J.K.A.U.: Sci., vol. 1, pp. 171-190 (1409 A.H. / 1989 A.D.)

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[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[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[7] given in Gill *et al.*[8]. The other technique uses a safeguarded procedure with the BFGS formula[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

$$\begin{aligned} &\min_{x} \ f(x) \\ &\text{subject to} \\ &g_i(x) = 0 \quad , \quad i = 1,2,\dots,m \\ &g_i(x) \leq 0 \quad , \quad i = m+1,\dots,p \end{aligned} \tag{1}$$

where $f, g_i : R^n \to R$

Squared Slack Variable Philosophy

If we introduced the slack variables $y_{m+1}, \dots, y_p$ and define $F, \bar{g}_i : R^{n+p-m} \to R$ by Tapia[6]

$$F(x,y) = f(x)$$

$$\bar{g}_i(x,y) = g_i(x) \quad , \quad i = 1, \dots, m$$

$$\bar{g}_i(x,y) = g_i(x) + \frac{1}{2} y_i^2 , i = m + \quad , p$$

then we may consider the following equality constrained optimization problem

minimize F(x,y)

subject to (2)

$$\bar{g}_i\,(x,y) = 0 \qquad , i = 1,2, \dots , p$$

The Lagrangian function associated with the problem (1) is given by

$$L(x,y,\lambda) = f(x) - \sum_{i=1}^{p} \lambda_i\, \bar{g}_i\,(x,y) \tag{3}$$

The equation[6,10]

$$\nabla L\,(x_*,y_*,\lambda_*) = \begin{array}{c} \nabla_x L\,(x_*,y_*,\lambda_*) \\ \nabla_y L\,(x_*,y_*,\lambda_*) \\ \nabla_\lambda L\,(x_*,y_*,\lambda_*) \end{array} = 0 \tag{4}$$

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

As usual a Taylor series for ∇L about x_r,y_r and λ_r gives

$$\nabla L\,(x_r + \delta x_r,y_r + \delta y_r,\lambda_r + \delta\lambda_r) = \nabla L\,(x_r,y_r,\lambda_r) + [\nabla^2 L\,(x_r',y_r,\lambda_r)] \left.\begin{array}{c} \delta x_r \\ \delta y_r \\ \delta\lambda_r \end{array}\right] + \tag{5}$$

Neglecting higher order terms and setting the left hand side to zero by virtue of (4) gives the iteration

$$[\nabla^2 L\,(x_r,y_r,\lambda_r)] \begin{array}{c} \delta x_r \\ \delta y_r \\ \delta\lambda_r \end{array} = - \nabla L\,(x_r,y_r,\lambda_r) \tag{6}$$

For simplicity we put $\nabla^2 L\,(x_r,y_r,\lambda_r) = \nabla^2 L\,(.)$,

$$\nabla^2_{rs} L\,(x_r,y_r,\lambda_r) = \nabla^2_{rs} L\,(.)\ , \forall r\ , s$$

and $\delta x = \delta x_r$, $\delta y = \delta y_r$ and $\delta\lambda = \delta\lambda_r$

Equation (6) gives

$$\begin{bmatrix} \nabla^2_x L\ (\cdot) & \nabla^2_{xy} L\ (\cdot) & \nabla^2_{x\lambda}\ (\cdot) & \delta x \\ \nabla^2_{yx} L\ (\cdot) & \nabla^2_y L\ (\cdot) & \nabla^2_{y\lambda} L\ (\cdot) & \delta y \\ \nabla^2_{\lambda x}\ (\cdot) & \nabla^2_{\lambda y}\ (\cdot) & \nabla^2_\lambda L\ (\cdot) & \delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x L\,(\cdot) \\ \nabla_y L\,(\cdot) \\ \nabla_\lambda L\,(\cdot) \end{bmatrix}$$

Formulae for $\nabla^2 L\ (\cdot)$ and $\nabla L\ (\cdot)$ are readily obtained from (3) giving the system

$$\begin{array}{cccc} \text{order} & n & p-m & p \\ n & G(x,\lambda) & 0 & -A \\ p-m & 0 & -\Lambda_{\lambda_I} & -\Lambda_{o,y} \\ p & -A^T & -\Lambda^T_{o,y} & 0 \end{array} \left[\begin{array}{c} \delta x \\ \delta y \\ \delta\lambda \end{array}\right. = - \begin{array}{c} \nabla_x f - A\lambda \\ -\Lambda_{\lambda_I} y_r \\ -\bar{g}(x,y) \end{array} \tag{7}$$

$$\Lambda_{o,y} = [\,0\ \Lambda_y\,]$$

and $\Lambda_y = \text{diag}\,(y_{m+1}, y_{m+2}, \dots, y_p)$

$$\Lambda_{\lambda_I} = \text{diag}\,(\lambda_{m+1}, \lambda_{m+2}, \dots, \lambda_p)$$

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

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

In fact it is more convenient to write

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

and solve the system

$$G(x_{r+1} - x_r) - A\lambda_{r+1} = -\nabla f \tag{8}$$

$$-\Lambda_{\lambda_1}\, y_{r+1} - \Lambda_{o,y}\,(\lambda_{r+1} - \lambda_r) = 0 \tag{9}$$

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

Equation (8) gives

$$x_{r+1} = x_r - G^{-1}\,\nabla f + G^{-1}\,A\lambda_{r+1} \tag{11}$$

Equation (9) gives

$$y_{r+1} = -\,(\Lambda_{\lambda_I})^{-1}\,\Lambda_{o,y}\,(\lambda_{r+1} - \lambda_r)$$

That is

$$\begin{bmatrix} y_{m+1} \\ y_{m+2} \\ \vdots \\ y_p \end{bmatrix}_{r+1} = -\begin{bmatrix} (\dfrac{\lambda_{r+1} - \lambda_r}{\lambda_r}\ y_r)_{m+} \\ \\ \\ \dfrac{\lambda_{r+1} - \lambda_r}{\lambda_r}\ y_r)_p \end{bmatrix}$$

From which we have

$$(\frac{y_{r+1}}{y_r})_i = (\frac{\lambda_r - \lambda_{r+1}}{\lambda_r})_i \ , \ i = m + 1, \quad , p$$

or (12)

$$(\frac{\lambda_{r+1}}{\lambda_r})_i = (\frac{y_r - y_{r+1}}{y_r})_i \ , \ i = m + 1, \dots , p$$

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

$$A^T G^{-1} A\lambda_{r+1} = A^T G^{-1} \nabla f - \Lambda^T_{o,y} \begin{bmatrix} -(\frac{\lambda_{r+1}}{\lambda_r} y_r)_{m+1} \\ \vdots \\ -(\frac{\lambda_{r+1}}{\lambda_r} y_r)_p \end{bmatrix} - \bar{g}(x,y) = 0$$

Let $w = (0, 0, \quad , \frac{1}{2} y^2_{m+1} , \quad , \frac{1}{2} y^2_p)$, that is

$$\bar{g}(x,y) = g(x) + w$$

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 $Y^2 = \text{diag}(0, 0, \ldots, y^2_{m+1}, y^2_{m+2}, \ldots, y^2_p)$

Equation (10) gives

$$-A^T (x_{r+1} - x_r) = \begin{bmatrix} 0 \\ -(\frac{\lambda_{r+1}}{\lambda_r} y_r^2)_{m+1} \\ \\ -(\frac{r_{r+1}}{\lambda_r} y_r^2)_p \end{bmatrix} + g(x_r) + w_r$$

Then we have

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

and $\qquad i = m + 1, \ldots, p$

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

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

The method requires initial approximations x_o, λ_o, 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.

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

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

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

The fomula (16) can be put in the form

$$x_{r+1} = x_r + Z_r \tag{17}$$

where

$$Z_r = - G^{-1}(x_r,\lambda_r)\, \nabla f(x_r) + G^{-1}(x_r,\lambda_r)\, A(x_r)\, \lambda_{r+1} \tag{18}$$

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

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

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

$$x_{r+1} = x_r - \bar{v} + V_r \lambda_{r+1}$$

$$\lambda_{r+1} = (A^T(x_r)\, V_r)^{-1} U_r \tag{21}$$

with

$$= G^{-1}(x_r,\lambda_r)\, \nabla f(x_r) \tag{22}$$

and

$$U_r = A^T(x_r)\, \bar{v}_r - g(x_r) \tag{23}$$

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 $x \in R^n$, and x_o,λ_o are sufficiently close to x_*,λ_*, the sequence of approximations x_r,λ_r converges to both the solution vector x_* and the vector of optimum Lagrange multipliers λ_*[10].

The Approximation of The Hessian Matrix G(x,λ)

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

$$B_{r+} = B_r - \frac{B_r Z_r Z_r^T B_r}{Z_r^T B_r Z_r} + \frac{\gamma_r \gamma_r^T}{Z_r^T \gamma_r} \tag{24}$$

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_o 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

$\mu_r = 0$ if B_r is safely positive definite

$\mu_r > 0$ is sufficiently large that B_r is safely positive definite otherwise.

Clearly the smallest possible μ_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 μ_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 α_1 on μ_r using Gerschgorian circle theorem, as follows. The matrix B is said to be strictly diagonally dominant if,

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

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

$$\min_{1\leq i\leq n} P_i \geqslant \min_{1\leq i\leq n} \{b_{ii} - \sum_{\substack{j=1 \\ j\neq i}}^{n} |b_{ij}|\}$$

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

We let

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

if $\alpha_1 = 0$ the matrix B is positive definite

if $\alpha_1 > 0$ α_1 I must be added to B so that $\bar{B} = B + \alpha_1 I$ is strictly diagonally dominant.

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

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

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

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 jth step Cholesky's method is then given by

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

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

In the modification, the procedure acts directly to limit the size of the elements of

L_r 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}, $k = 1,2, \dots, j-1$, 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_{jj}^2 are modified so that they are positive and that each of the resulting off-diagonal elements is less than β_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

$$1_{jj}^{-2} = \max(\delta, |1_{jj}^2|, \theta_j^2 / \beta_j^2) \qquad (27)$$

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

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

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

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

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

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 $\bar{B}_r = B_r + E_r$.

The value of β_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 μ_j of the diagonal matrix E_r associated with the modified Cholesky factorization of B_r is bounded and satisfies

$$0 \leq \mu_j \leq 1_{jj}^{-2} \leq \max\{\delta, |b_{jj}| + (j-1)\beta_j^2, \eta_j/\beta_j + (j-1)\beta_j\} \qquad (29)$$

where

$$\eta_j = \max \{\}b_{ij}| : i = j + 1 , \quad , n \}$$

Proof

From (27) : $\bar{1}_{jj} \geqslant \theta_j / ß_j$,

and from the modified Cholesky factorization we have

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

so that $|1_{ij}| \leqslant ß_j$, and

$$|1_{ij}\, \bar{1}_{jj}| \leqslant |b_{ij}| + \sum_{k=1}^{j-1} ß_k^2 , i = j + \qquad n.$$

If $ß_j \geqslant ß_k$, $k = 1, 2 \quad . (j-1)$, it follows that

$$\theta_j = \max_{i=j+1 , \dots n} |1_{ij}\, \bar{1}_{jj}| \leqslant \eta_j + (j-1)\, ß_j^2$$

From (25) we have similarly:

$$1_{jj}^2 \leqslant |b_{jj}| + \sum_{k=1}^{j-1} ß_k^2 \leqslant |b_{jj}| + (j-1)\, ß_j^2$$

Using these bounds with (27 and 28):

$$0 \leqslant \mu_j \leqslant 1_{jj}^{-2} \max \{\delta, |b_{jj}| + (j-1)\, ß_j^2, \eta_j / ß_j + (j-1)\, ß_j\} \qquad \square$$

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

$$1_{jk}^2 \leqslant b_{jj} \qquad\qquad k = 1,2 , \dots , j-1$$

and if

$$ß^2 \geqslant \max \{ |b_{jj}| : j = 1,2 , \dots , n \}$$

we have $1_{jk}^2 \leqslant ß^2$ automatically and no modification will be necessary. The final choice of ß is

$$ß^2 = \max \{\delta , \gamma , \eta / n \}$$

where γ and η are the largest in modulus of the diagonal and the off-diagonal elements of B_r respectively. Condition (27) combined with the choice

$$ß_j = ß = \max \{\delta , \gamma , \eta / n \}$$

will give a larger lower bound than δ.

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 $\geqslant \delta'$ and the ratio of its largest element to the largest absolute off-diagonal is $\geqslant 1 + 2\ \delta^{\frac{1}{2}}$.

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, μ_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 μ_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 $\mu = \min\{\mu_2, \mu_3\}$, $\bar{B} = B + \mu I$, calculates the Cholesky decomposition $L\,L^T$ of $\bar{B}$, 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,[7,8,11] given above, and this will be called "Technique I"

ii) We follow Powell's recommendation [Powell[9,12]], 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, γ_r is replaced by the vector

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

where θ_r is the parameter[13]:

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

The parameter θ_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 $B_o = I$ and for

$$B_o = |f(x_o) + \|g(x_o)\|\,| \,.\, I$$

where I is the unit matrix of order n, and $|.|$ is the modulus of the sum of the function and the Euclidean norm of constraints evaluated at x_o

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

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0,$$
$$x_2\, x_3 - 5\, x_1\, x_4 = 0,$$
$$x_5^3 + x_2^3 + 1 = 0 \;.$$

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_o = 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 $B_o = |f(x_o) + \|g(x_o)\|\,|$. 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.

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

iter r	x_1	x_2	x_3	x_4	x_5	f	$\|g(x_r)\|$
0	–1.0	1.0	1.0	–1.0	–1.0	–1.0	6.480740698
1*	–0.782515	1.776486	2.048665	–0.782515	–2.109819	–4.701791	4.155117801
2	–0.759671	1.593748	1.833541	–0.75971	–1.771703	–2.988781	0.55012577
3	–0.763248	1.598701	1.821861	–0.763248	–1.721225	–9.920459	0.0135756
4	–0.763596	1.596221	1.826421	–0.7635959	–1.717594	–2.919733	5.71386001E-05
5	–0.763667	1.595468	1.827634	–0.763667	–1.716935	–2.919702	2.716628E-06
6	–0.763632	1.595824	1.827063	–0.763632	–1.717242	–2.919701	6.0030368E-07
7	–0.763648	1.595656	1.827332	–0.763648	–1.717097	–2.919701	1.3464468E-07
8	–0.763641	1.595735	1.827205	–0.763641	–1.717166	–2.919700	2.9374785E-08
9	–0.763644	1.595698	1.827265	–0.763644	–1.717113	–2.919700	7.1223451E-09
0	–1.0	1.0	1.0	–1.0	–1.0	–1.0	6.480740698
1*	–0.782515	1.776485	2.048665	–0.782515	–2.109819	–4.701791	4.155117801
2	–0.764827	1.533736	1.925459	–0.764827	–1.729156	–2.987070	0.604302913
3	–0.7615246	1.682428	1.703842	–0.761525	–1.783456	–2.964804	0.120895158
4	–0.772533	1.560065	1.897210	–0.772533	–1.683949	–2.974551	0.070491933
5	–0.763004	1.616181	1.797335	–0.763004	–1.734668	–2.933522	0.017088607
6	–0.764787	1.586919	1.842042	–0.764487	–1.709465	–2.922773	0.00383146
7	–0.763293	1.600038	1.820450	–0.763293	–1.720866	–2.920378	8.410527E-04
8	–0.763850	1.593707	1.830473	–0.763850	–1.715410	–2.919851	1.873000E-04
9	–0.763551	1.596663	1.825723	–0.763555	–1.717966	–2.919734	4.167593E-05
10	–0.763686	1.595262	1.827966	–0.763688	–1.716757	–2.919708	9.272739E-06
11	–0.763627	1.595921	1.826907	–0.763619	–1.717326	–2.919700	2.060767E-06
12	–0.763638	1.595610	1.827406	–0.763668	–1.717057	–2.919700	4.55568E-07

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

iter r	x_1	x_2	x_3	x_4	x_5	f	$\|g(x_r)\|$
0	–1.0	1.0	1.0	–1.0	–1.5	–1.5	5.6527095
	–0.790062	2.049713	1.850909	–0.790062	–1.762835	–4.174578	4.6335946
2	–0.774225	1.677623	1.797606	–0.774225	–1.703145	–3.078755	0.7948358
3	–0.766745	1.583584	1.852772	–0.766745	–1.701676	–2.935227	0.0456123
4	–0.763835	1.594395	1.829487	–0.763835	–1.716065	–2.920504	0.0010581
5	–0.763588	1.596304	1.8262955	–0.763588	–1.717656	–2.919715	1.8225101E-05
6	–0.763670	1.595433	1.827691	–0.763670	–1.716904	–2.919703	3.589456E-06
7	–0.763631	1.595841	1.827036	–0.763631	–1.717257	–2.919701	7.894632E-07
8	–0.763649	1.595648	1.827345	–0.763649	–1.717090	–2.919701	1.749312E-07
9	–0.763640	1.595739	1.827199	–0.763640	–1.717169	–2.919700	3.9510649E-08
10	–0.763644	1.595696	1.827268	–0.763644	–1.717132	–2.919700	8.905773E-09

iter r	x_1	x_2		x_4	x_5	f	$\|g(x_r)\|$
0	-1.0	1.0	1.0	-1.0	-1.5	-1.5	5.6527095
1*	-0.79006	2.049713	1.850909	-0.790062	-1.762835	-4.174578	4.6335946
2	-0.787749	1.634949	1.888268	-0.787749	-1.645451	-3.152304	0.9343125
3	-0.764252	1.592170	1.833953	-0.764252	-1.715895	-2.926449	0.0192923
4	-0.763593	1.596322	1.826285	-0.763593	-1.717665	-2.919788	1.0934702E-04
5	-0.763667	1.595463	1.827643	-0.763667	-1.716930	-2.919703	3.434885E-06
6	-0.763632	1.595826	1.827059	-0.763632	-1.717244	-2.919701	6.269344E-07
7	-0.763648	1.595655	1.827344	-0.763648	-1.717096	-2.919701	1.3886597E-07
8	-0.763641	1.595736	1.827204	-0.763641	-1.717167	-2.919700	3.1289974E-08
9	-0.763644	1.595697	1.827265	-0.763644	-1.717133	-2.919700	6.994948E-09
10	-0.763643	1.595716	1.827237	-0.76343	-1.717149	-2.919700	1.39671042E-09

TABLE 3. Technique II, $B_o = I$

	x_1	x_2	x_3	x_4	x_5	f	$\|g(x_r)\|$
	-1.0	1.0	1.0	-1.0	-1.5	-1.5	5.6527095
	-0.796529	1.933312	2.031977	-0.796529	-1.711102	-4.264811	3.8953451
	-0.788279	1.539120	2.021176	-0.788279	-1.574045	-3.042657	0.7662572
	-0.764460	1.652087	1.748291	-0.764460	-1.782437	-3.008636	0.2052846
	-0.768515	1.568253	1.876148	-0.768515	-1.694284	-2.944246	0.0336552
	-0.762995	1.609554	1.806578	-0.762995	-1.728900	-2.92668	0.008598
	-0.764373	1.589582	1.837403	-0.764373	-1.711810	-2.921146	0.0018064
	-0.763389	1.598687	1.822539	-1.763389	-1.719708	-2.920023	4.010699E-04
	-0.763783	1.594323	1.829486	-0.763783	-1.715914	-2.919772	8.937683E-05
	-0.763582	1.596368	1.826193	-0.763582	-1.717712	-2.919716	1.989133E-05
	-0.763674	1.595400	1.827743	-0.763673	-1.716876	-2.919704	4.42576E-06
	-0.763628	1.595856	1.827011	-0.763631	-1.717270	-2.919701	9.848276E-07
12	-0.763656	1.595641	1.827356	-0.763644	-1.717084	-2.919701	2.192526E-07
13	-0.763620	1.595743	1.827193	-0.763620	-1.717172	-2.919700	4.945792E-08

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 $B_o = |f(x_o) + \|g(x_o)\||\,.\,I$. 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 $(y_{r+1})_i$ $\forall i$ when $(y_r)_i$ approaches zero.

The numerical experimentation showed that the best choice of $(y_{r+1})_i$ is the following:

$$(y_{r+1})_i = \sqrt{-2g_i(x_r)} \qquad \text{if } g_i(x_r) < 0 \qquad (30)$$

and

$$(y_{r+1})_i = \sqrt{2g_i(x_r)} \qquad \text{if } g_i(x_r) \geq 0$$

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 $(y_{r+1})_s = 0$

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

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 $(\lambda_r)_i$, $i = m + 1, \ldots, p$, are positive. To maintain positive definiteness of this matrix we replace every component i of the diagonal matrix $\Lambda^{-1} Y^2$, given as $(\frac{y_r^2}{\lambda_r})_i$ by $(\frac{y_r^2}{-|\lambda_r|})_i$, $i = m + 1, \ldots p$

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 λ_1 and λ_2, of the function $f = f(x_1, x_2, x_3)$ and of the Euclidean norm of the constraints, $\|g(x_r)$, at $r = 0,1,2 \ldots$

TABLE 4. Technique II, ($y_1 = 0$ since $g_1(x) = 0$)
up: $B_o = |f(x_o) + \|g(x_o)\| | . I$
down: $B_o = I$

iter r	x_1	x_2	x_3	y_2	λ_1	λ_2	f	$\|g(x_r)\|$
0	0.0	0.0	0.0	0.0	0.0	0.0	0.	4.4721360
1	0.0	4.0	3.0	0.00000	25.5968	7.7082	102	9
2	0.0	1.749998	1.874998	0.00000	11.9990	0.5603	15.23433	1.2656155
3	0.0	1.309782	1.654891	0.00190	3.6431	–2.6585	8.987812	0.04844707
4	0.0	1.291532	1.645768	0.00000	4.2105	–4.2750	8.781994	8.3327046E-05
5	0.0	1.291503	1.645751	0.00000	4.2218	–4.2801	8.781660	7.6413742E-09
6	0.0	1.291503	1.645751	0.00006	4.2218	–4.2801	8.781660	5.1511994E-09
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	4.4721360
1	0.0	4.0	2.750	1.0	25.0378	7.1492	93.5	7.5790109
2	0.0	1.986987	1.741002	1.0	12.8890	0.0356	17.22937	1.136436
3	0.0	1.297891	1.647891	1.0049	3.0923	–0.0179	8.806900	0.0090675
4	0.0	–0.299118	2.128376	0.0	–0.1658	–0.0543	4.637609	2.5662748
5	0.0	1.365959	1.682979	0.0	3.7972	–5.3707	9.646309	0.198378
6	0.0	1.292019	1.646010	0.00005	4.0893	–4.2864	8.787431	0.001366
7	0.0	1.291503	1.645751	0.00000	4.2218	–4.2801	8.781660	6.2018629E-08
8	0.0	1.291503	1.645751	0.00005	4.2218	–4.2801	8.781660	6.8981446E-09
9	0.0	1.291503	1.645751	0.00000	4.2218	–4.2801	8.781660	7.1635815E-09

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

$$B_o = |f(x_o) + \|g(x_o)\| | . I$$

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

$$B_o = I ,$$

I is the unit matrix of order n.

The algorithm was also applied to Rosen-Suzuki Problem[13].

The matrix B_o is selected to be

$$B_o = |f(x_o) + \|g(x_o)\| | . I$$

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

$$g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \le 0 ,$$

$$g_2(x) = x_1^2 + 2\,x_2^2 + x_3^2 + 2\,x_4^2 - x_1 - x_4 - 10 \le 0 ,$$

$$g_3(x) = 2\,x_1^2 + x_2^2 + x_3^2 + 2\,x_1 - x_2 - x_4 - 5 \le 0$$

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,
$B_o = |f(x_o) + \|g(x_o)\||$

iter r	x_1	x_2	x_3	x_4	y_1	y_2	y_3	f	$\|g(x_r)\|$
0	0	0	0	0	$\sqrt{16}$	$\sqrt{20}$	$\sqrt{10}$	0	13.7477
1*	0.290957	0.872872	0.290957	−0.290957	4.	4.4721	3.1623	−12.8654	11.439899
2	0.168968	0.826310	0.515102	−0.465481	3.7374	4.0342	2.8228	−17.5932	10.774924
3	−0.043453	0.794579	2.738494	−1.597991	3.6009	3.9015	2.8347	−54.2644	9.136067
4	0.219037	0.950505	3.000186	−1.579063	0.0	3.3204	2.7752	−58.4579	12.549073
5	−0.105412	0.758823	2.103292	−1.248170	0.0	3.7955	3.4834	−45.1809	1.099245
6	0.120099	0.908280	2.056588	−0.942685	0.0	0.3347	0.7751	−44.7419	1.55730
7	0.309217	1.159072	2.064798	−0.717407	0.0	1.7356	0.0	−45.2435	1.809037
8	0.028190	0.891998	2.073131	−0.969846	0.0	1.7416	0.0	−44.4928	1.324838
9	−0.005525	0.956735	2.018494	−0.987516	0.0	1.6045	0.0	−44.0178	1.151626
10	−0.000056	1.003201	1.999838	−1.000118	0.0	1.5176	0.0	−44.0078	0.987189
11	−0.000470	0.999212	2.000565	−0.999483	0.0	1.4051	0.0717	−44.0000	1.003007
12	0.000861	1.000582	2.000305	−0.999010	0.0	1.4163	0.0058	−44.0051	1.002260
14	−0.000126	0.999831	2.000138	−0.999868	0.0	1.4163	0.0	−44.0000	1.000656
17	0.000003	1.000077	1.999976	−1.000014	0.0	1.4147	0.0	−44.0000	0.999724

TABLE 6. Technique II,
$B_o = |f(x_o) + \|g(x_o)\||$

iter r	x_1	x_2	x_3	x_4	y_1	y_2	y_3		$\|g(x_r)\|$
0	0.0	0.0	0.0	0.0	$\sqrt{16}$	$\sqrt{20}$	$\sqrt{10}$	0	13.7477
1*	−0.290957	0.872872	0.290957	−0.290957	4.0000	4.4721	3.1623	−12.8654	11.439899
2	0.227057	0.982069	1.342288	−0.133672	3.7374	4.0342	2.8228	−30.5384	8.091270
3	0.463250	1.346000	2.216928	−0.312471	2.9806	3.5424	2.2467	−45.8355	2.4724919
4	0.056752	0.461095	3.094950	−0.429753	0.0	1.5509	0.0	−51.0335	7.097985
5	−0.209451	1.001272	2.369861	−0.672923	0.0	1.2244	3.1241	−45.7049	1.454993
6	−0.107105	1.495515	1.877370	−1.246395	0.0	1.0457	0.0	−44.1916	2.6103846
7	−0.114579	0.989740	2.149807	−0.923538	1.2882	0.0	0.7835	−44.8975	0.859676
8	−0.041360	0.989593	2.032260	−1.191543	0.0	1.1507	0.8150	−45.0985	0.7772435
9	−0.033777	1.028681	2.072073	−0.904652	0.0	0.5707	0.6812	−44.3559	1.0302519
10	0.023261	1.113711	2.000831	−0.953743	0.0	1.4239	0.0	−44.2213	0.7770819
11	0.001857	0.968718	2.013420	−0.994821	0.0	1.2375	0.0	−44.0626	1.0974758
12	0.002199	1.013053	1.995288	−1.003489	0.0	1.4812	0.2102	−44.0061	0.9510014
13	0.003252	0.997410	2.007843	−0.986850	0.0	1.3791	0.0677	−44.0444	1.047794
14	0.002902	1.002489	1.998196	−1.001312	0.0	1.4475	0.0	−44.0051	0.993577
15	−0.000162	0.986744	2.003934	−0.997951	0.0	1.4097	0.0	−44.0001	1.046997
18	0.000293	1.000646	1.999756	−1.000097	0.0	1.4187	0.0	−44.0007	0.998206

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 λ is (–0.9999, –1.796E-09, –2.0000).

At the solution, we have

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

We note that the negative sign of λ_2 of Table (4) and that for λ_1, λ_2, λ_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

[1] **Wilson, R.B.**, *A Simplicial Method for Convex Programming,* Ph.D. thesis, Harvard University, Cambridge, Ma. (1963).

[2] **Garcia-Palomares, U.M.** and **Mangasarian, O.L.**, Superlinearly convergent quasi-Newton algorithms for nonlinearly constrained optimization problems, *Mathematical Programming,* **11:** 1-13 (1976).

[3] **Han, S.P.**, A globally convergent method for nonlinear programming, *Journal of Optimization Theory and Applications,* **22:** 297-309 (1977).

[4] **Powell, M.J.D.**, *in:* **Mangasarian, O.L.**, **Meyer, R.R.** and **Robinson, S.M.** (**eds.**), *Nonlinear Programming,* Academic Press, New York, **3:** 27-63 (1978).

[5] **Powell, M.J.D.**, How bad are the BFGS and DFB methods when the objective function is quadratic?, *Mathematical Programming,* **34:** 34-47 (1986).

[6] **Tapia, R.A.**, On the role of slack variables, in quasi-Newton methods for constrained optimization, *in:* **Dixon, L.C.W.** and **Szego, G.P.** (**eds.**), *Numerical Optimization of Dynamic Systems,* North-Holland Publishing Company, 235-246 (1980).

[7] **Dennis, J.E.** and **Schnabel, R.B.**, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice Hall, Inc., Englewood, Cliffs, New Jersey (1983).

[8] **Gill, P.E.**, **Murray, W.** and **Wright, M.H.**, *Practical Optimization,* Academic Press, London (1981).

[9] **Powell, M.J.D.**, A fast algorithm for nonlinearly constrained optimization calculations, *in:* **Watson, G.A.**, (**ed.**), *Numerical Analysis,* Springer-Verlag, Berlin, 144-157 (1978).

[10] **Fletcher, R.**, *Practical Methods of Optimization, Constrained Optimization,* Vol. 2, John Wiley & Sons, Chichester (1981).

[11] **Gill, P.E.** and **Murray, W.**, Newton-type methods for unconstrained and linearly constrained optimization, *Mathematical Programming,* **7**: 311-350 (1974).

[12] **Powell, M.J.D.**, Algorithms for nonlinear constraints that use Lagrangian functions, *Mathematical Programming,* **14**: 224-248 (1978).

[13] **Yamashita, H.**, A globally convergent quasi-Newton method with an augmented Lagrangian type penalty function, *Mathematical Programming,* **23**: 75-86 (1982).

[14] **Bertocchi, M.**, **Cavali, E.** and **Spedicato, E.**, *in:* **Dixon, L.C.W.** and **Szego, G.P.**, **(ed.)**, *Numerical Optimization of Dynamic System,* North-Holland Publishing Company, 247-268 (1980.)

طريقة عددية لحل مشاكل الحل الأمثل ذات القيود باستخدام الطريقة المشابهة لطريقة نيوتن

حسن علي أحمد

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

يقدم البحث طريقة عددية لإيجاد حل موضعي أمثل لمشكلة برمجة غير خطية تحوى قيوداً على شكل متباينات . وفي هذه الطريقة تضاف إلى المتباينات أنصاف مربعات متغيرات ظل لتحويلها إلى معادلات وهو الأسلوب الذي اتبعه تابيا سنة ١٩٨٠م . ولكن يضاف إليه استخدام الطريقة شبه النيوتنية مع طريقة عددية مستقرة تعتمد على تعديل المصفوفة المتماثلة ذات التفاضل الثاني لدالة لاجرانج لكي تصبح مصفوفة إيجابية التعريف . كما يحدث تعديل في قطر المصفوفة المتكونة نتيجة إضافة متغيرات الظل لضمان استمرار إيجابية التعريف للمصفوفة أثناء الحل ، كذلك يحدث تعديل في المصفوفة المقربة لمصفوفة التفاضل الثاني عند بداية التقريب .

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

وقد طبقت الطريقة في حالة المعادلات والمتباينات وأعطيت أمثلة لذلك .