numerical results pertaining to examples 7-9 are given in Tables 3-11. The tables show the number of gradient steps \( N_g \), the number of restoration steps \( N_r \) and the number of search cycles \( N_s \) needed for convergence for the different versions of the algorithms and their various modes. The results for the various versions of an algorithm are not drastically different. Therefore,
any one of these may be used for practical purposes. For various modes of an
algorithm, the numbers of gradient steps for convergence are of the same order.
However, the numbers of restoration steps and search cycles differ widely from
the ideal case of mode 1 to the simplest case of mode 4. For a more stable al-
gorithm, mode 1 should be preferred to other modes. For efficiency, but per-
haps less stability, mode 4 may be used.

For comparison purposes, a simple gradient method with complete re-
stitution and complete search was programmed to solve the examples 7-9. The
results of the simple gradient algorithm along with the three particular algorithms
for complete search, complete restoration are given in Table 12. Only the gra-
dient steps $N_g$ required for convergence is shown. For the present algorithms,
the largest $N_g$ occurred with different versions is presented. One can clearly
see that the present algorithms converge much faster than the simple gradient
algorithm.
Table 3. Example 7, Algorithm - I

<table>
<thead>
<tr>
<th>Version</th>
<th>Mode</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>9i 16 20</td>
<td>9 16 9</td>
<td>7 6 15</td>
<td>7 6 7</td>
</tr>
<tr>
<td>B</td>
<td>9 17 21</td>
<td>9 15 9</td>
<td>11 11 27</td>
<td>11 11 11</td>
</tr>
<tr>
<td>C</td>
<td>7 13 17</td>
<td>8 16 8</td>
<td>9 9 22</td>
<td>9 9 9</td>
</tr>
<tr>
<td>D</td>
<td>9 15 20</td>
<td>9 16 9</td>
<td>11 11 28</td>
<td>11 11 11</td>
</tr>
</tbody>
</table>

Table 4. Example 7, Algorithm - II

<table>
<thead>
<tr>
<th>Version</th>
<th>Mode</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>11 19 26</td>
<td>11 19 11</td>
<td>11 11 27</td>
<td>11 11 11</td>
</tr>
<tr>
<td>B</td>
<td>11 19 27</td>
<td>11 19 11</td>
<td>11 11 30</td>
<td>11 11 11</td>
</tr>
<tr>
<td>C</td>
<td>11 19 26</td>
<td>11 19 11</td>
<td>11 11 27</td>
<td>11 11 11</td>
</tr>
<tr>
<td>D</td>
<td>11 19 27</td>
<td>11 19 11</td>
<td>11 11 30</td>
<td>11 11 11</td>
</tr>
</tbody>
</table>
Table 5. Example 7, Algorithm - III

<table>
<thead>
<tr>
<th>Mode</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_g$</td>
<td>$N_r$</td>
<td>$N_s$</td>
<td>$N_g$</td>
<td>$N_r$</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>16</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Example 8, Algorithm - I

<table>
<thead>
<tr>
<th>Version</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$N_g$</td>
<td>$N_r$</td>
</tr>
<tr>
<td>A</td>
<td>17</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>D</td>
<td>8</td>
</tr>
</tbody>
</table>
### Table 7. Example 8, Algorithm - II

<table>
<thead>
<tr>
<th>Version</th>
<th>Mode</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ng</td>
<td>N_r</td>
<td>N_s</td>
<td>Ng</td>
<td>N_r</td>
</tr>
<tr>
<td>A</td>
<td>17</td>
<td>20</td>
<td>40</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>B</td>
<td>17</td>
<td>21</td>
<td>41</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>C</td>
<td>17</td>
<td>20</td>
<td>40</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>D</td>
<td>17</td>
<td>21</td>
<td>40</td>
<td>17</td>
<td>22</td>
</tr>
</tbody>
</table>

### Table 8. Example 8, Algorithm - III

| Mode | | | |
|      | 1 | 2 | 3 | 4 |
|      | Ng | N_r| N_s| Ng | N_r| N_s| Ng | N_r| N_s| Ng | N_r| N_s|
|      | 8  | 14 | 17 | 8  | 14 | 8  | 11 | 8  | 23 | 11 | 8  | 11 |
Table 9. Example 9, Algorithm - I

<table>
<thead>
<tr>
<th>Version</th>
<th>Mode</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( N_g )</td>
<td>( N_r )</td>
<td>( N_s )</td>
<td>( N_g )</td>
<td>( N_r )</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>20</td>
<td>31</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>22</td>
<td>30</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>21</td>
<td>31</td>
<td>11</td>
<td>22</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>21</td>
<td>29</td>
<td>10</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 10. Example 9, Algorithm - II

<table>
<thead>
<tr>
<th>Version</th>
<th>Mode</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( N_g )</td>
<td>( N_r )</td>
<td>( N_s )</td>
<td>( N_g )</td>
<td>( N_r )</td>
</tr>
<tr>
<td>A</td>
<td>12</td>
<td>23</td>
<td>35</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>B</td>
<td>12</td>
<td>23</td>
<td>36</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>23</td>
<td>35</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>D</td>
<td>12</td>
<td>23</td>
<td>36</td>
<td>12</td>
<td>22</td>
</tr>
<tr>
<td>Mode</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(N_g)</td>
<td>(N_r)</td>
<td>(N_s)</td>
<td>(N_g)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>21</td>
<td>27</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>21</td>
<td>10</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>8</td>
<td>25</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Table 12. Example 7-9, Mode 1

<table>
<thead>
<tr>
<th>Example</th>
<th>Simple Gradient</th>
<th>Algorithm I</th>
<th>Algorithm II</th>
<th>Algorithm III</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>5821</td>
<td>9</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>20</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>29</td>
<td>11</td>
<td>12</td>
<td>10</td>
</tr>
</tbody>
</table>
10. **Discussion and Conclusions.**

The problem of minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$ is considered. A general quadratically convergent algorithm is presented. The conjugate-gradient algorithm and the variable metric algorithm for constrained function minimization can be obtained as particular cases of the general algorithm. It is shown that, for the case of a quadratic function subject to a linear constraint, all of the particular algorithms behave identically if the one-dimensional search for stepsize is exact. Specifically, they all produce the same sequence of points and they all lead to the constrained minimal point in no more than $n-r$ descent steps.

The algorithms are modified so that they can be employed for the case of a nonquadratic function subject to a linear constraint and the case of a nonquadratic function subject to a nonlinear constraint. In both cases, the algorithms are restarted after every $n-r$ descent steps. However, for the later case, restoration phases must be inserted. This is because, a descent step generally leads to a point violating the constraint. In this connection, one can employ algorithms either with complete restoration or with incomplete restoration.

Tests were made using three particular algorithms. In case of a quadratic function subject to a linear constraint, all of the algorithms behave ideally as predicted. For the case of a nonquadratic function subject to a nonlinear constraint, the algorithms behave satisfactorily, generally converging much faster than the simple gradient algorithm.
References


