Square-Root Variable-Metric Methods for Minimization

D. G. Hull¹ and B. D. Tapley²

Communicated by H. Y. Huang

Abstract. Variable-metric methods are presented which do not need an accurate one-dimensional search and eliminate roundoff error problems which can occur in updating the metric for large-dimension systems. The methods are based on updating the square root of the metric, so that a positive-definite metric always results. The disadvantage of intentionally relaxing the accuracy of the one-dimensional search is that the number of iterations (and hence, gradient evaluations) increases. For problems involving a large number of variables, the square-root method is presented in a triangular form to reduce the amount of computation. Also, for usual optimization problems, the square-root procedure can be carried out entirely in terms of the metric, eliminating storage and computer time associated with computations of the square root of the metric.

Key Words. Nonlinear programming, variable-metric methods, parameter optimization, function minimization, mathematical programming.

1. Introduction

Over the last decade, considerable attention has been directed at the problem of minimizing nonlinear systems. When the function being minimized is highly nonlinear or ill-conditioned, the standard gradient or steepest-descent methods yield poor convergence characteristics. The Davidon-Fletcher-Powell method (Ref. 1) is the most widely adopted method for solving such problems and is acknowledged to have the best convergence characteristics.

The general problem which is considered can be described as follows: Determine the value of the \( n \)-vector \( x \) such that the scalar function value

1. Associate Professor, Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, Austin, Texas.
2. Professor and Chairman, Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, Austin, Texas.

© 1977 Plenum Publishing Corp., 227 West 17th Street, New York, N.Y. 10011. To promote free access to published material in the spirit of the 1976 Copyright Law, Plenum sells reprints articles from all its journals. This availability underlines the fact that no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, microfilming, recording, or otherwise, without written permission of the publisher. Shipment is prompt; rate per article is $7.50.
\( G(x) \) is minimized. In the subsequent discussion, it is assumed only that the function \( G \) and \( G_x \), exist for all \( x \) in the range of interest. The standard DFP method for solving this problem is described by the following algorithm.

**Step 1.** Guess \( x \) and a positive-definite, symmetric matrix \( H \) (usually \( H = I \), where \( I \) is the identity matrix). Compute \( G(x) \) and \( G_x(x) \).

**Step 2.** Compute the displacement \( \Delta x \) from the formula
\[
\Delta x = -\alpha H G_x^T;
\]  
(1)

use a one-dimensional search to determine the scalar stepsize \( \alpha \) which minimizes \( G \) in the \(-H G_x^T\) direction; the search yields the new function \( \bar{G}_x \), where
\[
\bar{G}_x = G(x + \Delta x).
\]

**Step 3.** Compute a new gradient \( \bar{G}_x^T \), and form the difference
\[
\Delta G_x^T = \bar{G}_x^T - G_x^T.
\]

**Step 4.** Compute a new metric \( \bar{H} \) from the relation
\[
\bar{H} = H + \Delta x \Delta x^T/\Delta x^T \Delta G_x^T - H \Delta G_x^T \Delta G_x H/\Delta G_x H \Delta G_x^T.
\]  
(2)

**Step 5.** If the process has not converged, repeat the procedure.

Two problems can arise with this algorithm. First, if the one-dimensional search is not conducted very accurately, the \( H \)-matrix can become nonpositive-definite. Second, if the problem involves a large number of variables and if \( H \) is nearly singular at some point, roundoff errors in the computation of Eq. (2) can cause \( \bar{H} \) to become nonpositive-definite. In either case, a decrease in the function \( G \) is no longer guaranteed, and the iteration process must be restarted with \( H = I \).

To allow a reduction in the accuracy with which the one-dimensional search must be made, Williamson (Ref. 2) has observed that the update formula (2) can be expressed in a square-root form, so that \( \bar{H} \) will always be positive-definite. This procedure, which also eliminates the roundoff problem for large systems, is based on an equivalent form of Eq. (2), that is, the well-known rank-one formula (see, for example, Ref. 3):
\[
\bar{H} = H - \left(1/P\right) H y y^T H,
\]  
(3)

where
\[
P = y^T H \Delta G_x^T, \quad y = \alpha G_x^T + \Delta G_x^T.
\]  
(4)

The equivalence of the two formulas has been established by Huang in Ref. 4.
The purpose of this paper is (i) to explore the extent to which the accuracy of the one-dimensional search can be relaxed using the square-root update, (ii) to reduce the amount of computer time required per iteration for large systems by modifying the square-root update, and (iii) to improve the square-root formulation for problems where roundoff error is not a factor.

2. Square-Root Update

As discussed by Williamson (Ref. 2), the algorithm developed by Potter (Ref. 5) for updating the covariance matrix can be used to ensure that \( H \) remains positive definite, regardless of the accuracy with which the one-dimensional search is conducted. This is achieved by writing \( H = SS^T \), where \( S \) is an \( n \times n \) matrix, and finding the update formula for \( S \). It should be noted that, regardless of the formula used for \( S \), \( H \) will be positive-definite and symmetric.

To determine \( S \), Eq. (3) is rewritten as

\[
\bar{S}S^T = S[I - (1/P)S^Tyy^TS]S^T. \tag{5}
\]

A solution for the term in brackets is sought such that

\[
I - (1/P)S^Tyy^TS = AA^T, \tag{6}
\]

where the matrix \( A \) is defined as

\[
A = I - (\nu/P)S^Tyy^TS. \tag{7}
\]

The constant \( \nu \) is to be determined such that the assumed relation holds. Carrying out the multiplication on the right-hand side of Eq. (6) leads to the following expression for \( \nu \):

\[
\nu = [1 \pm (1-Q/P)^{1/2}]/(Q/P), \tag{8}
\]

where

\[
Q = \nu^TSS^Ty. \tag{9}
\]

Either sign can be used, but the negative sign is chosen, because it makes \( \nu \) continuous in the region of \( Q/P = 0 \). As a consequence, the update formula for \( S \) becomes

\[
\bar{S} = S[I - (\nu/P)S^Tyy^TS], \tag{10}
\]

and \( H \) is obtained through the multiplication

\[
\bar{H} = \bar{S}\bar{S}^T \tag{11}
\]
It is seen from Eq. (8) that the existence of $S$ depends on the value of the quantity $Q/P$. If

$$Q/P < 1,$$

$\nu$ is real, and an $S$ exists. If

$$Q/P = 1,$$

$S$ exists, but it is singular. Finally, if

$$Q/P > 1,$$

$\nu$ is imaginary, so that there is no real $S$-matrix. When this happens, a real $S$-matrix must be generated to maintain a positive-definite $H$-matrix. As $Q/P$ varies in the range

$$-\infty \leq Q/P \leq 1,$$

$\nu$ takes on the values

$$0 \leq \nu \leq 1.$$

Hence, as proposed by Williamson (Ref. 2), a reasonable approach for the case where $Q/P > 1$ is to use $\nu = 1$ and the computed value of $P$ in Eq. (10) for the ratio $\nu/P$. If $Q/P = 1$, a value of $\nu$ such that $\nu < 1$ should be chosen to prevent $S$ from being singular. An alternate approach is to set $S$ equal to the identity matrix. One test with Rosenbrock's function showed that the latter procedure required 39 iterations to achieve convergence, whereas the $\nu = 1$ procedure needed only 24 iterations. The explanation for this behavior is that the $S$-matrix generated with $\nu = 1$ retains some information about the geometry of the function $G$, whereas using $S = I$ is the same as reinitializing the problem.

3. Numerical Examples

To investigate the extent to which the accuracy of the one-dimensional search can be relaxed, the following three example problems are solved.

**Example 3.1**
Rosenbrock function: $G = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$;
Minimum value: $G = 0$ at $x_1 = 1, x_2 = 1$;
Starting point: $x_1 = -1.2, x_2 = 1.0$.

**Example 3.2**
Wood function: $G = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_2 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1[(x_3 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1) \cdot (x_4 - 1)$;
Minimum value: \( G = 0 \) at \( x_1 = 1, \ x_2 = 1, \ x_3 = 1, \ x_4 = 1 \);
Starting point: \( x_1 = -3, \ x_2 = -1, \ x_3 = -3, \ x_4 = -1 \).
This function has the nonminimal stationary point
\[
G = 7.88 \text{ at } x_1 = -0.968, \ x_2 = 0.947, \ x_3 = -0.970, \ x_4 = 0.951.
\]

**Example 3.3**
Miele function: 
\[ G = (\exp(x_1) - x_2)^4 + 100(x_2 - x_3)^6 + \tan^4(x_3 - x_4) + x_1^8 + (x_4 - 1)^2; \]
Minimum value: \( G = 0 \) at \( x_1 = 0, \ x_2 = 1, \ x_3 = 1, \ x_4 = 1 \);
Starting point: \( x_1 = 1, \ x_2 = 2, \ x_3 = 1, \ x_4 = 2 \).

In solving these problems, three methods for determining the stepsize \( \alpha \) are used: (i) a standard one-dimensional search, (ii) scaling, and (iii) percent correction. Also, the gradient is computed numerically by forward differences using a relative increment of \( 10^{-6} \); convergence is assumed to occur when the change in \( G \) is less than \( 10^{-6} \).

3.1. **Standard One-Dimensional Search.** The one-dimensional search used here consists of a grid search (increasing or decreasing the interval by a factor of five) to bracket the minimum, followed by a sequence of parabolic interpolations to locate the minimum. The first grid point is computed from the relation
\[ \alpha = 2(G_{LB} - G_{a=0})/(dG/d\alpha)_{a=0}, \tag{12} \]
where \( G_{LB} \) is an estimate of the lower bound of \( G \) (here, \( G_{LB} = 0 \) for all examples). Another feature of the search is that, if the middle point gets within five percent of the interval of either endpoint, the parabolic interpolation is abandoned, and the grid search is restarted using an interval factor of two. The search is stopped when the relative difference between the parabola minimum and the function value at the parabola minimum is less than the search tolerance. The absolute difference is used when the value of \( G \) is less than \( 10^{-6} \). Because of the procedure employed, at least three function evaluations are needed to find the minimum.

The results are shown in Table 1, where RS means the number of restarts with \( \nu = 1 \), GE means the number of gradient evaluations (equals the number of iterations), and FE means the number of function evaluations in the one-dimensional search. There are two observations which can be made based on this data. First, convergence is achieved regardless of the search tolerance. Second, while the number of search function evaluations decreases with increasing tolerance, the number of gradient evaluations (iterations) increases. Hence, if the problem involves a large number of variables, so that the gradient is expensive to evaluate, it is better to do an accurate one-dimensional search.
Table 1. Standard one-dimensional search.

| Search tolerance | Example 3.1 | | Example 3.2 | | Example 3.3 | |
|------------------|-------------|-------------|-------------|-------------|-------------|
|                  | RS | GE | FE | RS | GE | FE | RS | GE | FE |
| $10^{-1}$        | 9  | 39 | 152 | 20 | 66 | 294 | 4  | 23 | 85 |
| $10^{-2}$        | 5  | 28 | 147 | 24 | 69 | 370 | 1  | 19 | 95 |
| $10^{-4}$        | 1  | 20 | 147 | 11 | 47 | 314 | 0  | 17 | 118|
| $10^{-6}$        | 2  | 23 | 227 | 13 | 41 | 453 | 0  | 17 | 168|

3.2. Scaling. Scaling is the name given to the process of decreasing $\alpha$ by some factor until a decrease in $G$ is achieved. Two procedures have been used to select the first value of $\alpha$. One is to set $\alpha = 1$, and the other is to compute it from Eq. (12). Convergence is achieved in all cases as shown in Table 2. Also, the results indicate that it is better to use $\alpha = 1$ for the first $\alpha$ than that given by Eq. (12); however, this has not been found to be true in general.

3.3. Percent Correction. While scaling can be thought of as a rough search, the percent correction approach cannot. Here, the philosophy is to require that $\|\Delta x\|/\|x\|$ be less than some percent (PCT) of the current $\|x\|$, that is,

$$\frac{\|\Delta x\|}{\|x\|} \leq \text{PCT}.$$  

(13)

In view of Eq. (1), the value of $\alpha$ is obtained from the relation

$$\alpha = \min(\alpha_*, 1),$$  

(14)

where

$$\alpha_* = \text{PCT} \frac{\|x\|}{\|\Delta x\|_{\alpha = 1}}.$$  

(15)

This procedure does not require that $G$ decrease on every iteration. The results presented in Table 3 for different values of PCT show that convergence is achieved for Examples 3.1 and 3.3. Example 3.2 is the function

Table 2. Scaling.

| $\alpha$     | Example 3.1 | | Example 3.2 | | Example 3.3 | |
|--------------|-------------|-------------|-------------|-------------|-------------|
| $\alpha = 1$ | RS | GE | FE | RS | GE | FE | RS | GE | FE |
| 10           | 10 | 37 | 95 | 1  | 22 | 59 | 0  | 16 | 22 |
| Eq. (12)     | 8  | 40 | 98 | 19 | 76 | 428| 6  | 30 | 104|
Table 3. Percent correction.

<table>
<thead>
<tr>
<th></th>
<th>Example 3.1</th>
<th></th>
<th>Example 3.2</th>
<th></th>
<th>Example 3.3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PCT</td>
<td>RS</td>
<td>GE</td>
<td>FE</td>
<td>RS</td>
<td>GE</td>
<td>FE</td>
</tr>
<tr>
<td>0.50</td>
<td>3</td>
<td>48</td>
<td>48</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>0.75</td>
<td>5</td>
<td>28</td>
<td>28</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1.00</td>
<td>1</td>
<td>92</td>
<td>92</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

with a nonminimal stationary point. In the neighborhood of this point, the gradient is very small, and a very small step is predicted with Eq. (1). Hence, the percent correction procedure stalls at such a point because very small steps are allowed. The iteration procedure was not continued past 200 iterations to see if convergence would ultimately occur.

4. Triangular Square-Root Update

For large systems, the computation time for the square-root update can be reduced by employing an update formula in which $S$ is defined by an analytic Cholesky decomposition (see Ref. 6). Hence, $S$ is lower triangular, that is, all elements above the diagonal are zero. The triangular matrix representation for $S$ leads to a reduction in the amount of computation time required to generate the $H$-matrix.

The derivation of the method begins with Eq. (5) rewritten in the form

$$SS^T = S[I - (1/P)zz^T]S^T,$$

(16)

where

$$z = S^Ty.$$  

(17)

Next, it is assumed that the term within brackets can be written as the product of a lower-triangular matrix $B$ and its transpose, that is,

$$I - (1/P)zz^T = BB^T.$$  

(18)

Finally, this expression is expanded in scalar form, so that the elements $b_i$ of the $B$ matrix can be determined. If

$$\beta_1 = P,$$

$$\beta_i = P - \sum_{k=1}^{i-1} \frac{1}{z_k^2}, \quad i = 2 \rightarrow n + 1,$$

$$\beta_i = P - \sum_{k=1}^{i-1} \frac{z_k^2}{z_k^2}, \quad i = 2 \rightarrow n + 1.$$
the expressions for $b_{ij}$ are given by
\[
\begin{align*}
  b_{ii} &= \left(\beta_{i+1}/\beta_i\right)^{1/2}, & i &= 1 \rightarrow n, \\
  b_{ij} &= -z_i z_j \left[\beta_i (\beta_{i+1}/\beta_i)^{1/2}\right], & i &= 2 \rightarrow n, & j &= 1 \rightarrow i - 1.
\end{align*}
\] (20)

As long as the $\beta$'s all have the same sign (either positive or negative), a $B$-matrix can be determined, and the $H$-matrix will be positive-definite. On the other hand, if the $\beta$'s change sign, some of the elements of $B$ become imaginary, which will lead to a nonpositive-definite matrix $H$. To generate a positive-definite $H$-matrix, a constant is added to or subtracted from all of the $\beta$'s, depending on whether $\beta_1 = P$

is positive or negative, and the computation is continued. If the constant is chosen such that $\beta_{n+1} = 0$,

the relation

$\beta_{n+1} = P - Q$

indicates that $P$ is being set equal to $Q$, causing $H$ to become singular; the iteration process will stop prematurely. Hence, a value is chosen such that $\beta_{n+1} \neq 0$.

The update procedure has been tested on Example 3.1, and it yields essentially the same numerical results as those in Table 1.

5. Modified Square-Root Update

For optimization problems where the number of variables is small enough that roundoff error is not a factor in the computation of the $H$-matrix, the update procedure based on Eqs. (10) and (11) can be carried out entirely in terms of $H$. Performing the multiplication in Eq. (11) analytically leads to the relation

$H = H - (1/R)HYy^TH,$

(21)

where

$R = (P^2/\nu)/(2P - Q\nu).$

(22)

For real values of $\nu$ given by Eq. (8), the expression for $R$ reduces to $R = P$, as it should. If $\nu$ is imaginary, the values to be used in the expression for $R$ are $\nu = 1$ and $P, Q$ as computed.
The elimination of the $S$-matrix yields a reduction in both computer storage and processing time.

6. Conclusions

An inaccurate one-dimensional search for usual systems or roundoff error for large systems can lead to the formation of a nonpositive-definite metric $H$. When this occurs, the iteration procedure stops. One solution is to restart the iteration process with $H$ equal to the identity matrix. Another and more economical solution is to reformulate the update formula in terms of the matrix $S$, which is the square root of the $H$-matrix. Then, regardless of how accurately $S$ is calculated, the $H$-matrix will be positive-definite. In addition, when a restart is indicated, the $S$-matrix which is used still contains some information about the function being minimized.

The main disadvantage of intentionally relaxing the accuracy of the one-dimensional search, or eliminating it altogether, is that the number of iterations required to achieve convergence is increased. Hence, if the gradient is expensive to compute, an accurate one-dimensional search is needed to keep the number of gradient evaluations at a minimum.

For large systems, the computation time associated with the square-root update can be reduced by employing a lower-triangular formulation for the $S$-matrix.

For usual systems where roundoff error is not a problem, the square-root update can be performed entirely in terms of the $H$-matrix. Doing so saves both the computer time and storage normally required to manipulate the $S$-matrix.

References