A Numerical Algorithm for Solving Constrained Optimization Problems by Quasi-Newton Methods

H. Ali Ahmed
Mathematics Department, Faculty of Science,
King Abdulaziz University, Jeddah, Saudi Arabia.

Abstract. A numerical algorithm is presented for finding a local optimum of nonlinear programming problems that use quasi-Newton methods. The algorithm uses the squared slack variable philosophy and an updated formula with a numerically stable method for maintaining the positive definiteness at each iteration. A modification of the starting matrix of approximation of the Hessian by the BFGS formula is also given. Some numerical results are given to show the efficiency of the algorithm.

Introduction

In the last decade, a great deal of attention has been paid to extending Newton and quasi-Newton methods for solving general constrained optimization problems. One of the most promising approaches on this line is the method which iteratively solves linearly constrained subproblems.

This method was originated by Wilson[1]. Wilson's algorithm consists of a sequence of quadratic programming subproblems and converges locally with a quadratic rate. However, his method requires second derivatives of both objective and constrained functions.

Modified Wilson's methods with quasi-Newton updates are studied by Garcia and Mangasarian[2] and Han[3]. They show that the methods have superlinear rates of convergence. Powell[4] added further refinement and analysis.

Powell[5] studied the efficiency of BFGS during the calculation when applied to quadratic functions and noticed also the behaviour in sequential quadratic programming methods for constrained optimization whose step length do not exceed one.
The extension of quasi-Newton method to solve inequality constrained problems by converting them into equality constrained by the addition of squared slack variable is well known but rarely used. Tapia\cite{6} attempted to demonstrate that the squared slack variable approach to inequality constraints need not suffer from the standard criticisms attached to it: increased dimension, numerical instability and presence of singularities. Specifically, it is these removable singularities that eventually leads to a pure active constraint approach.

In this paper we develop the algorithm proposed by Tapia\cite{6} to solve constrained optimization problems.

The proposed algorithm applies quasi-Newton method and maintain positive definiteness of the Hessian of the Lagrangian function and also for the other matrix whose diagonal elements are augmented by a factor multiplied by the added squared slack variable. The algorithm comprises two different "Techniques" for maintaining the positive definiteness. The first uses a numerically stable method that is a modification of the modified Cholesky factorization\cite{7} given in Gill et al.\cite{8}. The other technique uses a safeguarded procedure with the BFGS formula\cite{9}.

In section II, the fundamental equations of the proposed algorithm and its basic features are described. In section III, refinements of the algorithm is described from computational point of view and some comments are given.

The Proposed Algorithm

The nonlinear programming problem to be considered in this paper is defined as

\[
\min_x f(x) \\
\text{subject to} \\
g_i(x) = 0, \quad i = 1, 2, \ldots, m \quad (1) \\
g_i(x) \leq 0, \quad i = m + 1, \ldots, p
\]

where \( f, g_i : \mathbb{R}^n \to \mathbb{R} \)

Squared Slack Variable Philosophy

If we introduced the slack variables \( y_{m+1}, \ldots, y_p \) and define \( F, \tilde{g}_i : \mathbb{R}^{n+p-m} \to \mathbb{R} \) by Tapia\cite{6}

\[
F(x,y) = f(x) \\
\tilde{g}_i(x,y) = g_i(x), \quad i = 1, \ldots, m \\
\bar{g}_i(x,y) = g_i(x) + \frac{1}{2} y_i^2, \quad i = m + 1, \ldots, p
\]

then we may consider the following equality constrained optimization problem

minimize \( F(x,y) \)
subject to
\[ \tilde{g}_i(x,y) = 0 \quad , \quad i = 1,2, \ldots , p \]

The Lagrangian function associated with the problem (1) is given by
\[ L(x,y,\lambda) = f(x) - \sum_{i=1}^{p} \lambda_i \tilde{g}_i(x,y) \quad (3) \]

The equation\[^{[6,10]}\]
\[ \begin{align*}
\nabla L(x_*,y_*,\lambda_*) &= \begin{bmatrix}
\nabla_x L(x_*,y_*,\lambda_*) \\
\nabla_y L(x_*,y_*,\lambda_*) \\
\nabla_\lambda L(x_*,y_*,\lambda_*)
\end{bmatrix} = 0
\end{align*} \quad (4) \]

is the stationary point (KT) condition at \( x_*,y_* \) and \( \lambda_* \).

As usual a Taylor series for \( \nabla L \) about \( x_r,y_r \) and \( \lambda_r \) gives
\[ \nabla L(x_r + \delta x_r,y_r + \delta y_r,\lambda_r + \delta \lambda_r) = \nabla L(x_r,y_r,\lambda_r) + [\nabla^2 L(x_r',y_r',\lambda_r')] \begin{bmatrix}
\delta x_r \\
\delta y_r \\
\delta \lambda_r
\end{bmatrix} \quad (5) \]

Neglecting higher order terms and setting the left hand side to zero by virtue of (4) gives the iteration
\[ \begin{bmatrix}
\nabla^2 L(x_r,y_r,\lambda_r') \\
\delta x_r \\
\delta y_r \\
\delta \lambda_r
\end{bmatrix} = - \nabla L(x_r,y_r,\lambda_r) \quad (6) \]

For simplicity we put \( \nabla^2 L(x_r,y_r,\lambda_r') = \nabla^2 L(\cdot) \),
\[ \nabla^2_{rs} L(x_r,y_r,\lambda_r) = \nabla^2_{rs} L(\cdot) , \forall r,s \]

and \( \delta x = \delta x_r, \delta y = \delta y_r, \) and \( \delta \lambda = \delta \lambda_r \)

Equation (6) gives
\[ \begin{bmatrix}
\nabla^2_{rs} L(\cdot) & \nabla^2_{sy} L(\cdot) & \nabla^2_{s\lambda} L(\cdot) \\
\nabla^2_{yr} L(\cdot) & \nabla^2_{yy} L(\cdot) & \nabla^2_{y\lambda} L(\cdot) \\
\nabla^2_{xr} L(\cdot) & \nabla^2_{xy} L(\cdot) & \nabla^2_{x\lambda} L(\cdot)
\end{bmatrix} \begin{bmatrix}
\delta x \\
\delta y \\
\delta \lambda
\end{bmatrix} = - \begin{bmatrix}
\nabla_x L(\cdot) \\
\nabla_y L(\cdot) \\
\nabla_\lambda L(\cdot)
\end{bmatrix} \quad (7) \]

Formulae for \( \nabla^2 L(\cdot) \) and \( \nabla L(\cdot) \) are readily obtained from (3) giving the system
\[ \Lambda_{o,y} = [0 \ \Lambda_y] \]

and \[ \Lambda_y = \text{diag} (y_{m+1}, y_{m+2}, \ldots, y_p) \]

\[ \Lambda_{\lambda} = \text{diag} (\lambda_{m+1}, \lambda_{m+2}, \ldots, \lambda_p) \]

\( A \) is the Jacobian matrix of constraint normals evaluated at \( x_r \), that is

\[ A = [\nabla_x g(x,y)]_{x_r} = [\nabla_x g(x)]_{x_r} \]

In fact it is more convenient to write

\[ \lambda_{r+1} = \lambda_r + \delta \lambda \quad \delta y = y_{r+1} - y_r \quad \delta x = x_{r+1} - x_r \]

and solve the system

\[ G(x_{r+1} - x_r) - A\lambda_{r+1} = -\nabla f \quad (8) \]

\[ -\Lambda_{\lambda} y_{r+1} - \Lambda_{o,y} (\lambda_{r+1} - \lambda_r) = 0 \quad (9) \]

\[ -A^T (x_{r+1} - x_r) - \Lambda^T_{o,y} (y_{r+1} - y_r) = \bar{g} (x,y) \quad (10) \]

Equation (8) gives

\[ x_{r+1} = x_r - G^{-1} \nabla f + G^{-1} A\lambda_{r+1} \quad (11) \]

Equation (9) gives

\[ y_{r+1} = -(A\lambda_r)^{-1} \Lambda_{o,y} (\lambda_{r+1} - \lambda_r) \]

That is

\[
\begin{bmatrix}
y_{m+1} \\
y_{m+2} \\
\vdots \\
y_p
\end{bmatrix}_{r+1} = -
\begin{bmatrix}
\frac{\lambda_{r+1} - \lambda_r}{\lambda_r} y_{m+1} \\
\frac{\lambda_{r+1} - \lambda_r}{\lambda_r} y_{m+2} \\
\vdots \\
\frac{\lambda_{r+1} - \lambda_r}{\lambda_r} y_p
\end{bmatrix}
\]

From which we have

\[
\left( \frac{y_{r+1}}{y_r} \right)_i = \left( \frac{\lambda_r - \lambda_{r+1}}{\lambda_r} \right)_i, \quad i = m + 1, \ldots, p
\]

or

\[
\left( \frac{\lambda_{r+1}}{\lambda_r} \right)_i = \left( \frac{y_r - y_{r+1}}{y_r} \right)_i, \quad i = m + 1, \ldots, p
\]

Equations (10) and (11) give

\[ A^T G^{-1} A\lambda_{r+1} = A^T G^{-1} \nabla f - \Lambda^T_{o,y} (y_{r+1} - y_r) - \bar{g} (x,y) = 0 \]
Using (12), we have

\[ A^T G^{-1} A \lambda_{r+1} = A^T G^{-1} \nabla f - \Lambda^{T} \begin{bmatrix} - \frac{\lambda_{r+1}}{\lambda_r} y_{m+1} \\ \vdots \\ - \frac{\lambda_{r+1}}{\lambda_r} y_p \end{bmatrix} \]

\[ -g(x,y) = 0 \]

Let \( w = (0, 0, \frac{1}{2} y_{m+1}^2, \frac{1}{2} y_p^2) \), that is

\[ \tilde{g}(x,y) = g(x) + w \]

we have

\[ \lambda_{r+1} = (A^T G^{-1} A - \Lambda^{-1} Y^2)^{-1} (A^T G^{-1} \nabla f - g(x) - w) \]

where

\[ \Lambda^{-1} = \text{diag} (\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_p^{-1}) \]

and \( Y^2 = \text{diag} (0, 0, \ldots, y_{m+1}^2, y_{m+2}^2, \ldots, y_p^2) \)

Equation (10) gives

\[ -A^T (x_{r+1} - x_r) = \begin{bmatrix} - \frac{\lambda_{r+1}}{\lambda_r} y_{m+1}^2 \\ - \frac{\lambda_{r+1}}{\lambda_r} y_p^2 \end{bmatrix} + g(x_r) + w_r \]

Then we have

\[ \left( \frac{\lambda_{r+1}}{\lambda_r} \right)_i = \left[ \frac{A^T (x_{r+1} - x_r) + g(x_r) + w_r}{y_r^2} \right]_i \]

and

\[ (y_{r+1})_i = (y_r)_i - \left[ \frac{A^T (x_{r+1} - x_r) + g(x_r) + w_r}{y_r} \right]_i \]

where \((U)_i\) denotes the \(i\)-th component of the vector \(U\).

The method requires initial approximations \(x_0, \lambda_0\), and uses (13), (11) and (14) to generate the iterative sequence \(\{x_r, y_r, \lambda_r\}\).

As it is clear from these equations, the use of the squared slack variables does not necessitate the increase of dimension of the problem except that, it just uses a new vector \(y\) of \((p - m)\) components.
Equality Constrained Problem

If the problem to be considered is an equality constrained problem, that is \( p = m \), then we have the following two fundamental equations.

\[
\lambda_{r+1} = \left[ A^T (x_r) G^{-1} (x_r, \lambda_r) A (x_r) \right]^{-1} \left[ A^T (x_r) G^{-1} (x_r, \lambda_r) \nabla f (x_r) - g (x_r) \right] \quad (15)
\]

\[
x_{r+1} = x_r - G^{-1} (x_r, \lambda_r) \nabla f (x_r) + G^{-1} (x_r, \lambda_r) A (x_r) \lambda_{r+1} \quad (16)
\]

The method requires initial approximations \( x_0, \lambda_0 \), and uses (15) and (16) to generate the iterative sequence \( \{x_r, \lambda_r\} \).

The formula (16) can be put in the form

\[
x_{r+1} = x_r + Z_r
\]

where

\[
Z_r = - G^{-1} (x_r, \lambda_r) \nabla f (x_r) + G^{-1} (x_r, \lambda_r) A (x_r) \lambda_{r+1}
\]

Let the Jacobian matrix \( A(x_r) \), which is of order \( n \times m \), be partitioned to \( m \) column vectors each of which has \( n \) elements, that is

\[
A = [A_1 A_2 \ldots A_m] \quad \text{and hence} \quad G^{-1} A = [G^{-1} A_1 G^{-1} A_2 \ldots G^{-1} A_m]
\]

Let \( v_j = G^{-1} A_j, j = 1, \ldots, m \) (19)

which can be solved by Cholesky factorization. Let \( V \) be a matrix whose column vectors are \( v_1, v_2, \ldots, v_m \), then

\[
A^T G^{-1} A = [A^T v_1 A^T v_2 \ldots A^T v_m] = A^T V
\]

Let \( G = L^T L \), then we have

\[
A^T V = A^T G^{-1} A = A^T (L^{-1})^T (L^{-1}) A = (L^{-1} A)^T (L^{-1} A)
\]

that we retain a symmetrical form. Equations (15) and (16) give

\[
x_{r+1} = x_r - \tilde{v} + V_r \lambda_{r+1}
\]

\[
\lambda_{r+1} = (A^T (x_r) V_r)^{-1} U_r
\]

with

\[
\tilde{v} = G^{-1} (x_r, \lambda_r) \nabla f (x_r)
\]

and

\[
U_r = A^T (x_r) \tilde{v}_r - g (x_r)
\]

We note that \( A^T V = A^T G^{-1} A \) is positive definite as long as \( G \) is positive definite (provided that the columns of \( A \) are linearly independent), and hence equation (21) can also be solved by Cholesky factorization.
From the fundamental equations (15)-(18), it is clear that if $G(x, \lambda)$ is positive definite for all $x \in \mathbb{R}^n$, and $x_0, \lambda_0$ are sufficiently close to $x_*, \lambda_*$, the sequence of approximations $x_r, \lambda_r$ converges to both the solution vector $x_*$ and the vector of optimum Lagrange multipliers $\lambda_*^{10}$.

**The Approximation of The Hessian Matrix $G(x, \lambda)$**

For the proposed algorithm, the Hessian matrix $G(x, \lambda)$ is approximated by the BFGS formula referred by Powell$^9$ on account of its success in solving unconstrained minimization problems. The BFGS formula is given by

$$B_{r+} = B_r - \frac{Z_r Z_r^T B_r}{Z_r^T B_r Z_r} + \frac{\gamma_r \gamma_r^T}{Z_r^T \gamma_r}$$

where $G(x_r, \lambda_r)$ is replaced by $B(x_r, \lambda_r)$, that is, $B_r$.

and $Z_r = x_{r+1} - x_r$,

also $\lambda_r = \nabla_x L(x_{r+1}, \lambda_r) - \nabla_x L(x_r, \lambda_r)$.

$B_0$ is selected to be the unit matrix $I$, of order $n$.

The formula (24) maintains positive definiteness if the condition $Z_r^T \gamma_r > 0$ is satisfied. However, this is not always the case due to the negative curvature of the Lagrangian function.

The use of BFGS formula with $Z_r^T \gamma_r > 0$ should ensure in theory that all Hessian (or inverse Hessian) approximation remain positive definite. However, in practice it is not uncommon for rounding errors to cause the updated matrix to become singular or indefinite. The use of Cholesky factorization allows one to avoid this serious problem: the loss (through rounding errors) of positive definiteness in the Hessian (or inverse Hessian) approximation.

In the following a numerically stable method for maintaining positive definiteness and forming $Z_r$ in this case is presented. It is a modification of the modified Cholesky factorization given in Gill and Murray$^{11}$, and Gill *et al.*$^{8}$. The result is the following$^{7}$:

$$\bar{B}_r = B_r + \mu_r I$$

where

$$\mu_r = 0 \quad \text{if } B_r \text{ is safely positive definite}$$

$$\mu_r > 0 \quad \text{is sufficiently large that } B_r \text{ is safely positive definite otherwise.}$$

Clearly the smallest possible $\mu_r$ (when $B_r$ is not positive definite) is slightly larger than the magnitude of the most negative eigen value of $B_r$. Although this can be computed without too much trouble, a much simpler algorithm is provided that may result in the larger $\mu_r$. 


We first apply the Gill and Murray modified Cholesky factorization algorithm to $B_r$, which results in

$$B_r + E_r = L_r L_r^T,$$

$L_r$ is a lower triangular matrix and $E_r$ a diagonal matrix with nonnegative diagonal elements that are zero if $B_r$ is safely positive definite. If $E_r = 0$, $\mu_r = 0$. If $E_r \neq 0$, we calculate an upper bound $\alpha_1$ on $\mu_r$ using Gershgorian circle theorem, as follows. The matrix $B$ is said to be strictly diagonally dominant if,

$$b_{ii} - \sum_{j=1, j \neq i}^{n} |b_{ij}| > 0$$

and if $p_1, p_2, \ldots, p_n$ are the eigenvalues of $B$, we have

$$\min_{1 \leq i \leq n} p_i \geq \min_{1 \leq i \leq n} \left\{ b_{ii} - \sum_{j=1, j \neq i}^{n} |b_{ij}| \right\}$$

$$\max_{1 \leq k \leq n} p_k \leq \max_{1 \leq k \leq n} \left\{ b_{kk} + \sum_{j=1, j \neq k}^{n} |b_{kj}| \right\}$$

We let

$$\alpha_1 = \max \left\{ (p_{\max} - p_{\min}) \delta^{1/2} - p_{\min}, 0 \right\}$$

If $\alpha_1 = 0$ the matrix $B$ is positive definite.

If $\alpha_1 > 0$ $\alpha_1$ must be added to $B$ so that $\tilde{B} = B + \alpha_1 I$ is strictly diagonally dominant.

Since $\alpha_2 = \max_{1 \leq i \leq n} \{ E_{ii} \} = \max_{1 \leq i \leq n} \{ \mu_i \}$

is also an upper bound on $\mu_r$, we set $\mu_r = \min \{ \alpha_1, \alpha_2 \}$ and conclude the algorithm by calculating the Cholesky factorization of

$$\tilde{B}_r = B_r + \mu_r I.$$  

In the classical Cholesky method, the decomposition $B_r = L_r L_r^T$ is performed in $n$ steps in each of which a column of $L_r$ is determined. The $j$th step Cholesky's method is then given by

$$l_{jj}^2 = b_{jj} - \sum_{k=1}^{j-1} l_{jk}^2$$

$$l_{ij} = (b_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}, \quad i = j + 1, \ldots, n$$

In the modification, the procedure acts directly to limit the size of the elements of
L, when the matrix $B_r$ is not positive definite. It is clear from equation (25) that the elements of the jth row of $L_r$, $1_{jk}$, $k = 1, 2, \ldots, j - 1$, are computed as part of the computation of $1_{jj}$. It is possible for $1_{jj}$ to be very small and hence from equation (26) for $1_{ij}$ to be large. If the $1_{ij}$ elements are considered too large, then can be reduced in modulus by increasing the diagonal elements $b_{jj}$. The algorithm is identical to the classical Cholesky method except that the elements $1_{ij}^2$ are modified so that they are positive and that each of the resulting off-diagonal elements is less than $\beta_j$ in modulus. The parameters $\beta_j$ is the bound imposed on the elements $1_{ij}$ in order that the factorization has to be numerically stable.

Let $1_{jj}^2$ be the modified $1_{jj}^2$. It can be written as

$$1_{jj}^2 = \max (\delta, |1_{jj}^2|, \theta_j^2 / \beta_j^2)$$

(27)

where $\delta$ is the machine precision and if $1_{jj}^2 < \delta$ this corresponds to $B_r$ not being sufficiently positive definite. $\theta_j$ is assumed to be

$$\theta_j = \max_i \{|b_{ij} - \sum_{k=1}^{j-1} 1_{ik} 1_{jk}| : i = j + 1, \ldots, n\}$$

That is $\theta_j = \max_i \{|1_{ij} 1_{jj}| : i = j + 1, \ldots, n\}$

and $|1_{ij}| \leq \beta_j$, $i = j + 1, \ldots, n$

$$1_{jj}^2 = b_{jj} + \mu_j - \sum_{k=1}^{j-1} 1_{jk}^2 = 1_{jj}^2 + \mu_j$$

where

$$\mu_j = 1_{jj}^2 - 1_{jj}^2$$

If this equation for $1_{jj}^2$ is compared with (25), it is clear that the definition of the off-diagonal elements given by (26) is identical to that which would have resulted if, on applying Cholesky's method, the diagonal elements of the matrix to be factorized had been given by $b_{jj} + \mu_j$. The factors obtained by the modified procedure are, therefore, identical to those obtained by applying Cholesky's method to the matrix $B_r = B_r + E_r$.

The value of $\beta_j$ can be determined from the result of the following theorem.

**Theorem**

Let $B_r$ be a symmetric matrix with bounded elements. The jth diagonal element $\mu_j$ of the diagonal matrix $E_r$ associated with the modified Cholesky factorization of $B_r$ is bounded and satisfies

$$0 \leq \mu_j \leq 1_{jj}^2 \leq \max \{ \delta, |b_{jj}| + (j - 1) \beta_j^2, \eta_j / \beta_j + (j - 1) \beta_j \}$$

(29)

where
\[ \eta_j = \max \{ |b_{ij}| : i = j + 1, \ldots, n \} \]

**Proof**

From (27): \( \overline{1}_{jj} \geq \theta_j / \beta_j \), and from the modified Cholesky factorization we have

\[ 1_{ij} = (b_{ij} - \sum_{k=1}^{j-1} l_{ik}^1 1_{jk}) / \overline{1}_{jj}, \quad i = j + 1, \ldots, n, \]

so that \( |1_{ij}| \leq \beta_j \), and

\[ |1_{ij} \overline{1}_{jj}| \leq |b_{ij}| + \sum_{k=1}^{j-1} \beta_k^2, \quad i = j + 1, \ldots, n. \]

If \( \beta_j \geq \beta_k \), \( k = 1, 2, \ldots, (j-1) \), it follows that

\[ \theta_j = \max_{i=j+1, \ldots, n} |1_{ij} \overline{1}_{jj}| \leq \eta_j + (j-1) \beta_j^2. \]

From (25) we have similarly:

\[ 1_{jj}^2 \leq |b_{ij}| + \sum_{k=1}^{j-1} \beta_k^2 \leq |b_{ij}| + (j-1) \beta_j^2. \]

Using these bounds with (27 and 28):

\[ 0 \leq \mu_j \leq 1_{jj}^2 \max \{ \delta, |b_{ij}| + (j-1) \beta_j^2, \eta_j / \beta_j + (j-1) \beta_j \} \]

The choice of (27) ensures positive-definiteness with \( 1_{jj}^2 = 1_{jj}^2 \) if \( 1_{jj}^2 > 0 \) and sufficiently large. To avoid modification as far as possible, and keep \( \mu_j \) small when modification is necessary, we need to choose \( \beta_j \) as large as possible. Formula (25) for an unmodified matrix implies that each

\[ 1_{jk}^2 \leq b_{jk}, \quad k = 1, 2, \ldots, j-1 \]

and if

\[ B^2 \geq \max \{|b_{ij}| : j = 1, 2, \ldots, n\} \]

we have \( 1_{jk}^2 \leq B^2 \) automatically and no modification will be necessary. The final choice of \( \beta \) is

\[ \beta^2 = \max \{ \delta, \gamma, \eta / n \} \]

where \( \gamma \) and \( \eta \) are the largest in modulus of the diagonal and the off-diagonal elements of \( B \), respectively. Condition (27) combined with the choice

\[ \beta_j = \beta = \max \{ \delta, \gamma, \eta / n \} \]

will give a larger lower bound than \( \delta \).
The algorithm for finding a positive definite matrix $\bar{B}_r = B_r + \mu_r I$ can be stated as follows:

1) If $B$ has any negative diagonal elements or the absolute value of the largest off-diagonal element of $B$ is greater than the largest diagonal element of $B$, set $B = B + \mu_1 I$, where $\mu_1 > 0$ is chosen so that the new diagonal is all positive, with the ratio of its smallest to largest element $\geq \delta'$ and the ratio of its largest element to the largest absolute off-diagonal is $\geq 1 + 2 \delta^2$.

2) A perturbed Cholesky decomposition is performed on $B$. It results in $B + E = L \, L^T$, $E$ a non-negative diagonal matrix that is implicitly added to $B$ during the decomposition and contains one or more positive elements if $B$ is not safely positive definite. On output, $\mu_2$ contains the maximum elements of $E$, that is $\mu_2 = \max \{ \mu_j, j = 1, \ldots n \}$.

3) If $\mu_2 = 0$ (i.e. $E = 0$), then $B = L \, L^T$ is safely positive definite and the algorithm terminates, returning $B$ and $L$. Otherwise, it calculates the number $\mu_3$ that must be added to the diagonal of $B$ to make $(B + \mu_3 I)$ safely strictly diagonally dominant. Since both $(B + \mu_2 I)$ and $(B + \mu_3 I)$ are safely positive definite, it then calculates $\mu = \min \{ \mu_2, \mu_3 \}, B = B + \mu I$, calculates the Cholesky decomposition $L \, L^T$ of $B$, and return $B$ and $L$.

Results and Discussion

In this section we discuss further refinements of the algorithm proposed above to accommodate practical calculations.

The matrix $G(x, \lambda)$ which is approximated by the BFGS formula given in (24) is also updated by two techniques:

i) The modification of the modified Cholesky Factorization,\cite{i,ii,iii} given above, and this will be called “Technique I”

ii) We follow Powell’s recommendation \cite{powell9, powell12}, that will be called “Technique II”.

The Powell’s recommendation is as follows:

In formula (24), if the condition,

$$Z_r \, \gamma_r > 0$$

can not be satisfied due to the negative curvature of the objective function, $\gamma_r$ is replaced by the vector

$$\gamma'_r = \theta_r \gamma_r + (1 - \theta_r) B_r \, Z_r$$

where $\theta_r$ is the parameter\cite{iv}. 


\[
\theta_r = \begin{cases} 
1 & \text{if } Z_r^T \gamma_r \geq 0.2 (Z_r^T B_r Z_r) \\
\frac{0.8 Z_r^T B_r Z_r}{Z_r^T B_r Z_r - Z_r^T \gamma_r} & \text{otherwise}
\end{cases}
\]

The parameter \( \theta_r \) is determined to satisfy the condition
\[ Z_r^T \gamma_r \geq 0.2 (Z_r^T B_r Z_r). \]

The algorithm was tested for \( B_0 = I \) and for
\[ B_0 = |f(x_0) + \|g(x_0)\|| \cdot I \]
where \( I \) is the unit matrix of order \( n \), and \( |\cdot| \) is the modulus of the sum of the function and the Euclidean norm of constraints evaluated at \( x_0 \).

We now give certain examples to apply the algorithm for the two cases:

**Case I**

Equality constrained problem.

**Example**

Minimize the function\(^{[14]}\)
\[
f(x) = x_1 x_2 x_3 x_4 x_5
\]
subject to the constraints
\[
x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0,
x_2 x_3 - 5 x_1 x_4 = 0,
x_3^2 + x_2^2 + 1 = 0.
\]

Tables (1), (2) and (3) show the results for this problem, where \( r \) indicates the iteration number, \( x_1, \ldots, x_5 \) are the elements of the vector \( x_r \) (the current point), \( f = f(x_r) \) is the function, and \( \|g(x_r)\| \) is the Euclidean norm of constraints evaluated at \( x_r \).

Table (1) shows the solution for the initial point \((-1, 1, 1, -1, -1)\) where we selected \( B_0 = I \). Table (1) up: shows the result, using Technique I, while Table (1) down: shows it using Technique II.

Table (2) shows the solution for the initial point \((-1, 1, 1, -1, -1.5)\), where we selected \( B_0 = |f(x_0) + \|g(x_0)\|| \cdot I \) for both techniques (Table 2 up: Technique I; Down: Technique II).

We note that the matrix \( B_r \) was not positive definite at \( r = 1 \) (indicated by the asterisk) and was then modified by the corresponding technique to have the next current point.
A Numerical Algorithm for Solving Constrained.

**TABLE 1.** up: Technique I, down: Technique II 
with $B_n = I$

<table>
<thead>
<tr>
<th>iter</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$f$</th>
<th>$|g(x_r)|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>6.480740698</td>
</tr>
<tr>
<td>1*</td>
<td>-0.782515</td>
<td>1.776486</td>
<td>2.048665</td>
<td>-0.782515</td>
<td>-2.109819</td>
<td>-4.701791</td>
<td>4.155117801</td>
</tr>
<tr>
<td>2</td>
<td>-0.759671</td>
<td>1.593748</td>
<td>1.835341</td>
<td>-0.75971</td>
<td>-1.771703</td>
<td>-2.988781</td>
<td>0.55012577</td>
</tr>
<tr>
<td>3</td>
<td>-0.763248</td>
<td>1.598701</td>
<td>1.821861</td>
<td>-0.763248</td>
<td>-1.721225</td>
<td>-9.920459</td>
<td>0.0135756</td>
</tr>
<tr>
<td>4</td>
<td>-0.763596</td>
<td>1.596221</td>
<td>1.826421</td>
<td>-0.7635959</td>
<td>-1.717594</td>
<td>-2.919733</td>
<td>5.71386001E-05</td>
</tr>
<tr>
<td>5</td>
<td>-0.763667</td>
<td>1.595468</td>
<td>1.827634</td>
<td>-0.763667</td>
<td>-1.716935</td>
<td>-2.919707</td>
<td>2.716298E-06</td>
</tr>
<tr>
<td>6</td>
<td>-0.763632</td>
<td>1.595824</td>
<td>1.827063</td>
<td>-0.763632</td>
<td>-1.717242</td>
<td>-2.919701</td>
<td>6.003068E-07</td>
</tr>
<tr>
<td>7</td>
<td>-0.763648</td>
<td>1.595656</td>
<td>1.827332</td>
<td>-0.763648</td>
<td>-1.717097</td>
<td>-2.919701</td>
<td>1.3464468E-07</td>
</tr>
<tr>
<td>8</td>
<td>-0.763641</td>
<td>1.595735</td>
<td>1.827205</td>
<td>-0.763641</td>
<td>-1.717166</td>
<td>-2.919700</td>
<td>2.9374785E-08</td>
</tr>
<tr>
<td>9</td>
<td>-0.763644</td>
<td>1.595698</td>
<td>1.827265</td>
<td>-0.763644</td>
<td>-1.717113</td>
<td>-2.919700</td>
<td>7.1223451E-09</td>
</tr>
</tbody>
</table>

**TABLE 2.** up: Technique I, down: Technique II 
with $B_o = \|f(x_o)\| + \|g(x_o)\|$. I

<table>
<thead>
<tr>
<th>iter</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$f$</th>
<th>$|g(x_r)|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>6.480740698</td>
</tr>
<tr>
<td>1*</td>
<td>-0.782515</td>
<td>1.776486</td>
<td>2.048665</td>
<td>-0.782515</td>
<td>-2.109819</td>
<td>-4.701791</td>
<td>4.155117801</td>
</tr>
<tr>
<td>2</td>
<td>-0.759671</td>
<td>1.593748</td>
<td>1.835341</td>
<td>-0.75971</td>
<td>-1.771703</td>
<td>-2.988781</td>
<td>0.55012577</td>
</tr>
<tr>
<td>3</td>
<td>-0.763248</td>
<td>1.598701</td>
<td>1.821861</td>
<td>-0.763248</td>
<td>-1.721225</td>
<td>-9.920459</td>
<td>0.0135756</td>
</tr>
<tr>
<td>4</td>
<td>-0.763596</td>
<td>1.596221</td>
<td>1.826421</td>
<td>-0.7635959</td>
<td>-1.717594</td>
<td>-2.919733</td>
<td>5.71386001E-05</td>
</tr>
<tr>
<td>5</td>
<td>-0.763667</td>
<td>1.595468</td>
<td>1.827634</td>
<td>-0.763667</td>
<td>-1.716935</td>
<td>-2.919707</td>
<td>2.716298E-06</td>
</tr>
<tr>
<td>6</td>
<td>-0.763632</td>
<td>1.595824</td>
<td>1.827063</td>
<td>-0.763632</td>
<td>-1.717242</td>
<td>-2.919701</td>
<td>6.003068E-07</td>
</tr>
<tr>
<td>7</td>
<td>-0.763648</td>
<td>1.595656</td>
<td>1.827332</td>
<td>-0.763648</td>
<td>-1.717097</td>
<td>-2.919701</td>
<td>1.3464468E-07</td>
</tr>
<tr>
<td>8</td>
<td>-0.763641</td>
<td>1.595735</td>
<td>1.827205</td>
<td>-0.763641</td>
<td>-1.717166</td>
<td>-2.919700</td>
<td>2.9374785E-08</td>
</tr>
<tr>
<td>9</td>
<td>-0.763644</td>
<td>1.595698</td>
<td>1.827265</td>
<td>-0.763644</td>
<td>-1.717113</td>
<td>-2.919700</td>
<td>7.1223451E-09</td>
</tr>
</tbody>
</table>

A Numerical Algorithm for Solving Constrained.
Table 3 shows the solution for the same initial point as in Table (2), where $B_o = I$, using Technique II.

By comparing Table (2) down and Table (3), we note that the number of iterations required to have the solution with high accuracy is decreased when using $B_o = |f(x_o)| + \|g(x_o)\| \cdot I$. The same result was obtained using Technique I.

In the following we give some refinements concerning the application of the algorithm to inequality constrained problems.

The relation (16) gives incorrect values of $(y_{r+1})_i$ when $(y_r)_i$ approaches zero.
The numerical experimentation showed that the best choice of \((y_{r+1})_i\) is the following:

\[
(y_{r+1})_i = \sqrt{-2g_i(x_r)} \quad \text{if } g_i(x_r) < 0
\]

and

\[
(y_{r+1})_i = \sqrt{2g_i(x_r)} \quad \text{if } g_i(x_r) \geq 0
\]

where \(i = m + 1, \ldots, p\).

The choice of \(y_{r+1}\) given by (30) will make the algorithm less sensitive to poor values of \((y_r)_i\).

In addition the algorithm forces \((y_{r+1})_i = 0\)

where \(g_i(x_r) = \max (g_i(x_r) > 0), \ i = m + 1, \ldots, p\)

We cannot guarantee that when we have \(g_i(x_r) > 0\), the choice of \((y_{r+1})_i = 0, \forall i\)
leads to the required solution of the problem.

The matrix \((A^T G^{-1} A - \Lambda^{-1} Y^2)\) in relation (13) may not be positive definite when one or more of the \((\lambda_i), i = m + 1, \ldots, p\), are positive. To maintain positive definiteness of this matrix we replace every component \(i\) of the diagonal matrix \(\Lambda^{-1} Y^2\), given as \((\frac{Y^2}{\lambda_r})_{i}, i = m + 1, \ldots p\)

**Case II**

We now show the behaviour of the algorithm on examples of nonlinear optimization problems having inequality constraints.

We now show the behaviour of the algorithm, on a simple example. A BASIC test program was written and applied to the problem.

Minimize the function

\[
f(x) = x_1^3 + 2 x_2^2 x_3 + 2 x
\]

Subject to

\[
g_1(x) = x_1^2 + x_2 + x_3^2 - 4 = 0
\]

\[
g_2(x) = x_1^2 - x_2 + 2 x_3 - 2 \leq 0
\]

Table (4) gives the values of \(x_1, x_2, x_3\), of a slack variable \(y_2\), of the multipliers \(\lambda_1\) and \(\lambda_2\), of the function \(f = f(x_1, x_2, x_3)\) and of the Euclidean norm of the constraints, \(\|g(x_r)\|, \text{at } r = 0, 1, 2 \ldots\)
Table (4), up: shows the results, where

\[ B_0 = |f(x_o) + \|g(x_o)\|| \times I \]

Table (4), down: shows the results, where

\[ B_0 = I, \]

I is the unit matrix of order n.

The algorithm was also applied to Rosen-Suzuki Problem\(^{13}\).

The matrix \( B_0 \) is selected to be

\[ B_0 = |f(x_o) + \|g(x_o)\|| \times I \]

The problem is:

Minimize the function

\[ f(x) = x_1^2 + x_2^2 + 2 x_3^2 + x_4^2 - 5 x_1 - 5 x_2 - 21 x_3 + 7 x_4 \]

Subject to

\[ g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 - x_1 - x_2 + x_3 - x_4 - 8 \leq 0, \]
\[ g_2(x) = x_1^2 + 2 x_2^2 + x_3^2 + 2 x_4^2 - x_1 - x_4 - 10 \leq 0, \]
\[ g_3(x) = 2 x_1^2 + x_2^2 + x_3^2 + 2 x_1 - x_2 - x_4 - 5 \leq 0 \]
where $x_* = (0,1,2,-1)$ and $f(x_*) = -44$

Tables (5) and (6) show the solution for the initial point $(0,0,0,0,\sqrt{16},\sqrt{20},\sqrt{10})$. 

**Table 5. Technique I,**

<table>
<thead>
<tr>
<th>iter</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_3$</th>
<th>$f$</th>
<th>$g(x_*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\sqrt{16}$</td>
<td>$\sqrt{20}$</td>
<td>$\sqrt{10}$</td>
<td>0</td>
<td>13.7477</td>
</tr>
<tr>
<td>1*</td>
<td>0.290957</td>
<td>0.872872</td>
<td>0.290957</td>
<td>-0.290957</td>
<td>4.00000</td>
<td>4.4721</td>
<td>3.1623</td>
<td>0</td>
<td>-12.8654</td>
</tr>
<tr>
<td>2</td>
<td>0.169868</td>
<td>0.826310</td>
<td>0.515102</td>
<td>-0.465481</td>
<td>3.7374</td>
<td>4.0342</td>
<td>2.8228</td>
<td>0</td>
<td>-17.5932</td>
</tr>
<tr>
<td>3</td>
<td>-0.043453</td>
<td>0.794579</td>
<td>2.738494</td>
<td>-1.599702</td>
<td>3.6009</td>
<td>3.9015</td>
<td>2.8347</td>
<td>0</td>
<td>-54.2644</td>
</tr>
<tr>
<td>4</td>
<td>0.219037</td>
<td>0.950505</td>
<td>3.000186</td>
<td>-1.579063</td>
<td>3.00000</td>
<td>3.2304</td>
<td>2.7752</td>
<td>0</td>
<td>-58.4579</td>
</tr>
<tr>
<td>5</td>
<td>-0.105412</td>
<td>0.758823</td>
<td>2.103292</td>
<td>-1.248170</td>
<td>3.7955</td>
<td>3.4834</td>
<td>0.0</td>
<td>0</td>
<td>-45.1809</td>
</tr>
<tr>
<td>6</td>
<td>0.120099</td>
<td>0.908280</td>
<td>2.056588</td>
<td>-0.942685</td>
<td>3.00000</td>
<td>0.3347</td>
<td>0.7751</td>
<td>0</td>
<td>-44.7419</td>
</tr>
<tr>
<td>7</td>
<td>0.309217</td>
<td>1.150972</td>
<td>2.064797</td>
<td>-0.717407</td>
<td>1.7356</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>-45.2435</td>
</tr>
<tr>
<td>8</td>
<td>0.282190</td>
<td>0.891998</td>
<td>2.073131</td>
<td>-0.968946</td>
<td>1.7416</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>-44.4928</td>
</tr>
<tr>
<td>9</td>
<td>-0.005529</td>
<td>0.956735</td>
<td>2.018494</td>
<td>-0.987516</td>
<td>1.6045</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>-44.0178</td>
</tr>
<tr>
<td>10</td>
<td>-0.000056</td>
<td>1.003201</td>
<td>1.999838</td>
<td>-1.000118</td>
<td>1.5176</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>-44.0078</td>
</tr>
<tr>
<td>11</td>
<td>-0.000470</td>
<td>0.999212</td>
<td>2.000565</td>
<td>-0.999483</td>
<td>1.4051</td>
<td>0.0717</td>
<td>0.0</td>
<td>0</td>
<td>-44.0000</td>
</tr>
<tr>
<td>12</td>
<td>0.000861</td>
<td>1.000083</td>
<td>2.000305</td>
<td>-0.999010</td>
<td>1.4163</td>
<td>0.0058</td>
<td>0.0</td>
<td>0</td>
<td>-44.0051</td>
</tr>
<tr>
<td>13</td>
<td>-0.000126</td>
<td>0.999831</td>
<td>2.000138</td>
<td>-0.999686</td>
<td>1.4163</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>-44.0000</td>
</tr>
<tr>
<td>14</td>
<td>0.000003</td>
<td>1.000077</td>
<td>1.999976</td>
<td>-1.000014</td>
<td>1.4147</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>-44.0000</td>
</tr>
</tbody>
</table>

**Table 6. Technique II,**

<table>
<thead>
<tr>
<th>iter</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_3$</th>
<th>$g(x_*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\sqrt{16}$</td>
<td>$\sqrt{20}$</td>
<td>$\sqrt{10}$</td>
<td>0</td>
</tr>
<tr>
<td>1*</td>
<td>0.290957</td>
<td>0.872872</td>
<td>0.290957</td>
<td>-0.290957</td>
<td>4.00000</td>
<td>4.4721</td>
<td>3.1623</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.169868</td>
<td>0.826310</td>
<td>0.515102</td>
<td>-0.465481</td>
<td>3.7374</td>
<td>4.0342</td>
<td>2.8228</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>-0.043453</td>
<td>0.794579</td>
<td>2.738494</td>
<td>-1.599702</td>
<td>3.6009</td>
<td>3.9015</td>
<td>2.8347</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.219037</td>
<td>0.950505</td>
<td>3.000186</td>
<td>-1.579063</td>
<td>3.00000</td>
<td>3.2304</td>
<td>2.7752</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>-0.105412</td>
<td>0.758823</td>
<td>2.103292</td>
<td>-1.248170</td>
<td>3.7955</td>
<td>3.4834</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.120099</td>
<td>0.908280</td>
<td>2.056588</td>
<td>-0.942685</td>
<td>3.00000</td>
<td>0.3347</td>
<td>0.7751</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.309217</td>
<td>1.150972</td>
<td>2.064797</td>
<td>-0.717407</td>
<td>1.7356</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0.282190</td>
<td>0.891998</td>
<td>2.073131</td>
<td>-0.968946</td>
<td>1.7416</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>-0.005529</td>
<td>0.956735</td>
<td>2.018494</td>
<td>-0.987516</td>
<td>1.6045</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>-0.000056</td>
<td>1.003201</td>
<td>1.999838</td>
<td>-1.000118</td>
<td>1.5176</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>-0.000470</td>
<td>0.999212</td>
<td>2.000565</td>
<td>-0.999483</td>
<td>1.4051</td>
<td>0.0717</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0.000861</td>
<td>1.000083</td>
<td>2.000305</td>
<td>-0.999010</td>
<td>1.4163</td>
<td>0.0058</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>-0.000126</td>
<td>0.999831</td>
<td>2.000138</td>
<td>-0.999686</td>
<td>1.4163</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0.000003</td>
<td>1.000077</td>
<td>1.999976</td>
<td>-1.000014</td>
<td>1.4147</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>-0.000162</td>
<td>0.986744</td>
<td>2.003934</td>
<td>-0.997951</td>
<td>1.4097</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0.000293</td>
<td>1.000646</td>
<td>1.999756</td>
<td>-1.000097</td>
<td>1.4187</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table (5) shows the result using technique I, where the solution is obtained for less than 17 iterations.

Table (6) shows the result using the technique II, where the solution is obtained for more than 18 iterations.

The value of $\lambda$ is $(-0.9999, -1.796E-09, -2.0000)$.

At the solution, we have

$$x_* = (0, 1, 2, -1), \quad y = (0, \sqrt{2}, 0), \quad f = -44, \quad \|g(x_*)\| = 1$$

We note that the negative sign of $\lambda_2$ of Table (4) and that for $\lambda_1, \lambda_2, \lambda_3$, of the last example is due to the selection of positive sign of the second term of the right hand side of the equation of the Lagrangian function (3).

We note also that in Tables (5) and (6), the asterisk indicates that the matrix $B_r$ at $r = 1$ is indefinite. The calculated eigenvalues shows that they lie in the range $-82.96$ to $41.75$. The matrix $B$ was then modified by the corresponding technique to have the next current point.

The proposed algorithm may be useful for solving constrained optimization problems that use Lagrangian functions. It does not suffer from the increased dimension, numerical instability and presence of singularities caused by the presence of slack variables in inequality problems. It maintains the positive definiteness at each iteration and the selection of $B_0$, in BFGS formula, to be the modulus of the sum of the function and the Euclidean norm of constraints may be useful for acceleration of convergence, and for convergence from initial points that are far from the solution.

References


طريقة عدديّة لحل مشاكل الحل الأمل ذات القيود باستخدام الطريقة المشابهة لطريقة نيوتن

حسن عـملي أحمد
قسم الرياضيات، كلية العلوم، جامعة الملك عبد العزيز، جدة، المملكة العربية السعودية

يقدم البحث طريقة عدديّة لإيجاد حل موضعيّ أمثل لمشكلة برمجة غير خطية تحتوي قيوداً على شكل متباينات. وفي هذه الطريقة تضاف إلى المتباينات أتساف مرتبات متغيرة قليلاً لتحويلها إلى معادلات وهو الأسلوب الذي اتبعه نابيا سنة 1980 م. ولكن يضاف إليه استخدام الطريقة شبه النيوتنية مع طريقة عدديّة مستمرة تعتمد على تعديل المصفوفة المقابلة ذات الفاصل الثاني لدالة لاجرانج لكي تصبح مصفوفة إيجابية التعريف. كما يحدث تعديل في قطر المصفوفة المكافئة نتيجة إضافة متغيرات ظل لضمان استمرار إيجابية التعريف للمصفوفة أثناء الحل، وكذلك يحدث تعديل في المصفوفة المقابلة لمصفوفة الفاصل الثاني عند بداية التّنقيب.

ويهدف البحث من هذه التعديلات إلى إيجاد حل أمثل في عدد أقل من الخطوات وكذلك عند البدء من نقطة بعيدة عن الحل.

وقد طبقت الطريقة في حالة المعادلات المتباينات وأعطت أمثلة لذلك.