INFORMATION TO USERS

This dissertation 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.

University Microfilms
300 North Zeib Road
Ann Arbor, Michigan 48105
A Xerox Education Company
MOSELEY, Phillip Eugene, 1941-
MODIFIED METHOD OF MULTIPLIERS FOR MATHEMATICAL
PROGRAMMING PROBLEMS.

Rice University, Ph.D., 1972
Mathematics

University Microfilms, A XEROX Company, Ann Arbor, Michigan
RICE UNIVERSITY

MODIFIED METHOD OF MULTIPLIERS

FOR MATHEMATICAL PROGRAMMING PROBLEMS

by

PHILLIP E. MOSELEY

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Thesis Director's Signature

Houston, Texas

March, 1972
PLEASE NOTE:

Some pages may have
indistinct print.

Filmed as received.

University Microfilms, A Xerox Education Company
ABSTRACT

Modified Method of Multipliers

for Mathematical Programming Problems

by

PHILLIP E. MOSELEY

In this thesis, the numerical solution of the basic problem of mathematical programming is considered. This is the problem of minimizing a function \( f(x) \) subject to a constraint \( \varphi(x) = 0 \). Here, \( f \) is a scalar, \( x \) an \( n \)-vector, and \( \varphi \) a \( q \)-vector, with \( q < n \).

The approach employed is based on the introduction of a special function which allows one to view the vector \( x \) as unconstrained. Specifically, the function \( f(x) \) is replaced by the augmented penalty function \( W(x, \lambda, k) = f(x) + \lambda^T \varphi(x) + k\varphi'(x)\varphi(x) \). Here, the \( q \)-vector \( \lambda \) is an approximation to the Lagrange multiplier and the scalar \( k > 0 \) is the penalty constant.

Previously, the augmented penalty function \( W(x, \lambda, k) \) was used by Hestenes in his method of multipliers. In Hestenes' version, the method of multipliers involves cycles, in each of which the multiplier and the penalty constant are held constant. After the minimum of the augmented penalty function is achieved in any given cycle, the multiplier \( \lambda \) is updated, while the penalty constant \( k \) is held unchanged.

In this thesis, two modifications of the method of multipliers are presented in order to improve its convergence characteristics. The improved convergence is achieved by (i) increasing the updating frequency so that the number of
iterations in a cycle is $\Delta N = 1$ for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and $\Delta N = n$ for the conjugate-gradient algorithm, (ii) imbedding Hestenes' updating rule for the multiplier $\lambda$ into a one-parameter family and determining the scalar parameter $\beta$ so that the error in the optimum condition is minimized, and (iii) updating the penalty constant $k$ so as to cause some desirable effect in the ordinary-gradient algorithm, the conjugate-gradient algorithm, the modified-quasilinearization algorithm. For the sake of identification, Hestenes' method of multipliers is called Method MM-1, the modification including (i) and (ii) is called Method MM-2, and the modification including (i), (ii), (iii) is called Method MM-3.

Evaluation of the theory is accomplished with seven numerical examples involving small systems ($n \leq 5$ and $q \leq 3$). The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints. Each example is solved with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm, which are employed in conjunction with Methods MM-1, MM-2, and MM-3.

The numerical results show that (a) for given $k$, method MM-2 generally exhibits faster convergence than Method MM-1, (b) in both methods MM-1 and MM-2, the number of iterations for convergence has a minimum with respect to $k$, and (c) the number of iterations for convergence of Method MM-3 is close to the minimum with respect to $k$ of the number of iterations for convergence of Method MM-2. In this light, Method MM-3 has very
desirable characteristics.

Finally, the modified methods of multipliers MM-2 and MM-3 are illustrated for a nonlinear system containing a large number of variables and constraints \((n = 40\) and \(q = 20\)). The ordinary-gradient algorithm is employed, and it is shown that the trend established for small systems also applies to the large system chosen.
ACKNOWLEDGMENT

The author is indebted to Professor Angelo Miele for numerous helpful discussions. He would also like to thank Mr. Alejandro Velasco Levy for checking the numerical accuracy of the examples included. The author also expresses his appreciation for the patience and understanding of his wife, Cecilia, without whom this thesis would have been impossible. Finally, and perhaps of most significance, the author wishes to acknowledge the sacrifice which his boys, Barry and Michael, have made during the preparation of this thesis. This research was supported by the National Science Foundation, Grant No. GP-32453.
TABLE OF CONTENTS

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction .................................................. 1</td>
</tr>
<tr>
<td>2.</td>
<td>Statement of the Problem ................................. 3</td>
</tr>
<tr>
<td>3.</td>
<td>Review of Penalty Function Methods ...................... 5</td>
</tr>
<tr>
<td>4.</td>
<td>Modifications of the Method of Multipliers .......... 12</td>
</tr>
<tr>
<td>5.</td>
<td>Unconstrained Minimization Algorithms .................. 20</td>
</tr>
<tr>
<td>6.</td>
<td>Stepsize Determination ........................................ 26</td>
</tr>
<tr>
<td>7.</td>
<td>Experimental Conditions ...................................... 28</td>
</tr>
<tr>
<td>8.</td>
<td>Numerical Examples: Small Systems ....................... 32</td>
</tr>
<tr>
<td>9.</td>
<td>Numerical Example: Large System .......................... 37</td>
</tr>
<tr>
<td>10.</td>
<td>Results and Conclusions .................................... 42</td>
</tr>
<tr>
<td></td>
<td>Tables ........................................................... 44</td>
</tr>
<tr>
<td></td>
<td>References ..................................................... 47</td>
</tr>
</tbody>
</table>
1. **Introduction**

Over the past several years, considerable work has been done on the numerical solution of the constrained minimization problem. This is the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, $f$ is a scalar, $x$ an $n$-vector, and $\varphi$ a $q$-vector, with $q < n$.

The methods employed are generally based on one of two basic ideas. One approach ensures constraint satisfaction, at least to first order, at the end of each iteration (see, for example, Refs. 1-3). The other approach depends on the construction of a special function which has an unconstrained minimum point coincident with the solution of the original constrained minimization problem. With regard to the latter approach, the standard penalty function method (see, for example, Refs. 4-6) and Hestenes' method of multipliers (Ref. 7) must be mentioned.

Obviously, any approach which permits one to view the vector $x$ as unconstrained is attractive from the point of view of the computational effort required per iteration. Whether it is attractive from the point of view of the overall computational effort required to solve a given problem remains to be seen on the basis of actual numerical experimentation. Particularly for large systems, it might be convenient to avoid the solution of sets of linear equations which characterize, for instance, the algorithms of Refs. 1-3.

In this respect, the method of multipliers must be considered as a prime candidate for the numerical optimization of these systems. Indeed,
the numerical experiments of Refs. 8-9 indicate that the method of multipliers is more stable and exhibits faster convergence than the standard penalty function method. Crucial to the method of multipliers is the manner in which the multiplier $\lambda$ is estimated and the penalty constant $k$ is updated at the beginning of each cycle. These key questions are considered in this thesis, in which the objective is to develop techniques to improve the convergence characteristics of the method of multipliers. The resulting algorithm is called modified method of multipliers.
2. **Statement of the Problem**

The constrained minimization problem consists of minimizing the function

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

subject to the constraint

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

In the above equations, \( f \) is a scalar, \( x \) an \( n \)-vector, and \( \varphi \) a \( q \)-vector, with \( q < n \). Here, all vectors are column vectors. It is assumed that the first and second partial derivatives of the functions \( f \) and \( \varphi \) exist and are continuous and that the constrained minimum exists.

2.1. **First-Order Conditions.** The constrained minimization problem can be recast as that of minimizing the augmented function

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

subject to the constraint (2). The \( q \)-vector \( \lambda \) is referred to as the Lagrange multiplier, and the superscript \( T \) denotes the transpose of a matrix.

From ordinary theory of maxima-minima, it is known that the optimal solution must satisfy the relations

\[ \varphi(x) = 0 \quad , \quad F_x(x, \lambda) = f_x(x) + \varphi_x(x)\lambda = 0 \]  \hspace{1cm} (4)

which are a system of \( q+n \) equations in \( x \) and \( \lambda \). The subscript \( x \) denotes the gradient of a function: in this case, \( f_x \) and \( F_x \) are \( n \)-vectors, and \( \varphi_x \) is an \( nxq \) matrix, defined in such a way that its \( i \)th column is the gradient of the
ith scalar component of $\varphi$ with respect to $x$. Furthermore, it is assumed that the matrix $\varphi_x$ has rank $q$ at the solution point of system (4).

2.2. Approximate Solutions. In general, the system (4) is nonlinear; consequently, approximate methods must be employed. Here, we introduce the scalar performance indexes

$$P(x) = \varphi^T(x) \varphi(x), \quad Q(x, \lambda) = F^T_x(x, \lambda) F_x(x, \lambda)$$

which measure the errors in the constraint and the optimum condition, respectively.

At the solution of the constrained minimization problem, $P = 0$ and $Q = 0$, while $P > 0$ and/or $Q > 0$ for any approximation of the solution. When approximate techniques are employed, they must ultimately lead to values of $x, \lambda$ such that

$$P(x) \leq \varepsilon_1, \quad Q(x, \lambda) \leq \varepsilon_2$$

Alternatively, (6) can be replaced by

$$R(x, \lambda) \leq \varepsilon_3$$

where

$$R(x, \lambda) = P(x) + Q(x, \lambda)$$

denotes the cumulative error in the constraint and the optimum condition.

Here, $\varepsilon_1, \varepsilon_2, \varepsilon_3$ are small, preselected numbers. Note that satisfaction of Ineq. (7) implies satisfaction of Ineqs. (6) if one chooses $\varepsilon_1 = \varepsilon_2 = \varepsilon_3$. 
3. **Review of Penalty Function Methods**

The penalty function method is based on the construction of a function that has an unconstrained minimum point coincident with the solution of the original constrained minimization problem. In this section, two versions of the penalty function method are reviewed: (i) the standard penalty function method and (ii) the method of multipliers. Method (i) is based on the standard penalty function (9) and Method (ii) is based on the augmented penalty function (20).

3.1. **Standard Penalty Function Method.** This method is based on the consideration of the standard penalty function

\[ U(x, k) = f(x) + k\varphi^T(x)\varphi(x) \quad (9) \]

This is obtained by adding to the function \( f(x) \) a term quadratic in the constraint \( \varphi(x) \), \( k > 0 \) being the penalty constant.

The problem of minimizing the function (1) subject to the constraint (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the function (9) with respect to \( x \) for given \( k \). Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of each cycle:

\[ U_x(x, k) = f_x(x) + 2k\varphi_x(x)\varphi(x) = 0 \quad (10) \]

If the penalty constant \( k \) is arbitrary, the vector \( x \) which satisfies Eq. (10) is such that \( \varphi(x) \neq 0 \). However, if one defines the Lagrange multiplier to be
\[ \lambda = 2k \varphi(x) \]  

(11)

Eq. (10) reduces to

\[ F_x(x, \lambda) = f_x(x) + \varphi_x(x)\lambda = 0 \]  

(12)

meaning that the combination of \( x \) and \( \lambda \) thus obtained satisfies exactly the optimum condition.

In order to obtain constraint satisfaction, increasingly larger values of the penalty constant must be employed in successive cycles of the standard penalty function method. In this connection, let \( k_1 = k \) denote the penalty constant of the present cycle and \( k_2 \) denote the penalty constant of the next cycle, with \( k_2 > k_1 \). Because of the jump in \( k \), the standard penalty function increases by the amount

\[ U(x, k_2) - U(x, k_1) = (k_2 - k_1)P(x) \]  

(13)

and the gradient of the standard penalty function increases by the amount

\[ U^T_x(x, k_2)U_x(x, k_2) - U^T_x(x, k_1)U_x(x, k_1) = (k_2 - k_1)^2 P^T_x(x)P_x(x) \]  

(14)

where

\[ P(x) = \varphi^T_x(x)\varphi(x) \quad , \quad P_x(x) = 2\varphi^T_x(x)\varphi(x) \]  

(15)

The positiveness of the right-hand sides of Eqs. (13)-(14) is the key to the mechanism on which the standard penalty function method is based.

After a sufficient number of cycles, the constraint error can be made as small as desired providing the penalty constant has become sufficiently
large. Theoretically speaking, the condition $\alpha(x) = 0$ is desired at convergence. Consequently, the multiplier $\lambda$ defined by Eq. (11) can be identical with the multiplier satisfying Eqs. (4) (which is generally nonzero) only if $k = \infty$.

This result suggest that substantial difficulties might be encountered in the computer implementation of the standard penalty function method, above all if the ordinary-gradient algorithm is employed.

**Numerical Implementation.** From the above considerations, the following outline of the standard penalty function method emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the standard penalty function

$$U(x, k) = f(x) + k\varphi^T(x)\varphi(x)$$

(16)

is minimized with respect to $x$ for given $k$. The minimum of $U(x, k)$ is achieved when the following stopping condition is satisfied:

$$U^T_x(x, k)U_x(x, k) \leq \varepsilon_4$$

(17)

where $\varepsilon_4$ is a small, preselected number.

(c) The solution point of any given cycle is chosen as the starting point of the next cycle of the standard penalty function method.

(d) For the next cycle, a higher value of the penalty constant is selected, one choice being
\[ k_2 = \pi k_1 \]  \hspace{1cm} (18)

where \( \pi > 1 \) is the penalty constant ratio.

(e) After updating the penalty constant, one resets \( k = k_2 \), returns to (b), and continues iteratively.

(f) The algorithm is terminated when the following stopping condition is satisfied:

\[ \varphi^T(x)\varphi(x) + U^T_x(x, k)U_x(x, k) \leq \varepsilon_5 \]  \hspace{1cm} (19)

where \( \varepsilon_5 \) is a small, preselected number.

3.2. Method of Multipliers. This method is based on the consideration of the augmented penalty function

\[ W(x, \lambda, k) = f(x) + \lambda^T \varphi(x) + k\varphi^T(x)\varphi(x) \]  \hspace{1cm} (20)

This is obtained by adding to the penalty function \( U(x, k) \) a term linear in the constraint \( \varphi(x) \), the \( q \)-vector \( \lambda \) being an approximation to the Lagrange multiplier. The use of this function was suggested by Hestenes (Ref. 7) in order to circumvent the numerical difficulties associated with the extremely large values of the penalty constant required by the standard penalty function method.

The problem of minimizing the function (1) subject to the constraint (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the function (20) with respect to \( x \) for given \( \lambda \) and \( k \). Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of each cycle:

\[ W_x(x, \lambda, k) = f_x(x) + \varphi_x(x) + 2k\varphi^T_x(x)\varphi(x) = 0 \]  \hspace{1cm} (21)
and is equivalent to

\[ W_{x} (x, \lambda, k) = f_{x} (x) + \varphi_{x} (x)[\lambda + 2k\varphi(x)] = 0 \] (22)

If the penalty constant \(k\) and the multiplier \(\lambda\) are arbitrary, the vector \(x\) which satisfies Eq. (22) is such that \(\varphi(x) \neq 0\). However, by means of a proper updating rule, a new Lagrange multiplier can be found such that the optimum condition is satisfied exactly. In this connection, let \(\lambda_{1} = \lambda\) denote the Lagrange multiplier of the present cycle and \(\lambda_{2}\) denote the Lagrange multiplier of the next cycle. If \(\lambda_{2}\) is chosen to be

\[ \lambda_{2} = \lambda_{1} + 2k\varphi(x) \] (23)

Eq. (22) reduces to

\[ F_{x} (x, \lambda_{2}) = f_{x} (x) + \varphi_{x} (x)\lambda_{2} = 0 \] (24)

meaning that the combination of \(x\) and \(\lambda_{2}\) thus obtained satisfies exactly the optimum condition.

At the end of any given cycle, whenever the value of the Lagrange multiplier is changed from \(\lambda_{1}\) to \(\lambda_{2}\), the augmented penalty function increases by the amount

\[ W(x, \lambda_{2}, k) - W(x, \lambda_{1}, k) = 2kP(x) \] (25)

and the gradient of the augmented penalty function increases by the amount

\[ W_{x}^T (x, \lambda_{2}, k)W_{x} (x, \lambda_{2}, k) - W_{x}^T (x, \lambda_{1}, k)W_{x} (x, \lambda_{1}, k) = k^2 P_{x}^T (x)P_{x} (x) \] (26)

where \(P(x)\) and \(P_{x} (x)\) are given by Eqs. (15). The positiveness of the right-hand side of Eqs. (25)-(26) is the key to the mechanism on which the method of multipliers is based.
The attention of the reader is called on the essential similarity between Eqs. (13)-(14) and Eqs. (25)-(26). In the standard penalty function method, the drive towards constraint satisfaction is supplied by increasing the penalty constant from cycle to cycle. In the method of multipliers, the drive toward constraint satisfaction is supplied by changing the multiplier from cycle to cycle in accordance with Eq. (23).

While the standard penalty function method requires infinitely large values of the penalty constant at convergence, this is not the case with the method of multipliers. In the latter method, convergence can be achieved with finite values of the penalty constant. All that it is required is that $k$ be sufficiently large.

**Numerical Implementation.** From the above considerations, the following outline of the method of multipliers (Method MM-1) emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the augmented penalty function

$$W(x, \lambda, k) = f(x) + \lambda^T \varphi(x) + k \varphi^T (x) \varphi(x)$$

(27)

is minimized with respect to $x$ for given $\lambda$ and $k$. The minimum of $W(x, \lambda, k)$ is achieved when the following stopping condition is satisfied:

$$W_x^T (x, \lambda, k) W_x (x, \lambda, k) \leq \epsilon_6$$

(28)

where $\epsilon_6$ is a small, preselected number.
(c) The solution point of any given cycle is chosen as the starting point of the next cycle of the method of multipliers.

(d) For the next cycle, the multiplier is updated according to the simple rule

$$\lambda_2 = \lambda_1 + 2k\phi(x)$$  \hspace{1cm} (29)

(e) After updating the multiplier, one resets $\lambda = \lambda_2$, returns to (b), and continues iteratively.

(f) The algorithm is terminated when the following stopping condition is satisfied:

$$\epsilon^T(x)\epsilon(x) + W^T_x(x,\lambda, k)W_x(x,\lambda, k) \leq \epsilon_7$$ \hspace{1cm} (30)

where $\epsilon_7$ is a small, preselected number.

(g) To start the algorithm some assumption concerning the multiplier is necessary. The simplest assumption is

$$\lambda = 0$$  \hspace{1cm} (31)

and is equivalent to stating that the augmented penalty function (20) and the standard penalty function (9) are identical for the first cycle of the algorithm.
4. **Modifications of the Method of Multipliers**

The method of multipliers described in Section 3 has one drawback: a sequence of unconstrained minimization problems must be solved, each possibly requiring a large number of iterations $\Delta N$. Consequently, the total number of iterations for convergence $N_\star = \Sigma(\Delta N)$ may become excessive for practical applications.

In order to accelerate convergence, we explore here several modifications of the method of multipliers. These modifications are obtained by (i) shortening the length of a cycle, (ii) improving the estimate of the multiplier, and (iii) selecting the penalty constant in an appropriate fashion.

4.1. **Updating Frequency.** Let a cycle be defined as any sequence of iterations in which the multiplier $\lambda$ and the penalty constant $k$ are held unchanged, while the vector $x$ is viewed as unconstrained. Let $\Delta N$ denote the number of iterations in a cycle, regardless of whether complete convergence or incomplete convergence is achieved.

To shorten the cycle, we assign *a priori* the value of $\Delta N$. Precisely, we choose $\Delta N$ as the smallest number of iterations compatible with the characteristics of the particular algorithm being considered. Therefore,

$$\Delta N = 1 \quad (32)$$

for the ordinary gradient algorithm and the modified-quasilinearization algorithm and

$$\Delta N = n \quad (33)$$
for the conjugate-gradient algorithm. Therefore, the stopping condition (28) for a cycle is bypassed and is replaced by (32) for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by (33) for the conjugate-gradient algorithm.

4.2. **Multiplier Estimate.** Now, assume that the cycle length $\Delta N$ is defined by Eq. (32) for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by Eq. (33) for the conjugate-gradient algorithm. Also, assume that Hestenes' updating rule (29) is employed at the end of any given cycle. We note that the updated total error $R(x, \lambda_2)$ could be larger or smaller than the total error $R(x, \lambda_1)$ prior to updating. Since this uncertainty is undesirable, Hestenes' updating rule is renounced and is replaced with the more general updating rule

$$\lambda_2 = \lambda_1 + 2\beta \varphi(x) \quad \text{(34)}$$

where $\beta$ is a scalar parameter. This parameter must be determined so as to produce some optimum effect.

For given values of $x$ and $\lambda_1$, a change in $\beta$ causes a change in the updated multiplier $\lambda_2$. Consequently, the updated error in the optimum condition

$$Q(x, \lambda_2) = F_x^T(x, \lambda_2)F_x(x, \lambda_2) \quad \text{(35)}$$

changes, and the total error

$$R(x, \lambda_2) = P(x) + Q(x, \lambda_2) \quad \text{(36)}$$

changes. The optimum value of $\beta$ is that which gives $Q(x, \lambda_2)$, and hence $R(x, \lambda_2)$, the smallest value for given $x$ and $\lambda_1$. 
After combining (34)-(35), we obtain the relation

\[ Q(x, \lambda_1, \beta) = \left[ F_x(x, \lambda_1) + \beta P_x(x) \right]^T \left[ F_x(x, \lambda_1) + \beta P_x(x) \right] \]  

which is quadratic in \( \beta \) and admits the derivatives

\[ Q_{\beta}(x, \lambda_1, \beta) = 2P_x^T(x)[F_x(x, \lambda_1) + \beta P_x(x)] , \quad Q_{\beta\beta}(x, \lambda_1, \beta) = 2P_x^T(x)P_x(x) \]  

the first of which vanishes for

\[ \beta = -P_x^T(x)F_x(x, \lambda_1)/P_x^T(x)P_x(x) \]  

This value of \( \beta \) minimizes \( Q(x, \lambda_1, \beta) \), since \( Q_{\beta\beta}(x, \lambda_1, \beta) > 0 \) provided \( P_x(x) \) does not vanish.

The method of multipliers with cycle stopping condition (28) replaced by (32) or (33) and with multiplier updating rule (29) replaced by

\[ \lambda_2 = \lambda_1 + 2\beta \alpha(x) \]  

is called modified method of multipliers or Method MM-2.

**Remark.** The relationship between Hestenes' updating rule and the present updating rule can be obtained as follows. Let the gradient of the augmented penalty function prior to updating be rewritten as

\[ W_x(x, \lambda_1, k) = F_x(x, \lambda_1) + k P_x(x) \]  

Then, combining (39) and (41) yields the relation

\[ \beta = k - P_x^T(x)W_x(x, \lambda_1, k)/P_x^T(x)P_x(x) \]  

which shows that
\[ \beta = k \]  \hspace{1cm} (43)

if

\[ W_{x}(x, \lambda_{1}, k) = 0 \]  \hspace{1cm} (44)

or if

\[ P_{x}^{T}(x)W_{x}(x, \lambda_{1}, k) = 0 , \quad P_{x}(x) \neq 0 \]  \hspace{1cm} (45)

Clearly, Hestenes' updating rule and the present updating rule are identical if applied at complete convergence, that is, at a point where Ineq. (28) is satisfied.

4.3. **Penalty Constant Estimate.** Now, the question arises as to whether the penalty constant can be selected in such a manner as to improve the convergence characteristics of Method MM-2 or at least stabilize it. In this section, two techniques are presented for updating the penalty constant at the end of a cycle, one suitable for the ordinary-gradient algorithm and one suitable for the conjugate-gradient algorithm and the modified-quasilinearization algorithm. Method MM-2 with Eq. (40) completed by a relation updating the penalty constant is called Method MM-3.

**Ordinary-Gradient Algorithm.** When this algorithm is employed, the multiplier updating rule (40) induces an interesting characteristic; a descent property in the constraint error \( P(x) \). This characteristic is utilized in this section to establish an updating rule for the penalty constant.

While the ordinary-gradient algorithm is described in detail in Section 5, we note here that the displacement \( \Delta x \) leading from the nominal point \( x \) to the
varied point $\bar{x}$ is given by

$$\Delta x = -\alpha[F_{x}(x, \lambda) + kP_{x}(x)]$$  \hspace{1cm} (46)$$

where $\alpha > 0$ is the stepsize. Because of the displacement, the constraint error $P(x)$ changes. To first order, this change is given by

$$\delta P(x) = P_{x}^{T}(x)\Delta x$$  \hspace{1cm} (47)$$

with the implication that

$$\delta P = -\alpha P_{x}(x)[F_{x}(x, \lambda) + kP_{x}(x)]$$  \hspace{1cm} (48)$$

Because of Eqs. (34) and (39), the vectors $P_{x}(x)$ and $F_{x}(x, \lambda)$ are orthogonal, that is,

$$P_{x}^{T}(x) F_{x}(x, \lambda) = 0$$  \hspace{1cm} (49)$$

Therefore, the first-order change of $P(x)$ becomes

$$\delta P(x) = -\alpha k P_{x}^{T}(x) P_{x}(x)$$  \hspace{1cm} (50)$$

and is negative since $\alpha > 0$ and $k > 0$. This result guarantees that

$$P(\bar{x}) < P(x)$$  \hspace{1cm} (51)$$

providing $\alpha$ is sufficiently small.

Among all the values which can be attributed to the penalty constant, we select $k$ in such a way that, on the average, the constraints are satisfied to first order. Thereby, we determine $k$ from the relation

$$\delta P(x) = -2\mu P(x)$$  \hspace{1cm} (52)$$
where

\[ \mu = C \alpha \]  

is the restoration stepsize and \( C \) a constant to be specified. Comparing (50) and (52), we see that the appropriate value of the penalty constant should be

\[ k = 2CP(x)/P_x^T(x)P_x(x) \]  

where, for lack of a better choice, the proportionality constant is set at the level

\[ C = 1 \]  

Since both the numerator and the denominator of the right-hand side of Eq. (54) contain equal powers of \( \alpha(x) \), the penalty constant \( k \) varies slowly along the algorithm and is finite at convergence.

**Conjugate-Gradient Algorithm and Modified-Quasilinearization Algorithm.**

The penalty constant estimate developed for the ordinary-gradient algorithm is based on the descent property (50) and the descent requirement (52). It produces a slowly varying penalty constant (54), which is finite at convergence.

For the conjugate-gradient algorithm and the modified-quasilinearization algorithm, the penalty constant (54) is not desirable for the reasons indicated below. Consider a quadratic function \( f(x) \) and a linear constraint \( \varphi(x) = 0 \). Regardless of the value of \( k \), the augmented penalty function \( W(x, \lambda, k) \) is quadratic in \( x \). Since the optimality condition (22) is satisfied exactly after \( n \) iterations of the conjugate-gradient algorithm and after one iteration of the
modified quasilinearization algorithm, the theoretical optimum value of
the penalty constant should be \( k = \infty \), in that it guarantees simultaneous satisfaction
of the constraint equation \( \varphi(x) = 0 \) at the end of a cycle. In a practical digital
computer, this result means that large values of the penalty constant should
be employed if fast convergence is desired. Paradoxically, therefore,
while the method of multipliers has been generated by Hestenes with the
idea of keeping \( k \) small, large values of \( k \) are desirable in the linear-quadratic
case if one employs the conjugate-gradient algorithm or the modified-
quasilinearization algorithm.

If the function \( f(x) \) is nonquadratic and/or the constraint \( \varphi(x) = 0 \) is
nonlinear, the above reasoning is not true far away from the solution but
is approximately true near the solution. This leads to the concept of programming
\( k \) so as to achieve moderate values far away from the solution and large values
near the solution, even though these large values are not as large as those
needed with the standard penalty function method.

Possible choices of the penalty constant satisfying the above property
are the following:

\[
k_0 = |\lambda^T \varphi(x)| / P(x) \quad (56)
\]

or

\[
k_o = \sqrt{[\varphi^T x (x) \lambda]^T [\varphi^T x (x) \lambda] / P^T x (x) P x (x)} \quad (57)
\]

If Eq. (56) is employed, the order of magnitude of the linear term and the
quadratic term appearing in the augmented penalty function is the same. If
Eq. (57) is employed, the order of magnitude of the gradient of the linear
term and the gradient of the quadratic term appearing in the augmented penalty
function is the same.

It should be emphasized that the use of (56) or (57) is not void of
pitfalls. Thus, a decrease in the constraint error $P(x)$ causes an increase in
$k_o$, which in turn causes a further decrease in $P(x)$. Conversely, an increase
in the constraint error $P(x)$ causes a decrease in $k_o$, which in turn causes
a further increase in $P(x)$. Since this instability is undesirable, we propose
here a modification of (56) or (57) designed to produce a reduction in the
constraint error $P(x)$ and the error in the optimum condition $Q(x, \lambda)$ at
approximately the same rate. This modification is the following:

$$k_2 = \min(k_o, k_1) \text{ if } P(x) \leq Q(x, \lambda)$$

$$k_2 = \max(k_o, k_1) \text{ if } P(x) > Q(x, \lambda)$$  \hspace{1cm} (58)

where $k_o$ is given by (56) or (57).
5. **Unconstrained Minimization Algorithms**

In this section, the unconstrained minimization algorithms employed to compute the displacement vector $\Delta x$ in connection with Methods MM-2 and MM-3 are described. They are the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. All of these algorithms make use of the augmented penalty function

$$W(x, \lambda, k) = F(x, \lambda) + kP(x)$$  \hspace{1cm} (59)

where

$$F(x, \lambda) = f(x) + \lambda^T \varphi(x), \quad P(x) = \varphi^T(x)\varphi(x)$$  \hspace{1cm} (60)

and are employed with this understanding: in each cycle of $\Delta N$ iterations, the multiplier $\lambda$ and the penalty constant $k$ are held unchanged, and the vector $x$ is viewed as unconstrained.

5.1. **Ordinary-Gradient Algorithm.** Let $x$ denote the nominal point, $\bar{x}$ the varied point, $\Delta x$ the displacement leading from the nominal point to the varied point, and $\alpha$ the stepsize. With this understanding, the ordinary-gradient algorithm is represented by

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\lambda$$

$$P_x(x) = 2\varphi_x(x)\varphi(x)$$

$$W_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x)$$

$$p = W_x(x, \lambda, k)$$

$$\Delta x = -\alpha p$$

$$\bar{x} = x + \Delta x$$
For given nominal point $x$, multiplier $\lambda$, and penalty constant $k$, Eqs. (61) constitute a complete iteration leading to the varied point $\tilde{x}$, providing one specifies the stepsize $\alpha$.

**Descent Properties.** To first order, the changes in the functions $W(x, \lambda, k)$ and $P(x)$ are given by

$$
\delta W(x, \lambda, k) = W^T_x(x, \lambda, k) \Delta x, \quad \delta P(x) = P^T_x(x) \Delta x
$$

and, in the light of (49) and (61), become

$$
\delta W(x, \lambda, k) = -\alpha W^T_x(x, \lambda, k) W_x(x, \lambda, k), \quad \delta P(x) = -\alpha k P^T_x(x) P_x(x)
$$

(63)

Since the right-hand sides of (63) are negative, the following inequalities can be enforced for $\alpha$ sufficiently small:

$$
W(\tilde{x}, \lambda, k) < W(x, \lambda, k), \quad P(\tilde{x}) < P(x)
$$

(64)

While enforcement of (64-1) is mandatory, enforcement of (64-2) is optional, but can be used in order to give stability to the ordinary-gradient algorithm.

5.2. **Conjugate-Gradient Algorithm.** Let $x$ denote the nominal point, $\hat{x}$ the previous point, $\tilde{x}$ the varied point, $\Delta x$ the displacement leading from the nominal point to the varied point, $p$ the present search direction, $\hat{p}$ the previous search direction, $\gamma$ the directional coefficient, and $\alpha$ the stepsize. With this understanding, the conjugate-gradient algorithm is represented by

$$
F_x(x, \lambda) = f_x(x) + \phi_x(x) \lambda
$$

(65-1)

$$
P_x(x) = 2\phi_x(x) \psi(x)
$$

(65-2)
\[ W_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x) \]  
\[ \gamma = W_x^T(x, \lambda, k)W_x(x, \lambda, k)/W_x^T(x, \lambda, k)W_x(x, \lambda, k) \]  
\[ p = W_x(x, \lambda, k) + \gamma \hat{p} \]  
\[ \Delta x = -\alpha p \]  
\[ \hat{x} = x + \Delta x \]  

For given nominal point \( x \), multiplier \( \lambda \), directional coefficient \( \gamma \), and penalty constant \( k \), Eqs. (65) constitute a complete iteration leading to the varied point \( \hat{x} \), providing one specifies the stepsize \( \alpha \). For the first iteration of a cycle, Eq. (65-4) is bypassed and is replaced by \( \gamma = 0 \).

**Descent Properties.** To first order, the changes in the function \( W(x, \lambda, k) \) and \( P(x) \) are given by

\[ \delta W(x, \lambda, k) = W_x^T(x, \lambda, k)\Delta x \quad \delta P(x) = P_x^T(x)\Delta x \]  

and, in the light of (65), become

\[ \delta W(x, \lambda, k) = -\alpha W_x^T(x, \lambda, k)W_x(x, \lambda, k) + \gamma \hat{p} \]  
\[ \delta P(x) = -\alpha P_x^T(x)\left[F_x(x, \lambda) + kP(x) + \gamma \hat{p}\right] \]  

For the first iteration of a cycle, relation (49) must be applied in conjunction with \( \gamma = 0 \), leading to

\[ \delta W(x, \lambda, k) = -\alpha W_x^T(x, \lambda, k)W_x(x, \lambda, k) \quad \delta P(x) = -\alpha kP_x^T(x)P_x(x) \]  

For subsequent iterations, relation (49) does not hold and \( \gamma \neq 0 \). However, since the previous stepsize is optimized, the following orthogonality relation can be invoked:
with the consequence that

\[ \delta W(x, \lambda, k) = -\alpha W^T(x, \lambda, k) W_x(x, \lambda, k), \quad \delta P(x) = -\alpha P^T(x)[F_x(x, \lambda) + kP_x(x) + \gamma \hat{p}] \]  \( \text{(70)} \)

Inspection of (68) and (70) shows that the descent property on the augmented penalty function

\[ W(\bar{x}, \lambda, k) < W(x, \lambda, k) \]  \( \text{(71)} \)

can be enforced for all iterations of a cycle regardless of the value of \( k \).

On the other hand, the descent property on the constraint error

\[ P(\bar{x}) < P(x) \]  \( \text{(72)} \)

can be enforced for the first iteration of a cycle regardless of the value of \( k \) and for subsequent iterations only if \( k \) is sufficiently large.

5.3. **Modified-Quasilinearization Algorithm.** Let \( x \) denote the nominal point, \( \bar{x} \) the varied point, \( \Delta x \) the displacement leading from the nominal point to the varied point, \( p \) the search direction, \( \rho = \pm 1 \) the direction factor, and \( \alpha \) the stepsize. With this understanding, the modified-quasilinearization algorithm is represented by

\[ F_x(x, \lambda) = f_x(x) + \omega_x(x)\lambda \]  \( \text{(73-1)} \)

\[ P_x(x) = 2\varphi_x(x)p(x) \]  \( \text{(73-2)} \)

\[ W_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x) \]  \( \text{(73-3)} \)
\( F_{xx}(x, \lambda) = f_{xx}(x) + \varphi_{xx}(x)\lambda \)  \hspace{1cm} (73-4)

\( P_{xx}(x) = 2[\varphi_{xx}(x)\varphi(x) + \varphi(x)\varphi_{xx}(x)] \)  \hspace{1cm} (73-5)

\( W_{xx}(x, \lambda, k) = F_{xx}(x, \lambda) + kP_{xx}(x) \)  \hspace{1cm} (73-6)

\( W_{xx}(x, \lambda, k)A + W_{x}(x, \lambda, k) = 0 \)  \hspace{1cm} (73-7)

\( \rho = \text{sign}[W_{x}^{T}(x, \lambda, k)A] \)  \hspace{1cm} (73-8)

\( p = \rho A \)  \hspace{1cm} (73-9)

\( \Delta x = -\alpha p \)  \hspace{1cm} (73-10)

\( \bar{x} = x + \Delta x \)  \hspace{1cm} (73-11)

For given nominal point \( x \), multiplier \( \lambda \), and penalty constant \( k \), Eqs. (73) constitute a complete iteration leading to the varied point \( \bar{x} \), providing one specifies the stepsize \( \alpha \).

**Descent Properties.** To first order, the changes in the functions \( W(x, \lambda, k) \) and \( P(x) \) are given by

\[ \delta W(x, \lambda, k) = W_{x}^{T}(x, \lambda, k)\Delta x \quad , \quad \delta P(x) = P_{x}^{T}(x)\Delta x \]  \hspace{1cm} (74)

and, in the light of (73), become

\[ \delta W(x, \lambda, k) = -\alpha \text{sign}[W_{x}^{T}(x, \lambda, k)A] W_{x}^{T}(x, \lambda, k)A \]  \hspace{1cm} (75)

\[ \delta P(x) = -\alpha \text{sign}[F_{x}^{T}(x, \lambda)A + kP_{x}^{T}(x)A] P_{x}^{T}(x)A \]

Inspection of (75) shows that the descent property on the augmented penalty function
\[ W(\bar{x}, \lambda, k) < W(x, \lambda, k) \]  

(76)

can be enforced regardless of the value of \( k \). On the other hand, the descent property on the constraint error

\[ P(\bar{x}) < P(x) \]  

(77)

can be enforced for any \( k \) if

\[ F^T_x(x, \lambda)A/P^T_x(x)A > 0 \]  

(78)

and only for \( k \) sufficiently large if

\[ F^T_x(x, \lambda)A/P^T_x(x)A < 0 \]  

(79)
6. **Stepsize Determination**

For all of the previous algorithms, the position vector at the end of any step can be written as

\[ \tilde{x} = x - \alpha p \]  \hfill (80)

where \( p \) denotes the search direction. This is a one-parameter family of varied points \( \tilde{x} \), for which the augmented penalty function (59) takes the form

\[ W(\tilde{x}, \lambda, k) = W(x - \alpha p, \lambda, k) = \tilde{W}(\alpha) \]  \hfill (81)

A precise search to be employed with the conjugate-gradient algorithm and an approximate search to be employed with the ordinary-gradient algorithm and the modified-quasilinearization algorithm are described below.

**Precise Search.** We now assume that a minimum of \( \tilde{W}(\alpha) \) exists. Then, we employ some one-dimensional search scheme (for instance, quadratic interpolation, cubic interpolation, or quasilinearization) to determine the value of \( \alpha \) for which

\[ \tilde{W}_\alpha(\alpha) = 0 \]  \hfill (82)

Ideally, this procedure should be used iteratively until the modulus of the slope satisfies any of the following inequalities:

\[ |\tilde{W}_\alpha(\alpha)| \leq \varepsilon_8 \quad \text{or} \quad |\tilde{W}_\alpha(\alpha)| \leq \varepsilon_9 |\tilde{W}_\alpha(0)| \]  \hfill (83)

where \( \varepsilon_8 \) and \( \varepsilon_9 \) are small, preselected numbers. Of course, the value of \( \alpha \) satisfying Ineq. (83) must be such that
\[ \bar{W}(\alpha) < \bar{W}(0) \quad (84) \]

**Approximate Search.** Since the rigorous determination of \( \alpha \) might require excessive computing time, one might renounce solving Eq. (82) with a particular degree of precision and determine the stepsize in a noniterative fashion. For instance, one might employ a bisection procedure on \( \alpha \), starting from a reference value \( \alpha = \alpha_R \), until satisfaction of Ineq. (84) occurs. For the ordinary gradient algorithm, the reference stepsize \( \alpha_R \) can be chosen to be the first optimum value of \( \alpha \) supplied by the search procedure. For the modified-quasilinearization algorithm, the reference stepsize \( \alpha_R \) can be chosen to be \( \alpha_R = 1 \).

**Remark.** Optionally, Ineq. (84) can be completed by the additional inequality

\[ \tilde{P}(\alpha) < \tilde{P}(0) \quad (85) \]

designed to give stability to the algorithm. Inequality (85) can be enforced in the ordinary-gradient algorithm for any \( k \). In the conjugate-gradient algorithm and the modified-quasilinearization algorithm, Ineq. (85) can be enforced only for \( k \) sufficiently large.
7. **Experimental Conditions**

In order to evaluate the theory, several numerical examples were explored. Each example was solved with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm, which were employed in conjunction with Methods MM-1, MM-2, and MM-3. All of the algorithms were programmed in FORTRAN IV, and the numerical results were obtained using a Burroughs B-5500 computer and double-precision arithmetic.

**Starting Point of the Algorithm.** For all of the examples except Example 9.1, the nominal point chosen to start the algorithm was defined by

\[ x_1 = x_2 = \ldots = x_n = 2 \]  \hspace{1cm} (86)

where \( n \) denotes the dimension of the vector \( x \).

**Convergence of the Algorithm.** Convergence of an algorithm was defined through the inequality

\[ \phi^T(x)\varphi(x) + W_x^T(x, \lambda, k)W_x(x, \lambda, k) \leq 10^{-6} \]  \hspace{1cm} (97)

for the ordinary-gradient algorithm and the inequality

\[ \phi^T(x)\varphi(x) + W_x^T(x, \lambda, k)W_x(x, \lambda, k) \leq 10^{-12} \]  \hspace{1cm} (88)

for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.
Nonconvergence of the Algorithm. Conversely, nonconvergence of an algorithm was defined by means of the inequalities

\[ N > 1000 \quad \text{for the ordinary-gradient algorithm} \]

(a) \[ N > 200 \quad \text{for the conjugate-gradient algorithm} \quad (89) \]

\[ N > 100 \quad \text{for the modified-quasilinearization algorithm} \]

or

(b) \[ N_s \geq 20 \quad (90) \]

or

(c) \[ M \geq 0.4 \times 10^{69} \quad (91) \]

Here, \( N \) is the iteration number, \( N_s \) is the number of bisections of the stepsize \( \alpha \) required to satisfy Ineq. (84) and optionally Ineq. (85), and \( M \) is the modulus of any of the quantities employed in the algorithm. Satisfaction of Ineq. (89) indicates divergence or extreme slowness of convergence; satisfaction of Ineq. (90) indicates extreme smallness of the displacement \( \Delta x \); and satisfaction of Ineq. (91) indicates exponential overflow. Each of these situations is undesirable.

Convergence of a Cycle. When Method MM-1 was employed, convergence of a cycle was defined through the inequality

\[ \mathbf{W}_x^T(x, \lambda, k)\mathbf{W}_x(x, \lambda, k) \leq 10^{-6} \quad (92) \]

for the ordinary-gradient algorithm and the inequality

\[ \mathbf{W}_x^T(x, \lambda, k)\mathbf{W}_x(x, \lambda, k) \leq 10^{-12} \quad (93) \]

for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.
When Methods MM-2 and MM-3 were employed, convergence of a cycle was defined by

\[ \Delta N = 1 \]  \hspace{1cm} (94)

for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by

\[ \Delta N = \eta \]  \hspace{1cm} (95)

or (93) for the conjugate-gradient algorithm.

**Search Technique.** For the ordinary-gradient algorithm, an approximate search was employed. This consisted of one-step, corrected quasilinearization, followed by a bisection process until the inequality

\[ \tilde{W}(\alpha) < \tilde{W}(0) \]  \hspace{1cm} (96)

was satisfied. When Method MM-3 was employed, two versions of the ordinary-gradient algorithm were considered. In the first version, the additional inequality

\[ \tilde{P}(\alpha) < \tilde{P}(0) \]  \hspace{1cm} (97)

was bypassed. In the second version, Ineq. (97) was accounted for whenever \( \tilde{P}(0) > 10^{-6} \).

For the conjugate-gradient algorithm, a precise search was employed. This consisted of multistep, corrected quasilinearization such that, in any given step, the inequality

\[ \tilde{W}(\alpha) < \tilde{W}(0) \]  \hspace{1cm} (98)
was satisfied, where \( \alpha_0 \) is the nominal stepsize and \( \alpha \) is the varied stepsize. The search was terminated when the following stopping condition was satisfied:

\[
\bar{W}_\alpha^2(\alpha) \leq \bar{W}_\alpha^2(0) \times 10^{-6} \tag{99}
\]

For the modified-quasilinearization algorithm, an approximate search was employed. This consisted of assigning the value

\[
\alpha = 1 \tag{100}
\]

to the stepsize, followed by a bisection process until Ineq. (96) was satisfied.
8. **Numerical Examples: Small Systems**

In this section, seven numerical examples involving small systems (n ≤ 5 and q ≤ 3) are described. The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints.

**Example 8.1.** Consider the problem of minimizing the function

\[
f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \tag{101}
\]

subject to the constraints

\[
x_1 + 3x_2 = 0, \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0 \tag{102}
\]

This function admits the relative minimum \( f = 4.0930 \) at the point defined by

\[
x_1 = -0.7674, \quad x_2 = 0.2558, \quad x_3 = 0.6279, \quad x_4 = -0.1162, \quad x_5 = 0.2558 \tag{103}
\]

and

\[
\lambda_1 = 2.0465, \quad \lambda_2 = 2.2325, \quad \lambda_3 = -5.9534 \tag{104}
\]

**Example 8.2.** Consider the problem of minimizing the function

\[
f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \tag{105}
\]

subject to the constraint

\[
x_1(x_1^2 + 2) + x_3^4 - 4 - 3/2 = 0 \tag{106}
\]

This function admits the relative minimum \( f = 0.3256 \times 10^{-1} \) at the point defined by
\[ x_1 = 1.1048, \ x_2 = 1.1966, \ x_3 = 1.5352 \] (107)

and

\[ \lambda_1 = -0.1072 \times 10^{-1} \] (108)

**Example 8.3.** Consider the problem of minimizing the function

\[ f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6 \] (109)

subject to the constraints

\[ \frac{x_1 x_4 + \sin(x_4 - x_5)}{2} - \sqrt{2} = 0, \ x_2 + x_4^2 = 8 - \sqrt{2} = 0 \] (110)

This function admits the relative minimum \( f = 0.2415 \) at the point defined by

\[ x_1 = 1.1661, \ x_2 = 1.1821, \ x_3 = 1.3802, \ x_4 = 1.5060, \ x_5 = 0.6109 \] (111)

and

\[ \lambda_1 = -0.8553 \times 10^{-1}, \ \lambda_2 = -0.3187 \times 10^{-1} \] (112)

**Example 8.4.** Consider the problem of minimizing the function

\[ f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \] (113)

subject to the constraints

\[ x_1 + x_2^2 + x_3^2 - 2 - 3/2 = 0, \ x_2 - x_3^2 + x_4 + 2 - 2/2 = 0, \ x_1 x_5 - 2 = 0 \] (114)

This function admits the relative minimum \( f = 0.7877 \times 10^{-1} \) at the point
defined by

\[ x_1 = 1.1911, \; x_2 = 1.3626, \; x_3 = 1.4728, \; x_4 = 1.6350, \; x_5 = 1.6790 \] (115)

and

\[ \lambda_1 = -0.3882 \times 10^{-1}, \; \lambda_2 = -0.1672 \times 10^{-1}, \; \lambda_3 = -0.2879 \times 10^{-3} \] (116)

Example 8.5. Consider the problem of minimizing the function

\[ f = 0.01(x_1 - 1)^2 + (x_2 - x_2^2)^2 \] (117)

subject to the inequality constraint

\[ x_1 \leq -1 \] (118)

Introduce the auxiliary variable \( x_3 \) defined by

\[ x_1 + x_3^2 + 1 = 0 \] (119)

Then, the previous problem can be recast as that of minimizing the function (117) subject to the equality constraint (119). The function (117) admits the relative minimum \( f = 0.04 \) at the point defined by

\[ x_1 = -1, \; x_2 = 1, \; x_3 = 0 \] (120)

and

\[ \lambda_1 = 0.04 \] (121)

Example 8.6. Consider the problem of minimizing the function
\[ f = -x_1 \]

subject to the inequality constraints

\[ x_2 \geq x_1^3 , \quad x_2 \leq x_1^2 \]

(123)

Introduce the auxiliary variables \( x_3 \) and \( x_4 \) defined by

\[ x_2 - x_1^3 - x_2^3 = 0 , \quad x_1^2 - x_2^4 - x_2^2 = 0 \]

(124)

Then, the previous problem can be recast as that of minimizing the function (122) subject to the equality constraints (124). The function (122) admits the relative minimum \( f = -1 \) at the point defined by

\[ x_1 = 1 , \quad x_2 = 1 , \quad x_3 = 0 , \quad x_4 = 0 \]

(125)

and

\[ \lambda_1 = -1 , \quad \lambda_2 = -1 \]

(126)

Example 8.7. Consider the problem of minimizing the function

\[ f = \log x_3 - x_2 \]

(127)

subject to the equality constraint

\[ x_3^2 + x_2^2 - 4 = 0 \]

(128)

and the inequality constraint

\[ x_3 \geq 1 \]

(129)
Introduce the auxiliary variable $x_1$ defined by

$$ x_3 = 1 + x_1^2 $$

Then, the previous problem can be recast as that of minimizing the function

$$ f = \log(1 + x_1^2) - x_2 $$

subject to the equality constraint

$$ (1 + x_1^2)^2 + x_2^2 - 4 = 0 $$

Note that $x_3$ has been eliminated from the problem and can be computed 
a posteriori with (130). The function (131) admits the relative minimum $f = -\sqrt{3}$
at the point defined by

$$ x_1 = 0 , \quad x_2 = \sqrt{3} , \quad x_3 = 1 $$

and

$$ \lambda_1 = 1/2\sqrt{3} $$
9. **Numerical Example: Large System**

In this section, a nonlinear system containing a large number of variables and constraints (n = 40 and q = 20) is considered. This is a discretized optimal control problem involving one state variable, one control variable, and fixed endpoints.

**Example 9.1.** Consider the problem of minimizing the integral

\[
I = \int_0^1 x^2 u^2 \, dt \tag{135}
\]

subject to the differential constraint

\[
\dot{x} = u x^2 \tag{136}
\]

and the boundary conditions

\[
x(0) = 1 \quad , \quad x(1) = e \tag{137}
\]

where e = 2.71828. Here, x(t) is the state variable and u(t) is the control variable. This optimal control problem has the analytical solution

\[
x(t) = \exp(t) \quad , \quad u(t) = \exp(-t) \quad , \quad \lambda(t) = 2 \exp(-t) \tag{138}
\]

which yields the relative minimum I = 1 for the integral (135).

This problem can be transformed into a mathematical programming problem by employing a discrete-variable approximation (Ref. 10). First, divide the interval [0, 1] into \(\rho\) subintervals (\(\rho\) even) of equal length

\[
\Delta t = 1/\rho \tag{139}
\]
which yield \( n + 1 \) equally spaced points

\[
t_i = i/\rho, \quad i = 0, 1, \ldots, \rho
\]  

(140)

Now, for each \( t_i \), let the functions \( x(t) \) and \( u(t) \) take on the discrete values

\[
x(t_i) = x_i, \quad u(t_i) = u_i, \quad i = 0, 1, \ldots, \rho
\]  

(141)

and let some finite-difference representation be used for the integral (135) and the differential constraint (136). In this respect, Simpson's rule is utilized for the integral (135) and, as suggested in Ref. 10, the differential equation (136) is approximated by a set of difference equations according to Adams' interpolation scheme.

The resulting mathematical programming problem can be formulated as follows. Minimize the function

\[
f = \frac{1}{3} \left[ (x_0^2 u_0^2)^2 + 4 \sum_{j=1}^{\rho/2} (x_{2j-1}^2 u_{2j-1})^2 + 2 \sum_{j=1}^{\rho/2-1} (x_{2j}^2 u_{2j})^2 + (x_{\rho/\rho}^2 u_{\rho/\rho})^2 \right]
\]  

(142)

subject to the constraints

\[
x_1 - x_0 - (1/2\rho)(x_0^2 u_0^2 + x_1^2 u_1^2) = 0
\]  

(143)

\[
x_2 - x_1 - (1/12\rho)(-x_0^2 u_0^2 + 8x_1^2 u_1^2 + 5x_2^2 u_2^2) = 0
\]

and

\[
x_{j+1} - x_j - (1/24\rho)(x_j^2 u_{j-2}^2 - 5x_j^2 u_{j-1}^2 + 19x_j^2 u_{j+1}^2 + 9x_{j+1}^2 u_{j+1}^2) = 0, \quad j = 2, 3, \ldots, \rho - 1
\]  

(144)
and the boundary conditions

\[ x_0 = 1, \quad x_\rho = e \]  

(145)

Upon substituting (145) into (142)-(144), eliminating \( x_0 \) and \( x_\rho \), and assuming \( \rho = 20 \), we see that this mathematical programming problem is characterized by \( n = 40 \) variables and \( q = 20 \) constraints.

The nominal point chosen to start the algorithm is the point of coordinates

\[ x_i = i(e - 1)/\rho + 1, \quad u_i = 0, \quad i = 0, 1, \ldots, \rho \]  

(146)

where \( \rho = 20 \). For this nominal point, problem (142)-(145) was solved employing Methods MM-2 and MM-3 in conjunction with the ordinary-gradient algorithm.

**Search Technique.** For a large system, the computation of second derivatives requires excessive computing time. Therefore, a quadratic interpolation scheme was employed in the stepsize selection.

First, let the Chebyshev norm of the search direction \( p \) be defined as

\[ ||p|| = \max_i |p_i|, \quad i = 1, 2, \ldots, n \]  

(147)

Next, introduce the normalized search direction \( q \) and the normalized stepsize \( \theta \), which are defined as

\[ q = p/||p||, \quad \theta = \alpha||p|| \]  

(148)

With this understanding, the varied position vector (80) can be rewritten as

\[ \bar{x} = x - \theta q \]  

(149)
The following quadratic interpolation formula was employed:

\[
\tilde{W}(\theta) = A\theta^2 + B\theta + C
\] (150)

where the coefficients \( A, B, C \) are determined through three equally spaced values of the stepsize \( \theta_1, \theta_2, \theta_3 \) such that

\[
\theta_1 < \theta_2 < \theta_3
\] (151)

The stepsizes \( \theta_1, \theta_2, \theta_3 \) were chosen so that the value of \( \theta \) for which the minimum of \( \tilde{W}(\theta) \) occurs is bounded between \( \theta_1 \) and \( \theta_3 \). This is the case if \( \theta_1, \theta_2, \theta_3 \) satisfy the inequalities

\[
\tilde{W}(\theta_2) < \tilde{W}(\theta_1) \text{ and } \tilde{W}(\theta_2) < \tilde{W}(\theta_3)
\] (152)

The selection of \( \theta_1, \theta_2, \theta_3 \) is made employing either of the following sequences:

\[
\theta/\tau \in \{0, 1, 2, 4, 8, \ldots\} \quad (153)
\]

\[
\theta/\tau \in \{0, 1/2, 1/4, 1/8, \ldots\} \quad (154)
\]

where \( \tau \) is a scaling factor. The first of these sequences is employed if

\[
\tilde{W}(\tau) < \tilde{W}(0)
\] (155)

and the second is employed if

\[
\tilde{W}(\tau) > \tilde{W}(0)
\] (156)
In both cases $\theta_1 = 0$, and $\theta_2$ and $\theta_3$ are the last two elements for which satisfaction of Ineqs. (152) occurs.

Once $\theta_1$, $\theta_2$, $\theta_3$ have been chosen, the coefficients $A$, $B$, $C$ of the quadratic function (150) can be computed. With these coefficients known, the reference stepsize $\theta_R$ is deduced from

$$\tilde{W}_p(\theta_R) = 0 \quad (157)$$

that is, from

$$\theta_R = -B/2A \quad (158)$$

or

$$\theta_R = \theta_1 + \frac{1}{2} \tau[3\tilde{W}(\theta_1) - 4\tilde{W}(\theta_2) + \tilde{W}(\theta_3)]/[\tilde{W}(\theta_1) - 2\tilde{W}(\theta_2) + \tilde{W}(\theta_3)] \quad (159)$$

Remark. The computations presented here were done employing a scaling factor $\tau = 1$. However, from the numerical results, it became apparent that the number of function evaluations would have been reduced, had $\tau$ been chosen as the stepsize of the previous iteration.
10. **Results and Conclusions**

The examples described in Section 8 were solved with Hestenes' method of multipliers (Method MM-1) and the modified method of multipliers (Methods MM-2 and MM-3) according to the experimental conditions outlined. These methods were employed in conjunction with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. For Methods MM-1 and MM-2, several values of the penalty constant, ranging between $10^{-2}$ and $10^{2}$, were considered.

The numerical results are presented in Tables 1-10, where the number of iterations required for convergence $N_*$ is shown. Tables 1-4 refer to the ordinary-gradient algorithm, Tables 5-7 refer to the conjugate-gradient algorithm, and Tables 8-10 refer to the modified-quasilinearization algorithm.

Comparison of Methods MM-1 and MM-2 shows that, for given $k$, Method MM-2 generally exhibits faster convergence than Method MM-1. For both Methods MM-1 and MM-2, the number of iterations for convergence has a minimum with respect to $k$.

Concerning Method MM-3, the number of iterations for convergence is close to the minimum with respect to $k$ of the number of iterations for convergence of Method MM-2. In this light, Method MM-3 has very desirable characteristics.

For Method MM-3 employed in conjunction with the ordinary-gradient algorithm, numerical results were obtained in two ways: (i) not enforcing the descent property in the constraint error, and (ii) enforcing this property.
The results show that it is desirable to enforce the descent property on \( P \).

To further illustrate the convergence properties of Methods MM-2 and MM-3, the example of Section 9 was considered in conjunction with the ordinary-gradient algorithm. For Method MM-2, three values of the penalty constant were considered, namely, \( k = 0.1 \), \( k = 1 \), and \( k = 10 \). For \( k = 1 \), convergence in \( N_\ast = 960 \) iterations was achieved. For \( k = 0.1 \) and \( k = 10 \) nonconvergence (a) occurred. Concerning Method MM-3, convergence in \( N_\ast = 644 \) iterations was achieved in Case (i) and \( N_\ast = 613 \) iterations was achieved in Case (ii). In conclusion, the numerical results for the large system of Section 9 follow the same trend established for the small systems of Section 8.
Table 1. Method MM-1, ordinary-gradient algorithm, number of iterations \( N_*. \)

<table>
<thead>
<tr>
<th>Examples</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
<th>8.6</th>
<th>8.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-2})</td>
<td>(a)</td>
<td>24</td>
<td>553</td>
<td>572</td>
<td>(a)</td>
<td>(a)</td>
<td>66</td>
</tr>
<tr>
<td>(10^{-1})</td>
<td>649</td>
<td>73</td>
<td>847</td>
<td>191</td>
<td>549</td>
<td>(a)</td>
<td>16</td>
</tr>
<tr>
<td>(10^{0})</td>
<td>350</td>
<td>646</td>
<td>(a)</td>
<td>431</td>
<td>58</td>
<td>946</td>
<td>18</td>
</tr>
<tr>
<td>(10^{1})</td>
<td>863</td>
<td>(a)</td>
<td>(a)</td>
<td>377</td>
<td>761</td>
<td>762</td>
<td>22</td>
</tr>
<tr>
<td>(10^{2})</td>
<td>(a)</td>
<td>(a)</td>
<td>(a)</td>
<td>(a)</td>
<td>92</td>
<td>746</td>
<td>905</td>
</tr>
</tbody>
</table>

Table 2. Method MM-2, ordinary-gradient algorithm, number of iterations \( N_*. \)

<table>
<thead>
<tr>
<th>Examples</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
<th>8.6</th>
<th>8.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-2})</td>
<td>(a)</td>
<td>14</td>
<td>135</td>
<td>101</td>
<td>138</td>
<td>536</td>
<td>55</td>
</tr>
<tr>
<td>(10^{-1})</td>
<td>330</td>
<td>21</td>
<td>433</td>
<td>94</td>
<td>39</td>
<td>322</td>
<td>15</td>
</tr>
<tr>
<td>(10^{0})</td>
<td>82</td>
<td>661</td>
<td>(a)</td>
<td>306</td>
<td>33</td>
<td>278</td>
<td>17</td>
</tr>
<tr>
<td>(10^{1})</td>
<td>301</td>
<td>998</td>
<td>(a)</td>
<td>(a)</td>
<td>126</td>
<td>129</td>
<td>30</td>
</tr>
<tr>
<td>(10^{2})</td>
<td>(a)</td>
<td>(a)</td>
<td>(a)</td>
<td>(a)</td>
<td>198</td>
<td>(a)</td>
<td>389</td>
</tr>
</tbody>
</table>

Table 3. Method MM-3, ordinary-gradient algorithm, number of iterations \( N_*. \)
Descent property on \( P \) not enforced.

<table>
<thead>
<tr>
<th>Examples</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
<th>8.6</th>
<th>8.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eqs. (54)-(55)</td>
<td>88</td>
<td>13</td>
<td>193</td>
<td>101</td>
<td>20</td>
<td>229</td>
<td>19</td>
</tr>
</tbody>
</table>

Table 4. Method MM-3, ordinary-gradient algorithm, number of iterations \( N_*. \)
Descent property on \( P \) enforced.

<table>
<thead>
<tr>
<th>Examples</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
<th>8.6</th>
<th>8.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eqs. (54)-(55)</td>
<td>88</td>
<td>13</td>
<td>164</td>
<td>82</td>
<td>20</td>
<td>156</td>
<td>18</td>
</tr>
</tbody>
</table>

(a) Number of iterations exceeded 1000.
Table 5. Method MM-1, conjugate

<table>
<thead>
<tr>
<th>k</th>
<th>8.1</th>
<th>8.2</th>
<th>8.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>(a)</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>(a)</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>$10^0$</td>
<td>(a)</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>$10^1$</td>
<td>47</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>$10^2$</td>
<td>20</td>
<td>76</td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Method MM-2, conjugate

<table>
<thead>
<tr>
<th>k</th>
<th>8.1</th>
<th>8.2</th>
<th>8.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>(a)</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>(a)</td>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>$10^0$</td>
<td>(a)</td>
<td>33</td>
<td>7</td>
</tr>
<tr>
<td>$10^1$</td>
<td>47</td>
<td>48</td>
<td>11</td>
</tr>
<tr>
<td>$10^2$</td>
<td>20</td>
<td>71</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 7. Method MM-3, conjugate

<table>
<thead>
<tr>
<th>k</th>
<th>8.1</th>
<th>8.2</th>
<th>8.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eqs. (56),(58)</td>
<td>43</td>
<td>15</td>
<td>7</td>
</tr>
</tbody>
</table>

(a) Number of iterations exceed
Table 8. Method MM-1, modified-quasilinearization algorithm, number of iterations

<table>
<thead>
<tr>
<th>k</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
<th>8.6</th>
<th>8.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>(a)</td>
<td>14</td>
<td>(a)</td>
<td>(a)</td>
<td>56</td>
<td>(a)</td>
<td>68</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>(a)</td>
<td>11</td>
<td>31</td>
<td>34</td>
<td>21</td>
<td>(a)</td>
<td>(b)</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>46</td>
<td>14</td>
<td>37</td>
<td>17</td>
<td>14</td>
<td>86</td>
<td>30</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>10</td>
<td>13</td>
<td>19</td>
<td>13</td>
<td>26</td>
<td>29</td>
<td>50</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>4</td>
<td>13</td>
<td>25</td>
<td>22</td>
<td>46</td>
<td>20</td>
<td>(a)</td>
</tr>
</tbody>
</table>

Table 9. Method MM-2, modified-quasilinearization algorithm, number of iterations

<table>
<thead>
<tr>
<th>k</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
<th>8.6</th>
<th>8.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>(a)</td>
<td>9</td>
<td>90</td>
<td>(a)</td>
<td>(c)</td>
<td>(a)</td>
<td>49</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>(a)</td>
<td>8</td>
<td>20</td>
<td>26</td>
<td>(c)</td>
<td>(a)</td>
<td>(b)</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>46</td>
<td>11</td>
<td>(d)</td>
<td>12</td>
<td>16</td>
<td>58</td>
<td>23</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>10</td>
<td>10</td>
<td>17</td>
<td>11</td>
<td>21</td>
<td>19</td>
<td>47</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>4</td>
<td>12</td>
<td>25</td>
<td>21</td>
<td>45</td>
<td>16</td>
<td>(a)</td>
</tr>
</tbody>
</table>

Table 10. Method MM-3, modified-quasilinearization algorithm, number of iterations

<table>
<thead>
<tr>
<th>k</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
<th>8.6</th>
<th>8.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eqs. (56),(58)</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>9</td>
<td>13</td>
<td>21</td>
<td>11</td>
</tr>
</tbody>
</table>

(a) Number of iterations exceeded 100.
(b) Number of bisections exceeded 20.
(c) Exponential overflow.
(d) Algorithm converged to a different relative minimum.
References


