AN UPDATING RULE FOR THE PENALTY CONSTANT
USED IN THE PENALTY FUNCTION METHOD
FOR MATHEMATICAL PROGRAMMING PROBLEMS

by

GARY M. COGGINS

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

Thesis Director's signature:

Houston, Texas
May 1972
ABSTRACT

An Updating Rule for the Penalty Constant Used in the Penalty Function Method For Mathematical Programming Problems

by

GARY M. COGGINS

This thesis considers the problem of minimizing a function \( f(x) \) subject to a constraint \( \varphi(x) = 0 \), where \( f \) is a scalar quantity, \( x \) an \( n \)-vector, and \( \varphi \) a \( q \)-vector, with \( q \) being less than \( n \). The penalty function method is investigated; that is, the vector \( x \) is viewed as unconstrained, and the function \( f(x) \) is replaced by the penalty function \( U(x, k) = f(x) + kP(x) \). Here, the penalty constant \( k \) is a positive, scalar quantity, and \( P(x) = \varphi^T(x)\varphi(x) \) is the norm of the constraint error.

Crucial to the penalty function method is the prediction of the rate \( \pi = k_*/k \) at which the penalty constant should be increased when shifting from one cycle of the algorithm to the next. Here, \( k \) denotes the penalty constant of the present cycle, and \( k_* \) denotes the penalty constant of the next cycle.

In this dissertation, two updating rules are compared: (a) the penalty constant is increased at a constant rate \( \pi = \text{const} \), and (b) the penalty constant is increased at a variable rate \( \pi = 10\sqrt{P/P_*} \), where \( P \) is the constraint error at the beginning of a cycle and \( P_* \) is the desired constraint error at convergence.

In order to evaluate these updating rules, six numerical examples are
investigated. The first example deals with a quadratic function subject to linear constraints; the remaining examples deal with nonquadratic functions subject to nonlinear constraints. Each example is solved with the conjugate-gradient algorithm and the modified quasilinearization algorithm for several starting values of the penalty constant.

From the numerical experiments, the following conclusions, concerning the number of iterations for convergence, arise: (i) the variable-rate updating rule is superior to the constant-rate updating rule in both the conjugate gradient algorithm and the modified quasilinearization algorithm; and (ii) the above superiority is more pronounced in the conjugate-gradient algorithm than in the modified quasilinearization algorithm.
ACKNOWLEDGEMENT

The author is indebted to Drs. A. Miele and H.Y. Huang for helpful discussions. He would also like to thank Mr. A.V. Levy for checking the numerical accuracy of the figures included. This research was supported by the National Science Foundation, Grant No. GP-32453.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Statement of the Problem</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Penalty Function Method</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Conjugate-Gradient Algorithm</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Modified Quasilinearization Algorithm</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Experimental Conditions and Numerical Examples</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Numerical Results and Conclusions</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Tables</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>28</td>
</tr>
</tbody>
</table>
1. Introduction

During the past several years, a considerable amount of work has been done on the problem of minimizing a function \( f(x) \) subject to a constraint \( c(x) = 0 \) using numerical methods. In the above problem, \( f \) is a scalar quantity, \( x \) is an \( n \)-vector, and \( \phi \) is a \( q \)-vector, with \( q \) being less than \( n \).

For the solution of the problem indicated, two main approaches have been considered. One is to develop algorithms such that the constraints are satisfied, at least to first order, at the end of each iteration (Refs. 1-2). Another is to develop algorithms where new functions are introduced which allow us to view the vector \( x \) as unconstrained. The penalty function method (Refs. 3-6) is an approach of the latter type.

Crucial to the penalty function method is the prediction of the rate \( \pi = k_\ast / k \) at which the penalty constant should be increased when shifting from one cycle of the algorithm to the next. Here, \( k \) denotes the penalty constant of the present cycle, and \( k_\ast \) denotes the penalty constant of the next cycle.

In this dissertation, two techniques are compared: (a) the penalty constant is increased at a constant rate \( \pi = \text{const} \), and (b) the penalty constant is increased at a variable rate \( \pi = 10 \sqrt{(P/P_c)} \), where \( P \) is the constraint error at the beginning of a cycle and \( P_c \) is the desired constraint error at convergence. The comparison is done through several numerical examples, which are solved with both the conjugate-gradient algorithm and the modified quasilinearization algorithm.
2. **Statement of the Problem**

The problem is that of minimizing the function

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

subject to the constraint

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

where \( f \) is a scalar quantity, \( x \) an \( n \)-vector, \( \varphi \) a \( q \)-vector, and \( q \) is less than \( n \). In Eqs. (1)-(2), it is assumed that the first and second partial derivatives of the functions \( f(x) \) and \( \varphi(x) \) with respect to \( x \) exist and are continuous and that a constrained minimum of the function \( f(x) \) exists.

2.1. **Exact First-Order Conditions.** From theory of maxima and minima, it is known that the above problem is equivalent to that of minimizing the augmented function

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

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

If

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

\footnote{All vectors are column vectors.}
is the gradient of the augmented function, the optimum solution for \( x \) and \( \lambda \) must satisfy the simultaneous equations

\[
\varphi(x) = 0, \quad F'_x(x, \lambda) = 0
\]  

(5)

2.2. Approximate Solutions. Since the system of equations (5) is generally nonlinear, approximate methods must be used to solve the system. These methods are of two types: first-order methods and second-order methods. In this thesis, one method of each type is employed, specifically, the conjugate-gradient method and the modified quasilinearization method.

If one introduces the scalar quantities

\[
P(x) = \varphi^T(x) \varphi(x), \quad Q(x, \lambda) = F'^T_x(x, \lambda) F'_x(x, \lambda)
\]  

(6)

which measure the errors in the constraint and the optimum conditions, it is then observed that \( P = 0 \) and \( Q = 0 \) for the optimum solution, while \( P > 0 \) and/or \( Q > 0 \) for any approximate solution. When approximate methods are used, they must ultimately lead to values of \( x \) and \( \lambda \) such that

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

(7)

or such that

\[
R(x, \lambda) \leq \varepsilon_3
\]  

(8)

where

\[
R(x, \lambda) = P(x) + Q(x, \lambda)
\]  

(9)

\footnote{In Eq. (4), the gradients \( f_x \) and \( F_x \) denote n-vectors and the matrix \( \varphi_x \) is n x q.}
denotes the cumulative error in the constraint and the optimum condition.

In Eqs. \((7)-(8)\), \(e_1, e_2, e_3\) are small, preselected numbers, which determine the accuracy desired in the approximate solution. Also, if one choses

\[ e_1 = e_2 = e_3, \]

then satisfaction of Ineq. \((8)\) implies satisfaction of Ineqs. \((7)\).
3. **Penalty Function Method**

In the penalty function method, one views the vector $x$ as unconstrained and replaces the problem of minimizing (1) subject to (2) with the problem of minimizing the penalty function

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

subject to no constraint. Now, consideration is given to minimizing $U(x, k)$ with respect to $x$ for given $k$. Regardless of the algorithm employed, the first-order condition to be satisfied at convergence is

$$U(x, k) = f(x) + kP(x) = f(x) + 2k\psi_x(x)\varphi(x) = 0$$

(11)

If the exact optimum condition, given by Eq. (5-2), is rewritten in the form

$$F(x, \lambda) = f(x) + \psi_x(x)\lambda = 0$$

(12)

and is compared with Eq. (11), one sees that they are identical provided that

$$\lambda = 2k\varphi(x)$$

(13)

In Eq. (13), $\lambda$ is the multiplier of the exact solution given by Eq. (5-2) and $\varphi(x)$ is the constraint error of the penalty function method at convergence. From Eq. (13), several considerations follow.

(i) Since $\lambda \neq 0$ in Eq. (12) and since the desired constraint error is $\varphi(x) = 0$, then the theoretical value of the penalty constant should be $k = \infty$.

(ii) Since the penalty constant $k = \infty$ is not feasible on a computer, it must be replaced by some large but finite value, precisely
\[ k = \frac{1}{2} \sqrt{\left( \lambda^T \lambda / P \right)_{c}} \]  

where \( \lambda_c \) is the value of the multiplier at convergence and \( P_c \) is the desired constraint error at convergence. For example, if one assumes \( (\lambda^T \lambda)_c = 1 \) and \( P_c = 10^{-12} \), then the penalty constant required for convergence is \( k = 0.5 \times 10^6 \).

(iii) If the function \( f(x) \) is nonquadratic and/or the function \( \varphi(x) \) is nonlinear, large values of the penalty constant are objectionable far away from the solution, because of the combination of curvature and steepness effects. Near the solution, large values of \( k \) are tolerable (and even desirable), if one employs the conjugate-gradient algorithm or the modified quasilinearization algorithm. This is because the displacements \( \Delta x \) become so small that the function \( f(x) \) can be replaced by its quadratic expansion and the constraint \( \varphi(x) \) can be replaced by its linear expansion.

3.1. Description of the Method. For the above considerations, the following outline of the 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, one minimizes the penalty function (10) for a given value of the penalty constant \( k \). The minimum of \( U(x, k) \) is achieved when the following stopping condition is satisfied:

\[ U_x^T(x, k)U_x(x, k) \leq \varepsilon_4 \]  

(15)
where $\varepsilon_4$ is a small, preselected number.

(c) Next, the penalty constant is increased according to the updating rule

$$k_* = \pi k$$

(16)

where $\pi > 1$ is the penalty constant ratio. Then, one returns to (b), and the next cycle of the penalty function method is started.

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

$$\varphi^T(x)\varphi(x) + U^T_k(x, k)U_k(x, k) \leq \varepsilon_5$$

(17)

where $\varepsilon_5$ is a small, preselected number.

3.2. Updating Rule. Crucial to the penalty function method is the prediction of the rate $\pi = k_*/k$ at which the penalty constant should be increased when shifting from one cycle of the algorithm to the next. Here, $k$ denotes the penalty constant of the present cycle, and $k_*$ denotes the penalty constant of the next cycle.

The simplest method consists of keeping $\pi$ constant throughout the algorithm, for example,

$$\pi = 5 \quad \text{or} \quad \pi = 10$$

(18)

An alternate technique consists of varying $\pi$ and selecting it in such a way that, at the end of the next cycle, the constraint error has the value required for convergence. If Eq. (13) is applied once at the end of the present cycle and once at the end of the next cycle, the following relations are obtained:
\[ \lambda^T \lambda = 4k^2 p, \quad \lambda_\ast^T \lambda_\ast = 4k^2 p_\ast \quad (19) \]

and imply that
\[
\pi = \sqrt[1]{(P/P_\ast) \left( \frac{\lambda^T \lambda_\ast}{\lambda^T \lambda} \right)} \quad (20)
\]

If one neglects the change in the norm of the multiplier between the end of the present cycle and the end of the next cycle, Eq. (20) reduces to
\[
\pi = \sqrt{(P/P_\ast)} \quad (21)
\]

where \( P \) is the constraint error at the end of the present cycle and \( P_\ast \) is the constraint error at the end of the next cycle.

In order to increase the probability of constraint satisfaction at the end of the next cycle, the parameter \( P_\ast \) should be somewhat less than \( \varepsilon_1 \) in Ineq. (7-1), that is, somewhat less than \( P_C \) (the desired constraint error of convergence). Therefore, if one takes
\[
P_\ast = 10^{-2} P_C \quad (22)
\]

the penalty constant ratio becomes
\[
\pi = 10\sqrt{(P/P_C)} \quad (23)
\]

3.3. **Remark.** In Ref. 3, Kelley proposed that the penalty constant be evaluated with Eq. (14), with \( \lambda \) estimated from Eq. (12). Since \( \lambda \) is a \( q \)-vector and \( F_X \) is an \( n \)-vector, one has two choices: (i) to employ any \( q \) of Eqs. (12)
or (ii) to employ a least-square approach to the solution of Eqs. (12). In the latter case, the multiplier $\lambda$ satisfies the q-vector relation

$$\mathbf{w}_x^T(x) \mathbf{w}_x(x) \lambda + \mathbf{w}_x^T(x) f_x(x) = 0$$  \hspace{1cm} (24)

It is believed that the approach employed here is simpler than that of Ref. 3, in that the solution of systems of linear equations is bypassed. This is in keeping with the main concept of the penalty function method and is especially important for large systems.
4. **Conjugate-Gradient Algorithm**

Let \( x \) denote the nominal point, \( \hat{x} \) the previous point, \( \bar{x} \) 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. Both \( p \) and \( \hat{p} \) are \( n \)-vectors, while \( \gamma \) and \( \alpha \) are scalar quantities. With this understanding, the conjugate-gradient algorithm is represented by

\[
U_x(x, k) = f_x(x) + 2k \varphi_x(x) \psi(x)
\]

\[
\gamma = U_x^T(x, k) U_x(x, k) / U_x^T(\hat{x}, k) U_x(\hat{x}, k)
\]

\[
p = U_x^T(x, k) + \gamma \hat{p}
\]

\[
\Delta x = - \alpha p
\]

\[
\bar{x} = x + \Delta x
\]

For a given nominal point \( x \), directional coefficient \( \gamma \) and penalty constant \( k \), Eqs. (25) give a complete iteration leading to the varied point \( \bar{x} \), providing one specifies the value of the stepsize \( \alpha \). For the first iteration of a cycle, and every \( nth \) iteration thereafter, Eq. (25-2) is bypassed and is replaced by \( \gamma = 0 \).

4.1. **Stepsize Determination.** The position vector at the end of each step can be written as

\[
\bar{x} = x - \alpha p
\]
where \( p \) is the search direction used. This is a one parameter family of varied points \( x \), for which the penalty function takes the form

\[
U(x, k) = U(x - \alpha p, k) = \tilde{U}(\alpha)
\]  

(27)

A precise search to be used in determining \( \alpha \) is described as follows.

Assuming that a minimum of \( \tilde{U}(\alpha) \) exists, one employs some one-dimensional search scheme (for instance, quadratic interpolation, cubic interpolation, or quasilinearization) to determine the value of \( \alpha \) for which

\[
\tilde{U}_\alpha(\alpha) = 0
\]  

(28)

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

\[
|\tilde{U}_\alpha(\alpha)| \leq \varepsilon_6 \quad \text{or} \quad |\tilde{U}_\alpha(\alpha)| \leq \varepsilon_7 |\tilde{U}(0)|
\]  

(29)

where \( \varepsilon_6 \) and \( \varepsilon_7 \) are small, preselected numbers. Of course, the value of \( \alpha \) satisfying Ineq. (29) must be such that

\[
\tilde{U}(\alpha) < \tilde{U}(0)
\]  

(30)

In this thesis, a multistep, corrected quasilinearization search is used to determine the optimum value of \( \alpha \). If \( \mu \) denotes a scaling factor and \( \rho \) a direction factor such that

\[
0 \leq \mu \leq 1, \quad \rho = \pm 1
\]  

(31)
the search algorithm is described by

\[ \Lambda = - \frac{\tilde{U}_{\alpha}(\alpha_0)}{\tilde{U}_{\alpha\alpha}(\alpha_0)} \]

\[ \rho = \text{sign} [\tilde{U}_{\alpha\alpha}(\alpha_0)] \]

\[ \Delta \alpha = \mu \rho \Lambda \]

\[ \alpha = \alpha_0 + \Delta \alpha \]  

where \( \alpha_0 \) denotes the nominal stepsize. For given \( \alpha_0 \), Eqs. (32) give a complete iteration leading from the nominal stepsize \( \alpha_0 \) to the varied stepsize \( \alpha \), providing one specifies the scaling factor \( \mu \).

Once a nominal stepsize \( \alpha_0 \) is chosen, one sets \( \mu = 1 \) and computes the varied stepsize \( \alpha \) with Eqs. (32). If the following inequality is satisfied:

\[ \tilde{U}(\alpha) < \tilde{U}(\alpha_0) \]  

the scaling factor \( \mu \) is acceptable. If not, the value of \( \mu \) is successively bisected until Ineq. (33) is met. The value obtained for \( \alpha \) becomes the nominal value for the next search step, and the procedure is repeated until Ineq. (29) is satisfied. To start the search, the simplest assumption is

\[ \alpha_0 = 0 \]  

(34)
5. Modified Quasilinearization Algorithm

Let \( x \) denote the nominal point, \( \bar{x} \) the varied point, and \( \Delta x \) the displacement leading from the nominal point to the varied point. If \( \alpha \) denotes a scaling factor and \( \rho \) a direction factor such that

\[
0 \leq \alpha \leq 1, \quad \rho = \pm 1
\]  

the modified quasilinearization algorithm is represented by \(^3\)

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

\[
U_{xx}(x, k) = f_{xx}(x) + 2k[\varphi_{xx}(x) \varphi(x) + \varphi_x(x) \varphi_x^T(x)]
\]

\[
A = -U_{xx}^{-1}(x, k)U_x(x, k)
\]

\[
\rho = -\text{sign}[U_x^T(x, k)A]
\]

\[
\Delta x = \alpha \rho A
\]

\[
\bar{x} = x + \Delta x
\]

For a given nominal point \( x \) and penalty constant \( k \), Eqs. (36) give a complete iteration leading to the varied point \( \bar{x} \), providing one specifies the value of the stepsize \( \alpha \). In this connection, one sets \( \alpha = 1 \) and verifies whether the following inequality is satisfied: \(^4\)

\[
\tilde{U}(\alpha) < \tilde{U}(0)
\]  

\(^3\) In Eqs. (36), the matrices \( U_{xx} \) and \( f_{xx} \) are \( n \times n \), and \( \varphi_{xx} \) denotes an \( n \times n \times q \) three-dimensional array. 

\(^4\) The starting stepsize \( \alpha = 1 \) is suggested by the fact that it yields one-step convergence if \( \tilde{U}(\alpha) \) is quadratic.
If this is the case, the scaling factor $\alpha = 1$ is acceptable. Otherwise, the value of $\alpha$ is successively bisected until Ineq. (37) is met.
6. **Experimental Conditions and Numerical Examples**

Six numerical examples were solved using a Burroughs B-5500 Computer and double-precision arithmetic. The algorithms were programmed using FORTRAN-IV. The penalty function method was employed in conjunction with three penalty constant ratios, specifically

\[ \pi = 5, \quad \pi = 10, \quad \pi = 10^7 \sqrt{P} \]  

where \( P \) is the constraint error at the beginning of a cycle. In the experiments, both the number of iterations for convergence \( N_* \) and the number of cycles for convergence \( N_c \) were recorded.

Concerning the stepsize used in the conjugate-gradient algorithm, a precise one-dimensional search on the function \( \tilde{U}(\alpha) \) was conducted such that, in any given step, the inequality

\[ \tilde{U}(\alpha) < \tilde{U}(\alpha_0) \]  

was satisfied, where \( \alpha_0 \) is the nominal stepsize and \( \alpha \) is the varied stepsize. The search was terminated when the stopping condition

\[ \tilde{U}^2(\alpha) < \tilde{U}^2(0) \times 10^{-6} \]  

was satisfied. Concerning the stepsize used in the modified quasilinearization algorithm, a bisection process (starting from \( \alpha = 1 \)) was employed until the following stopping condition was satisfied:

\[ \tilde{U}(\alpha) < \tilde{U}(0) \]
For both the conjugate-gradient algorithm and the modified quasilinearization algorithm, convergence of a cycle was defined as

\[ U^T_x(x,k)U_x(x,k) \leq 10^{-12} \]  \hspace{1cm} (42)

and convergence of the complete algorithm was defined as

\[ \psi^T(x)\varphi(x) + U^T_x(x,k)U_x(x,k) \leq 10^{-12} \]  \hspace{1cm} (43)

Conversely, nonconvergence was defined by means of the following inequalities:

(a) \( N \geq 200 \) for the conjugate-gradient algorithm
(b) \( N \geq 100 \) for the modified quasilinearization algorithm
(c) \( N_s \geq 20 \)
(d) \( M \geq 0.4 \times 10^{69} \)  \hspace{1cm} (44)

Here, \( N \) is the iteration number, \( N_s \) is the number of stepsize bisections required to satisfy Ineq. (39) in the conjugate-gradient algorithm and Ineq. (41) in the modified quasilinearization algorithm, and \( M \) is the modulus of any quantity used in the algorithm. The nominal point chosen to start all of the examples was

\[ x_i = 2, \quad i = 1, 2, \ldots, n. \]  \hspace{1cm} (45)

**Example 6.1.** Consider the problem of minimizing the function

\[ f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \]  \hspace{1cm} (46)
subject to the constraints

\[ x_1 + 3x_2 = 0, \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0 \]  (47)

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

\[ x_1 = -0.7674, \quad x_2 = 0.2558, \quad x_3 = 0.6279, \quad x_4 = -0.1162, \quad x_5 = 0.2558 \]  (48)

and

\[ \lambda_1 = 2.0465, \quad \lambda_2 = 2.2325, \quad \lambda_3 = -5.9534 \]  (49)

**Example 6.2.** Consider the problem of minimizing the function

\[ f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \]  (50)

subject to the constraint

\[ x_1 (1 + x_2) + x_3^4 - 4 - 3\sqrt{2} = 0 \]  (51)

This function admits the relative minimum \( f = 0.3256 \times 10^{-1} \) at the point defined by

\[ x_1 = 1.1048, \quad x_2 = 1.1966, \quad x_3 = 1.5352 \]  (52)

and

\[ \lambda_1 = -0.1072 \times 10^{-1} \]  (53)
Example 6.3. Consider the problem of minimizing the function

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

subject to the constraints

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

This function admits the relative minimum \( f = 0.7677 \times 10^{-1} \) and the point defined by

\[ x_1 = 1.1911, \quad x_2 = 1.3626, \quad x_3 = 1.4728, \quad x_4 = 1.6350, \quad x_5 = 1.6790 \]  

and

\[ \lambda_1 = -0.3882 \times 10^{-1}, \quad \lambda_2 = -0.1672 \times 10^{-1}, \quad \lambda_3 = -0.2879 \times 10^{-3} \]

Example 6.4. Consider the problem of minimizing the function

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

subject to the inequality constraint

\[ x_1 \leq -1 \]  

Introduce the auxiliary variable \( x_3 \) defined by

\[ x_1 + x_3^2 + 1 = 0 \]  

Then, the previous problem can be recast as that of minimizing the function (58)
subject to the equality constraint (60). The function (58) admits the relative minimum \( f = 0.0400 \) at the point defined by

\[
x_1 = -1.0000, \quad x_2 = 1.0000, \quad x_3 = 0.0000
\]  

(61)

and

\[
\lambda_1 = 0.0400
\]  

(62)

Example 6.5. Consider the problem of minimizing the function

\[
f = -x_1
\]  

(63)

subject to the inequality constraints

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

(64)

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

\[
x_2 - x_1^3 - x_3^2 = 0, \quad x_1^2 - x_2 - x_4 = 0
\]  

(65)

Then, the previous problem can be recast as that of minimizing the function (63) subject to the equality constraints (65). The function (63) admits the relative minimum \( f = -1.0000 \) at the point defined by

\[
x_1 = 1.0000, \quad x_2 = 1.0000, \quad x_3 = 0.0000, \quad x_4 = 0.0000
\]  

(66)

and

\[
\lambda_1 = -1.0000, \quad \lambda_2 = -1.0000
\]  

(67)
Example 6.6. Consider the problem of minimizing the function

\[ f = \log x_3 - x_2 \]  

subject to the equality constraint

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

and the inequality constraint

\[ x_3 \geq 1 \]

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 \textit{a posteriori} with (71). The function (72) admits the relative minimum \( f = -1.7320 \) at the point defined by

\[ x_1 = 0.0000, \ x_2 = 1.7320, \ x_3 = 1.0000 \]

and

\[ \lambda_1 = 0.2886 \]
7. Numerical Results and Conclusions

For the previous examples and experimental conditions, the penalty function method was tested using both the conjugate-gradient algorithm and the modified quasilinearization algorithm. Several starting values of $k$, ranging from $10^{-2}$ to $10^2$, were considered. The numerical results are as shown in Tables 1-12, where the number of iterations at convergence $N_*$ and the number of cycles for convergence $N_c$ are given. From the examples, the following conclusions arise.

(i) For the conjugate-gradient algorithm, the variable-rate updating rule is much superior to the constant-rate updating rule.

(ii) For the modified quasilinearization algorithm, the variable-rate updating rule is also superior to the constant-rate updating rule, even though the superiority is not as pronounced as with the conjugate-gradient algorithm.

(iii) A remarkable property of the variable rate updating rule is that convergence occurs most of the times in $N_c = 2$ cycles and occasionally in $N_c = 3$ cycles. This makes unnecessary the use of the extrapolation devices employed, for example, in Ref. 5.
Table 1. Number of iterations at convergence $N_*$ for Example 6.1.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasilinearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>70</td>
<td>52</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>60</td>
<td>47</td>
</tr>
<tr>
<td>$10^0$</td>
<td>55</td>
<td>42</td>
</tr>
<tr>
<td>$10^1$</td>
<td>48</td>
<td>37</td>
</tr>
<tr>
<td>$10^2$</td>
<td>40</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 2. Number of cycles at convergence $N_c$ for Example 6.1.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasilinearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>$10^0$</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>$10^1$</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>$10^2$</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>
Table 3. Number of iterations at convergence $N_*$ for Example 6.2.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasilinearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>69</td>
<td>62</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>64</td>
<td>60</td>
</tr>
<tr>
<td>$10^0$</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>$10^1$</td>
<td>66</td>
<td>67</td>
</tr>
<tr>
<td>$10^2$</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 4. Number of cycles at convergence $N_c$ for Example 6.2.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasilinearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>$10^0$</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>$10^1$</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$10^2$</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 5. Number of iterations at convergence $N^*$ for Example 6.3.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^7\sqrt{P}$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^7\sqrt{P}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>(a)</td>
<td>(a)</td>
<td>176</td>
<td>29</td>
<td>24</td>
<td>73</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>184</td>
<td>188</td>
<td>86</td>
<td>24</td>
<td>20</td>
<td>51</td>
</tr>
<tr>
<td>$10^0$</td>
<td>186</td>
<td>162</td>
<td>100</td>
<td>22</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td>$10^1$</td>
<td>135</td>
<td>133</td>
<td>79</td>
<td>19</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>$10^2$</td>
<td>135</td>
<td>131</td>
<td>103</td>
<td>27</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 6. Number of cycles at convergence $N_c$ for Example 6.3.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^7\sqrt{P}$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^7\sqrt{P}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>(a)</td>
<td>(a)</td>
<td>2</td>
<td>11</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>$10^0$</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>$10^1$</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$10^2$</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 7. Number of iterations at convergence $N_\ast$ for Example 6.4.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasilinearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>65</td>
<td>53</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>56</td>
<td>47</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>55</td>
<td>48</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>81</td>
<td>76</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>173</td>
<td>170</td>
</tr>
</tbody>
</table>

Table 8. Number of cycles at convergence $N_c$ for Example 6.4.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasilinearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Table 9. Number of iterations at convergence $N_*$ for Example 6.5.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasi-linearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>115</td>
<td>92</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>93</td>
<td>76</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>79</td>
<td>64</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>67</td>
<td>51</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>53</td>
<td>46</td>
</tr>
</tbody>
</table>

Table 10. Number of cycles at convergence $N_c$ for Example 6.5.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Conjugate-gradient algorithm</th>
<th>Modified quasi-linearization algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi = 5$</td>
<td>$\pi = 10$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>
Table 11. Number of iterations at convergence $N_*$ for Example 6.6.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^{7}/P$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^{7}/P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>27</td>
<td>21</td>
<td>10</td>
<td>35</td>
<td>29</td>
<td>14</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>22</td>
<td>18</td>
<td>8</td>
<td>31</td>
<td>26</td>
<td>14</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>22</td>
<td>19</td>
<td>11</td>
<td>30</td>
<td>26</td>
<td>17</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>20</td>
<td>18</td>
<td>11</td>
<td>32</td>
<td>30</td>
<td>23</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>26</td>
<td>24</td>
<td>18</td>
<td>43</td>
<td>41</td>
<td>35</td>
</tr>
</tbody>
</table>

Table 12. Number of cycles at convergence $N_c$ for Example 6.6.

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^{7}/P$</th>
<th>$\pi = 5$</th>
<th>$\pi = 10$</th>
<th>$\pi = 10^{7}/P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>12</td>
<td>9</td>
<td>2</td>
<td>12</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>$10^{1}$</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>$10^{2}$</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
References


