NUMERICAL EXPERIMENTS ON THE
METHODS OF DUAL MATRICES FOR FUNCTION MINIMIZATION

by

JOE P. CHAMBLISS

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

Thesis Director's Signature:

Houston, Texas
May, 1972
ABSTRACT

Numerical Experiments on the Methods
of Dual Matrices for Function Minimization

by

JOE P. CHAMBLISS

Four algorithms of dual matrices for function minimization introduced in Ref. 1 are tested through several numerical examples. Three quadratic functions and five nonquadratic functions are investigated. For quadratic functions, the results show that the convergence is achieved in at most n+1 iterations, where n is the number of variables. Since one-dimensional search is not needed in these algorithms the total number of gradient evaluations for convergence is at most n+2. This represents a saving on the gradient evaluations versus 2n+1 required by the conventional quadratically convergent algorithms. For nonquadratic functions, the results show that these algorithms are very stable and efficient. Also, the effects of stepsize factor on these algorithms are investigated.
ACKNOWLEDGEMENT

The author is indebted to Dr. H. Y. Huang for suggesting the topic and helpful discussions. This research was supported by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No. AF-AFOSR-828-67.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Algorithms of Dual Matrices</td>
<td>3</td>
</tr>
<tr>
<td>3. Reverse Algorithm</td>
<td>7</td>
</tr>
<tr>
<td>4. Stepsize Factor</td>
<td>9</td>
</tr>
<tr>
<td>5. Experimental Conditions</td>
<td>12</td>
</tr>
<tr>
<td>6. Quadratic Examples and Numerical Results</td>
<td>14</td>
</tr>
<tr>
<td>7. Quadratic Examples: Discussion</td>
<td>22</td>
</tr>
<tr>
<td>8. Nonquadratic Examples and Numerical Results</td>
<td>23</td>
</tr>
<tr>
<td>9. Nonquadratic Examples: Discussion</td>
<td>30</td>
</tr>
<tr>
<td>References</td>
<td>32</td>
</tr>
</tbody>
</table>
1. Introduction

In a recent paper (Ref. 1), Huang introduced the method of dual matrices for the minimization of a function \( f(x) \), where \( f \) is a scalar function and \( x \) an \( n \)-vector. This method is characterized by the simultaneous use of two matrices. One matrix is such that a linearly independent set of directions can be generated. The other matrix is such that, at the point where the first matrix fails to yield a linearly independent gradient, it generates a displacement leading to the minimal point of the function. It differs from the conventional quadratically convergent algorithm (for instance, Refs. 2-3) in that the one-dimensional search for the optimal stepsize at each iteration is completely bypassed. It was proven that, for a quadratic function, the method yields the minimal point in at most \( n+1 \) iterations. Since the one-dimensional search is not needed, the total number of evaluations of the gradient vector \( g(x) \) for convergence is at most \( n+2 \). This represents a saving on the total computational effort versus \( 2n+1 \) gradient evaluations required by the conventional quadratic convergent algorithms.

In Ref. 1, three algorithms of the method were derived and a fourth one, a reverse algorithm which permits the use of only one matrix, was also given. In this report, these algorithms are tested and studied through several numerical examples. For convenience, three algorithms of dual matrices are given in Section 2 and the reverse algorithm is given in Section 3. The stepsize factor is given in Section 4. General experimental
conditions are given in Section 5. Quadratic examples and their numerical results and discussions are given in Sections 6-7, and the corresponding parts for nonquadratic examples are given in Sections 8-9.
2. **Algorithms of Dual Matrices**

An algorithm of dual matrices is defined by the following operations:

\[ p_i = A_i g_i, \quad q_i = B_i g_i \]  

\[ \Delta x_i = \begin{cases} -\mu \text{sign}(g_1^T p_i) p_i, & \text{if } |g_1^T p_i| > \epsilon_1 \\ -\mu \text{sign}(g_1^T q_i) q_i, & \text{if } |g_1^T p_i| \leq \epsilon_1 \end{cases} \]

\[ x_{i+1} = x_i + \Delta x_i \]  

where the subscript \( i \) denotes the present point and the subscript \( i+1 \) denotes the next point. Here, \( p_i \) and \( q_i \) are n-vectors; \( A_i \) and \( B_i \) are \( n \times n \) symmetric matrices; \( \Delta x_i \), an n-vector, is the displacement leading from the present point \( x_i \) to the next point \( x_{i+1} \); \( \mu \), a positive quantity, is the assigned stepsize and \( \epsilon_1 \) is some prescribed small quantity.

The sequence of operations is as follows: (a) at a given point \( x_i \), the gradient \( g_i \) is evaluated; (b) the symmetric matrices \( A_i \) and \( B_i \) either are given \underline{a priori} or are updated according to the rules to be discussed later in this section; (c) the directions \( p_i \) and \( q_i \) (when necessary) are computed according to Eqs. (1); (d) the quantity \( |g_1^T p_i| / g_1^T g_i | \) is computed and, depending on its magnitude, the displacement is given either by Eq. (2-1) or by Eq. (2-2); (e) the positive stepsize factor \( \mu \) is assigned according to the rules to be discussed in Section 4; and (f) the next point \( x_{i+1} \) is computed from Eq. (3).

Operations (a) through (f) form a complete iteration leading from the present point \( x_i \) to the next point \( x_{i+1} \).
2.1. Matrices $A_i$ and $B_i$. The symmetric matrices $A_i$ and $B_i$
are defined as follows for three different algorithms:

**Algorithm I**

$$A_0 = I, \quad A_i = A_{i-1} - \frac{A_{i-1} \Delta g_{i-1} \Delta g_{i-1}^T A_{i-1}}{\Delta g_{i-1}^T A_{i-1} \Delta g_{i-1}}$$  \hspace{1cm} (4)

$$B_0 = 0, \quad B_i = B_{i-1} + \frac{\Delta x_{i-1} \Delta x_{i-1}^T}{\Delta x_{i-1} \Delta g_{i-1}}$$  \hspace{1cm} (5)

**Algorithm II**

$$A_0 = I, \quad A_i = A_{i-1} - \frac{A_{i-1} \Delta g_{i-1} \Delta g_{i-1}^T A_{i-1}}{\Delta g_{i-1}^T A_{i-1} \Delta g_{i-1}}$$  \hspace{1cm} (6)

$$B_0 = I, \quad B_i = B_{i-1} + \frac{\Delta x_{i-1} \Delta x_{i-1}^T}{\Delta x_{i-1} \Delta g_{i-1}} - \frac{B_{i-1} \Delta g_{i-1} \Delta g_{i-1}^T B_{i-1}}{\Delta g_{i-1} \Delta g_{i-1}}$$  \hspace{1cm} (7)

**Algorithm III**

$$A_0 = I, \quad A_i = A_{i-1} - \frac{A_{i-1} \Delta g_{i-1} \Delta g_{i-1}^T A_{i-1}}{\Delta g_{i-1}^T A_{i-1} \Delta g_{i-1}}$$  \hspace{1cm} (8)

$$B_0 = I, \quad B_i = B_{i-1} + \frac{(\Delta x_{i-1} - B_{i-1} \Delta g_{i-1}) (\Delta x_{i-1} - B_{i-1} \Delta g_{i-1})^T}{(\Delta x_{i-1} - B_{i-1} \Delta g_{i-1})^T \Delta g_{i-1}}$$  \hspace{1cm} (9)

In Eqs. (4)-(9), the gradient difference $\Delta g_{i-1}$ is defined by

$$\Delta g_{i-1} = \Delta g_i - \Delta g_{i-1}$$  \hspace{1cm} (10)

and the matrices $A_0$ and $B_0$ denote the initial matrices. Thus, for the first iteration,
the initial matrices $A_0$ and $B_0$ are given. For the subsequent iterations, the matrices $A_1$ and $B_1$ are updated according to Eqs. (4)-(9).

It is noted that, while the initial matrix $B_0$ for Algorithm I must be a zero matrix, the other matrices can be in a more general form. According to the theory of Ref. 1, any positive-definite or negative-definite matrix can be used for the initial matrix $A_0$ in Eqs. (4), (6), and (8) and for the initial matrix $B_0$ in Eqs. (7) and (9). Here, for simplicity, the identity matrix is used for these occasions.

2.2. Restart. Let $x_\ell$ be the point where the inequality

$$|g_\ell^T p_\ell / g_\ell^T g_\ell| \leq \varepsilon_1$$

(11)

is satisfied. According to the algorithms, the displacement $\Delta x_\ell$ is given by Eq. (2-2). If the next point $x_{\ell+1}$ satisfies the required accuracy on the location of the minimal point, the algorithm is terminated. If the required accuracy is not achieved at the point $x_{\ell+1}$, further iterations must be carried out.

To proceed for further iterations, the matrices $A_\ell$ and $B_\ell$ must be restarted before matrices $A_{\ell+1}$ and $B_{\ell+1}$ are computed from the updating formulas (4)-(9), respectively. To this end, two different cases are distinguished. For Algorithm I, the $A$-matrix is restarted according to the rule:

$$A_\ell = \begin{cases} B_\ell, & \text{if } |g_\ell^T q_\ell / g_\ell^T g_\ell| > \varepsilon_2 \\ I, & \text{if } |g_\ell^T q_\ell / g_\ell^T g_\ell| \leq \varepsilon_2 \end{cases}$$

(12)

and the $B$-matrix is restarted by setting
\[ B_t = 0 \quad (13) \]

In Eqs. (12), \( \epsilon_2 \) is a prescribed small quantity. For Algorithms II and III, the A- and B-matrices are restarted by setting

\[ A_t = B_t, \quad B_t = B_t \quad (14) \]

Eq. (14-2) states that the B-matrix is not changed for these two algorithms.
3. **Reverse Algorithm**

The fourth algorithm of dual matrices, the reverse algorithm, is defined by the following operations:

\[ q_i = B_i g_i, \quad p_i = A_i g_i \]  \hspace{1cm} (15)

\[ \Delta x_i = -\mu \text{sign}(g_i q_i) q_i \]  \hspace{1cm} (16)

\[ x_{i+1} = x_i + \Delta x_i \]  \hspace{1cm} (17)

where the matrices \( B_i \) and \( A_i \) are defined by

\[ B_0 = I, \quad B_i = B_{i-1} + \frac{(\Delta x_{i-1} - B_{i-1} \Delta g_{i-1})(\Delta x_{i-1} - B_{i-1} \Delta g_{i-1})}{(\Delta x_{i-1} - B_{i-1} \Delta g_{i-1})^T \Delta g_{i-1}} \]  \hspace{1cm} (18)

\[ A_0 = I, \quad A_i = A_{i-1} - \frac{A_{i-1} \Delta g_{i-1} \Delta g_{i-1}^T A_{i-1}}{\Delta g_{i-1}^T A_{i-1} \Delta g_{i-1}} \]  \hspace{1cm} (19)

For the purpose of reporting the numerical results, this algorithm is called **Algorithm IV**. The sequence of operations is as follows: (a) at a given point \( x_i \), the gradient \( g_i \) is evaluated; (b) the symmetric matrices \( A_i \) and \( B_i \) are given or updated according to Eqs. (18-19); (c) the directions \( q_i \) and \( p_i \) are computed according to Eqs. (15); (d) the positive stepsze factor \( \mu \) is assigned according to the rules to be discussed in Section 4; (e) the displacement \( \Delta x_i \) is given by Eq. (16); and (f) the next \( x_{i+1} \) is computed from Eq. (17). Operations (a) through (f) form a complete iteration leading from the present point \( x_i \) to the next \( x_{i+1} \).
3.1. **Restart.** Let $x_\ell$ be the point where the inequality

$$|g_i^T p_l g_i^T g_{i'}| \leq \varepsilon_1$$

is satisfied, the displacement is $\Delta x_\ell$ is calculated from Eq. (16) and the next $x_{\ell+1}$ is obtained from Eq. (17). If the point $x_{\ell+1}$ satisfied the required accuracy on the location of the minimal point, the algorithm is terminated. On the other hand, if the required accuracy is not achieved the point $x_{\ell+1}$ further iterations must be carried out.

To proceed for further iterations, the matrices $B_\ell$ and $A_\ell$ must be restarted by setting

$$B_\ell = B_\ell, \quad A_\ell = B_\ell $$

before computing the new matrices $B_{\ell+1}$ and $A_{\ell+1}$ from Eqs. (18)-(19).

Eq. (21-1) states that the B-matrix is not changed at the restarting point.
4. **Stepsize Factor**

The positive stepsize factor $\mu$ used in Eqs. (2) for Algorithm I-III and Eq. (16) for Algorithm IV is some quantity in range

$$0 < \mu \leq \mu_0$$

(22)

where $\mu_0$ is the initial stepsize factor. For the purpose of this investigative study, four different choices of the initial stepsize factor are employed. Specifically, these are

**Scheme A:**

$$\mu_0 = \begin{cases} 
\min \left\{ 1, \frac{2(f_i - f_{lb})}{g_1^T S_1^1} \right\}, & \text{if } |g_1^T p_i / g_1^T g_1| > \epsilon_1 \\
1, & \text{if } |g_1^T p_i / g_1^T g_1| \leq \epsilon_1
\end{cases}$$

(23)

**Scheme B:**

$$\mu_0 = \min \left\{ 1, \frac{2(f_i - f_{lb})}{g_1^T S_1^1} \right\}$$

(24)

**Scheme C:**

$$\mu_0 = \begin{cases} 
\min \left\{ 10^3, \frac{2(f_i - f_{lb})}{g_1^T S_1^1} \right\}, & \text{if } |g_1^T p_i / g_1^T g_1| > \epsilon_1 \\
1, & \text{if } |g_1^T p_i / g_1^T g_1| \leq \epsilon_1
\end{cases}$$

(25)

**Scheme D:**

$$\mu_0 = \min \left\{ 10^3, \frac{2(f_i - f_{lb})}{g_1^T S_1^1} \right\}$$

(26)
In Eqs. (23)-(26), $f_i$ is the value of the function at the $i$th iteration, $f_{lb}$ is the estimated lower bound of the function, and $S_i$ is either $p_i$ or $q_i$ depending on whether the former or the latter is used for displacement.

It is noted that the initial stepsize factor $\mu_0$ has a upper limiting value of 1 for Scheme A and B while it has a upper limiting value of $10^3$ for Schemes C and D. Therefore, Schemes A and B characterize small-step process while Schemes C and D characterize large-step process.

4.1. Nonquadratic Functions. For a nonquadratic function, the enforcement on the reduction of function value at each iteration is essential for the stability of all algorithms. The reduction of function value is expressed by the inequality

$$f_{i+1} < f_i$$

meaning that the value of the function at the next point $f_{i+1}$ must be smaller than the value of the function at the present point $f_i$. Thus, starting from a initial stepsize factor $\mu = \mu_0$ given by Eqs. (23)-(26), the value of the function is evaluated. If Ineq. (27) holds, the factor is accepted. On the other hand, if Ineq. (27) is violated, the factor is reduced until Ineq. (27) is satisfied. According to the theory of Ref. 1, the displacement $\Delta x_i$ defined by Eqs. (2) or (16) guarantees the ultimate satisfaction of Ineq. (27) if the factor $\mu$ is sufficiently small.

For the purpose of this investigative research, a simple bisection process is employed to reduce the stepsize factor.
4.2. Quadratic Function. According to the theory in Ref. 1, the reduction of function value at each iteration is not required for ultimate convergence of a quadratic function. Therefore, initial stepsize factor $\mu_0$ given in Eqs. (23)-(26) can be employed without modification, that is

$$\mu = \mu_0$$  \hspace{1cm} (28)

It is true that, with such a fixed stepsize factor, the value of the function may actually increase at any iteration except the last. But, regardless of fluctuation in the values of the function, the last iteration always converges to the minimal point, barring the loss of accuracy due to round-off errors in computation.
5. **Experimental Conditions**

The four algorithms of dual matrices summarized in Sections 2 and 3 are tested through several numerical examples using a Burroughs B-5500 computer and double-precision arithmetic. The algorithms are programmed in FORTRAN IV. Further experimental conditions employed are given in the following paragraphs.

5.1. **Permanent Control Quantities.** The permanent control quantity \( \varepsilon_1 \) appearing in Eqs. (2), (11), (20), (23), and (25) is set at the value

\[
\varepsilon_1 = 10^{-8}
\]  

(29)

The quantity is used in Algorithms I - IV. For Algorithm I, the additional permanent control quantity \( \varepsilon_2 \) appearing in the restarting condition (12) is set at the value

\[
\varepsilon_2 = 10^{-7}
\]  

(30)

5.2. **Lower Bounds of Functions.** For all the examples tested, the initial choice of the stepsize factor \( \mu_0 \) is given by Eqs. (23)-(26). To employ these equations, the lower bound of a function must be given. For that matter, the lower bound is given by

\[
f_{lb} = 0
\]  

(31)

for all the examples tested.

5.3. **Stopping Condition.** The minimizing condition of a function

\[
g(x) = 0
\]  

(32)
is considered achieved if the inequality

$$T \mathbf{g}_1^T \mathbf{g}_1 \leq \varepsilon_3$$

(33)

is satisfied. In Ineq. (33), \(\varepsilon_3\) is a prescribed small quantity and defines the accuracy required of the optimal condition (32). For all the examples, the \(\varepsilon_3\) is chosen to be

$$\varepsilon_3 = 10^{-12}$$

(34)
6. Quadratic Examples and Numerical Results

In this section, three quadratic examples are given and the corresponding numerical results are presented. For simplicity, all the symbols used here are scalar quantities. For the stepsize factor, only Scheme A is employed.

Example 6.1. Minimize the function (Ref. 3)

\[
f = (x_1 + x_2 + 0.5x_4)^2 + (x_1 + 2x_2 + x_3 + x_4)^2
+ (x_2 + x_3 + 1.5x_4)^2 + (0.5x_1 + x_2 + 1.5x_3 - 0.5)^2
\]

The solution is \( f = 0 \) at the point defined by

\[
x_1 = 0.5 , \quad x_2 = -0.5 , \quad x_3 = 0.5 , \quad x_4 = 0
\]

The nominal point used for starting an algorithm is

\[
x_1 = 4 , \quad x_2 = 4 , \quad x_4 = 4 , \quad x_4 = 4
\]

The total number of iterations \( N \) and the total number of gradient evaluations \( K \) needed for convergence are

\[
N = 5 , \quad K = 6
\]
achieved. Therefore, only the largest error in the solution is indicated.

Example 6.2. Minimize the function (Ref. 3)

\[ f = (2x_1 + x_6 - 2)^2 + (2x_2 - 2x_3 + x_6)^2 + (-2x_2 + 2x_3 + x_6 + x_7)^2 \\
+ (2x_4 - 2x_5 + x_6 - 2x_7)^2 + (-2x_4 + 2x_5 + x_6 - 2x_7)^2 \\
+ (x_1 + x_2 + x_3 + x_4 + x_5 - 5)^2 + (x_3 - 2x_4 - 2x_5 + 3)^2 \] (39)

The solution is \( f = 0 \) at the point defined by

\[ x_1 = 1, \ x_2 = 1, \ x_3 = 1, \ x_4 = 1, \ x_5 = 1, \ x_6 = 0, \ x_7 = 0 \] (40)

The nominal point used for starting an algorithm is

\[ x_1 = -1, \ x_2 = -3, \ x_3 = -2, \ x_4 = -5, \ x_5 = -5, \ x_6 = 3, \ x_7 = -2 \] (41)

The total number of iterations \( N \) and the total number of gradient evaluations \( K \) needed for convergence are

\[ N = 7, \ K = 8 \] (42)

which are the same for Algorithms I-IV. The sequence of points generated by Algorithms I-III are the same and are given in Table 3. The sequence of points generated by Algorithm IV is given in Table 4.

Example 6.3. Minimize the function (Ref. 3)

\[ f = (2x_1 - 2x_2 + x_6)^2 + (-2x_1 + 4x_2 + 2x_3 + 3x_6 + x_8 - 4)^2 \\
+ (2x_2 + 2x_3 + x_7 - 4)^2 + (2x_4 + x_7 - 2)^2 + (2x_5 - 2x_7 - x_8 - 2)^2 \\
+ (x_1 + 3x_2 - 4)^2 + (x_3 + x_4 - 2x_5)^2 + (x_2 - x_5)^2 \] (43)
The solution is $f = 0$ at the point defined by

$$\begin{align*}
x_1 &= 1, \\
x_2 &= 1, \\
x_3 &= 1, \\
x_4 &= 1, \\
x_5 &= 1, \\
x_6 &= 0, \\
x_7 &= 0, \\
x_8 &= 0
\end{align*}$$

(44)

The nominal point used for starting an algorithm is

$$\begin{align*}
x_1 &= 5, \\
x_2 &= 5, \\
x_3 &= 3, \\
x_4 &= -3, \\
x_5 &= -5, \\
x_6 &= 5, \\
x_7 &= 10, \\
x_8 &= -10
\end{align*}$$

(45)

The total number of iterations $N$ and the total number of gradient evaluations $K$ for convergence are

$$N = 9, \quad K = 10$$

(46)

which are the same for Algorithms I-IV. The sequences of points generated by Algorithms I-III are the same and are given in Table 5. The sequence of points generated by Algorithm IV is given in Table 6.
### Table 1. Example 6.1, Algorithms I-III

<table>
<thead>
<tr>
<th>i</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$f$</th>
<th>$g^T g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.0000</td>
<td>4.0000</td>
<td>4.0000</td>
<td>4.0000</td>
<td>$0.828 \times 10^3$</td>
<td>$0.469 \times 10^5$</td>
</tr>
<tr>
<td>1</td>
<td>1.4737</td>
<td>-1.3351</td>
<td>0.3784</td>
<td>0.7494</td>
<td>$0.578 \times 10^0$</td>
<td>$0.284 \times 10^1$</td>
</tr>
<tr>
<td>2</td>
<td>1.3046</td>
<td>-1.3924</td>
<td>0.9180</td>
<td>0.3610</td>
<td>$0.723 \times 10^{-1}$</td>
<td>$0.827 \times 10^{-1}$</td>
</tr>
<tr>
<td>3</td>
<td>1.2841</td>
<td>-1.3144</td>
<td>0.8796</td>
<td>0.2916</td>
<td>$0.651 \times 10^{-1}$</td>
<td>$0.874 \times 10^{-1}$</td>
</tr>
<tr>
<td>4</td>
<td>1.2229</td>
<td>-1.2538</td>
<td>0.8554</td>
<td>0.2666</td>
<td>$0.568 \times 10^{-1}$</td>
<td>$0.840 \times 10^{-1}$</td>
</tr>
<tr>
<td>5</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>0.5000</td>
<td>0.0000</td>
<td>$0.160 \times 10^{-42}$</td>
<td>$0.854 \times 10^{-42}$</td>
</tr>
</tbody>
</table>

### Table 2. Example 6.1, Algorithm IV

<table>
<thead>
<tr>
<th>i</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$f$</th>
<th>$g^T g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.0000</td>
<td>4.0000</td>
<td>4.0000</td>
<td>4.0000</td>
<td>$0.828 \times 10^3$</td>
<td>$0.469 \times 10^5$</td>
</tr>
<tr>
<td>1</td>
<td>1.4737</td>
<td>-1.3351</td>
<td>0.3784</td>
<td>0.7494</td>
<td>$0.578 \times 10^0$</td>
<td>$0.284 \times 10^1$</td>
</tr>
<tr>
<td>2</td>
<td>1.3054</td>
<td>-1.3909</td>
<td>0.9189</td>
<td>0.3621</td>
<td>$0.720 \times 10^{-1}$</td>
<td>$0.678 \times 10^{-1}$</td>
</tr>
<tr>
<td>3</td>
<td>1.3048</td>
<td>-1.3046</td>
<td>0.8222</td>
<td>0.3373</td>
<td>$0.566 \times 10^{-1}$</td>
<td>$0.202 \times 10^{-1}$</td>
</tr>
<tr>
<td>4</td>
<td>1.2929</td>
<td>-1.3017</td>
<td>0.8222</td>
<td>0.3418</td>
<td>$0.549 \times 10^{-1}$</td>
<td>$0.145 \times 10^{-1}$</td>
</tr>
<tr>
<td>5</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>0.5000</td>
<td>0.0000</td>
<td>$0.290 \times 10^{-42}$</td>
<td>$0.143 \times 10^{-41}$</td>
</tr>
</tbody>
</table>
Table 3. Example 6.2, Algorithms I-III

<table>
<thead>
<tr>
<th>i</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$f$</th>
<th>$g_T$</th>
<th>$g_g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.0000</td>
<td>-3.0000</td>
<td>-2.0000</td>
<td>-5.0000</td>
<td>-5.0000</td>
<td>3.0000</td>
<td>-2.0000</td>
<td>0.991 x 10^3</td>
<td>0.401 x 10^5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.2743</td>
<td>-0.5279</td>
<td>-2.3955</td>
<td>1.2296</td>
<td>1.2296</td>
<td>1.3190</td>
<td>0.4721</td>
<td>0.693 x 10^2</td>
<td>0.275 x 10^4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.6729</td>
<td>-1.2916</td>
<td>0.1587</td>
<td>1.6609</td>
<td>1.6609</td>
<td>0.3980</td>
<td>1.2479</td>
<td>0.522 x 10^2</td>
<td>0.225 x 10^4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.1818</td>
<td>2.4092</td>
<td>3.6940</td>
<td>1.4615</td>
<td>1.4615</td>
<td>-2.1766</td>
<td>-1.0127</td>
<td>0.622 x 10^2</td>
<td>0.141 x 10^4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.6847</td>
<td>4.5002</td>
<td>4.1762</td>
<td>1.0036</td>
<td>1.0036</td>
<td>-3.9002</td>
<td>1.1657</td>
<td>0.188 x 10^3</td>
<td>0.538 x 10^4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-1.9490</td>
<td>5.2249</td>
<td>4.9300</td>
<td>1.2306</td>
<td>1.2306</td>
<td>-2.8511</td>
<td>1.3729</td>
<td>0.190 x 10^3</td>
<td>0.501 x 10^4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-1.7538</td>
<td>5.6871</td>
<td>5.1438</td>
<td>1.0877</td>
<td>1.0877</td>
<td>-2.3554</td>
<td>1.8387</td>
<td>0.192 x 10^3</td>
<td>0.503 x 10^4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>*410 x 10^{-37}</td>
<td>*448 x 10^{-36}</td>
<td></td>
</tr>
</tbody>
</table>
Table 4. Example 6.2, Algorithm IV

<table>
<thead>
<tr>
<th>i</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$f$</th>
<th>$\mathbf{g}^T \mathbf{g}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.0000</td>
<td>-3.0000</td>
<td>-2.0000</td>
<td>-5.0000</td>
<td>-5.0000</td>
<td>3.0000</td>
<td>-2.0000</td>
<td>$0.991 \times 10^3$</td>
<td>$0.401 \times 10^5$</td>
</tr>
<tr>
<td>1</td>
<td>1.2743</td>
<td>-0.5279</td>
<td>-2.3955</td>
<td>1.2296</td>
<td>1.2296</td>
<td>1.3190</td>
<td>0.4721</td>
<td>$0.693 \times 10^2$</td>
<td>$0.275 \times 10^4$</td>
</tr>
<tr>
<td>2</td>
<td>1.6614</td>
<td>-1.2989</td>
<td>0.1494</td>
<td>1.6324</td>
<td>1.6234</td>
<td>0.4091</td>
<td>1.2341</td>
<td>$0.512 \times 10^2$</td>
<td>$0.221 \times 10^4$</td>
</tr>
<tr>
<td>3</td>
<td>1.9890</td>
<td>1.8212</td>
<td>2.7546</td>
<td>1.2950</td>
<td>1.2950</td>
<td>-1.5734</td>
<td>-0.7863</td>
<td>$0.298 \times 10^2$</td>
<td>$0.685 \times 10^3$</td>
</tr>
<tr>
<td>4</td>
<td>-0.4970</td>
<td>2.4387</td>
<td>1.5371</td>
<td>0.7031</td>
<td>0.7031</td>
<td>-1.8776</td>
<td>1.2578</td>
<td>$0.712 \times 10^2$</td>
<td>$0.257 \times 10^4$</td>
</tr>
<tr>
<td>5</td>
<td>0.1265</td>
<td>1.4428</td>
<td>1.4755</td>
<td>1.1597</td>
<td>1.1597</td>
<td>0.5927</td>
<td>0.1019</td>
<td>$0.265 \times 10^1$</td>
<td>$0.266 \times 10^2$</td>
</tr>
<tr>
<td>6</td>
<td>1.0444</td>
<td>1.3718</td>
<td>1.2007</td>
<td>0.9166</td>
<td>0.9166</td>
<td>0.4112</td>
<td>0.3381</td>
<td>$0.161 \times 10^1$</td>
<td>$0.308 \times 10^2$</td>
</tr>
<tr>
<td>7</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>$0.104 \times 10^{-39}$</td>
<td>$0.611 \times 10^{-38}$</td>
</tr>
</tbody>
</table>
Table 5. Example 6.3, Algorithms I-III

<table>
<thead>
<tr>
<th>i</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_6 )</th>
<th>( x_7 )</th>
<th>( x_8 )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.0000</td>
<td>5.0000</td>
<td>3.0000</td>
<td>-3.0000</td>
<td>-5.0000</td>
<td>5.0000</td>
<td>10.0000</td>
<td>-10.0000</td>
<td>( 0.174 \times 10^4 )</td>
</tr>
<tr>
<td>1</td>
<td>5.2882</td>
<td>-0.7633</td>
<td>-0.1698</td>
<td>-3.5043</td>
<td>-2.3345</td>
<td>2.9828</td>
<td>7.5506</td>
<td>-11.4048</td>
<td>( 0.762 \times 10^3 )</td>
</tr>
<tr>
<td>2</td>
<td>-0.4852</td>
<td>0.2626</td>
<td>-1.7321</td>
<td>-4.0972</td>
<td>3.8545</td>
<td>4.0258</td>
<td>1.9845</td>
<td>-12.6391</td>
<td>( 0.551 \times 10^3 )</td>
</tr>
<tr>
<td>3</td>
<td>-20.0835</td>
<td>-4.5440</td>
<td>-5.1397</td>
<td>10.8133</td>
<td>-2.1988</td>
<td>1.6611</td>
<td>-0.6445</td>
<td>-12.2384</td>
<td>( 0.338 \times 10^4 )</td>
</tr>
<tr>
<td>4</td>
<td>-17.6400</td>
<td>-1.0105</td>
<td>-8.87551</td>
<td>16.6018</td>
<td>-3.3317</td>
<td>-0.3400</td>
<td>-4.5671</td>
<td>-13.7172</td>
<td>( 0.368 \times 10^4 )</td>
</tr>
<tr>
<td>5</td>
<td>-16.7516</td>
<td>-1.3432</td>
<td>-9.3868</td>
<td>17.6658</td>
<td>-4.1901</td>
<td>1.8097</td>
<td>-6.3910</td>
<td>-12.8151</td>
<td>( 0.372 \times 10^4 )</td>
</tr>
<tr>
<td>6</td>
<td>-16.7633</td>
<td>-1.2367</td>
<td>-9.7221</td>
<td>17.4525</td>
<td>-4.4076</td>
<td>1.6118</td>
<td>-6.7164</td>
<td>-11.9960</td>
<td>( 0.372 \times 10^4 )</td>
</tr>
<tr>
<td>7</td>
<td>-16.6764</td>
<td>-1.2481</td>
<td>-9.6418</td>
<td>17.5743</td>
<td>-4.2831</td>
<td>1.4145</td>
<td>-6.8983</td>
<td>-11.3269</td>
<td>( 0.372 \times 10^4 )</td>
</tr>
<tr>
<td>8</td>
<td>-16.5941</td>
<td>-1.2704</td>
<td>-9.4826</td>
<td>17.6632</td>
<td>-4.2164</td>
<td>1.1833</td>
<td>-7.1547</td>
<td>-10.7108</td>
<td>( 0.372 \times 10^4 )</td>
</tr>
<tr>
<td>9</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>( 0.793 \times 10^{-34} )</td>
</tr>
</tbody>
</table>
Table 6. Example 6.3, Algorithms IV

<table>
<thead>
<tr>
<th>i</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$f_{g/g}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.0000</td>
<td>5.0000</td>
<td>3.0000</td>
<td>-3.0000</td>
<td>-5.0000</td>
<td>5.0000</td>
<td>10.0000</td>
<td>-10.0000</td>
<td>0.174 x 10^4</td>
</tr>
<tr>
<td>1</td>
<td>5.2882</td>
<td>-0.7633</td>
<td>-0.1698</td>
<td>-3.5043</td>
<td>-2.3345</td>
<td>2.9828</td>
<td>7.5506</td>
<td>-11.4048</td>
<td>0.762 x 10^3</td>
</tr>
<tr>
<td>2</td>
<td>-0.3681</td>
<td>0.3637</td>
<td>-1.6311</td>
<td>-4.0738</td>
<td>3.6654</td>
<td>4.0466</td>
<td>2.1556</td>
<td>-12.5827</td>
<td>0.497 x 10^3</td>
</tr>
<tr>
<td>3</td>
<td>-11.5581</td>
<td>--2.1188</td>
<td>-3.1386</td>
<td>5.2386</td>
<td>-1.3003</td>
<td>2.6529</td>
<td>1.6646</td>
<td>-11.9927</td>
<td>0.105 x 10^4</td>
</tr>
<tr>
<td>4</td>
<td>1.1392</td>
<td>4.2290</td>
<td>-4.1548</td>
<td>4.1862</td>
<td>-1.5226</td>
<td>1.9028</td>
<td>0.8212</td>
<td>-13.0750</td>
<td>0.285 x 10^3</td>
</tr>
<tr>
<td>5</td>
<td>0.2575</td>
<td>0.9903</td>
<td>-1.6238</td>
<td>0.1016</td>
<td>-1.3748</td>
<td>5.7516</td>
<td>2.4578</td>
<td>-10.9214</td>
<td>0.424 x 10^2</td>
</tr>
<tr>
<td>6</td>
<td>-0.5208</td>
<td>1.3817</td>
<td>-1.4089</td>
<td>-1.0277</td>
<td>-0.8487</td>
<td>3.7021</td>
<td>3.7110</td>
<td>-10.8912</td>
<td>0.596 x 10^1</td>
</tr>
<tr>
<td>7</td>
<td>-0.4407</td>
<td>1.2955</td>
<td>-1.1009</td>
<td>-0.7735</td>
<td>-0.5891</td>
<td>3.6820</td>
<td>3.7954</td>
<td>-10.9295</td>
<td>0.451 x 10^1</td>
</tr>
<tr>
<td>8</td>
<td>-0.1786</td>
<td>1.1303</td>
<td>0.0879</td>
<td>-0.7091</td>
<td>-0.7994</td>
<td>2.5819</td>
<td>2.2876</td>
<td>-9.1173</td>
<td>0.809 x 10^1</td>
</tr>
<tr>
<td>9</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.419 x 10^{-32}</td>
<td>0.211 x 10^{-32}</td>
</tr>
</tbody>
</table>
7. **Quadratic Examples: Discussion**

For quadratic examples, all algorithms produce the solutions in the same number of iterations and the same number of gradient evaluations as predicted by the theory of Ref. 1, that is,

\[
N \leq n + 1, \quad K \leq n + 2 \tag{47}
\]

where \( n \) is the number of variables, \( N \) the number of iterations for convergence, and \( K \) the number of gradient evaluations needed for convergence. Moreover, the B-matrix for all the algorithms tends to the inverse of the hessian \( C \), that is,

\[
B_n = C^{-1} \tag{48}
\]

A word of caution is due at this moment. Two occasions may arise in which additional iterations and gradient evaluations are required in order to achieve the specified overall stopping condition (33): (a) the prescribed quantity \( \epsilon_1 \) in Eq. (2) is too large so that premature switching from one type of displacements to the other type of displacements occurs and (b) the loss of accuracy in computation due to round-off errors is too large so that the specified overall stopping condition (33) is not satisfied at the proper iteration. Whenever this happens, the switching and restarting conditions provided for the algorithms present very efficient steps to be followed so that rapid refinement of solutions can be expected for every further iteration.
8. Nonquadratic Examples and Numerical Results

In this section, five nonquadratic examples are given and the corresponding numerical results are presented. For simplicity, all the symbols used here are scalar quantities. For the initial stepsize factor, Schemes A-D are employed.

Example 8.1. Minimize the function (Ref. 4)

\[ f = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 \]  \hspace{1cm} (49)

The solution is \( f = 0 \) at the point defined by

\[ x_1 = 1 , \ x_2 = 1 \]  \hspace{1cm} (50)

The nominal point used for starting an algorithm is

\[ x_1 = -1.2 , \ x_2 = 1.0 \]  \hspace{1cm} (51)

The number of iteration for convergence \( N \), the total number of gradient evaluations for convergence \( K \), and the total number of bisections \( S \) are given in Table 7. The total computational effort in evaluating the function and its gradient is signified by the total number of equivalent function evaluations \( M \). Assuming that a gradient evaluation is \( n \) times as involved as a function evaluation and noting that a function evaluation accompanies a gradient evaluation or a bisection, the number \( M \) is given by

\[ M = (n + 1)K + S \]  \hspace{1cm} (52)
The number M is also given in the table. It is noted that the number of iterations for convergence N is related to the total number of gradient evaluations K by the equation

\[ N = K - 1 \]  \hspace{1cm} (53)

**Example 8.2.** Minimize the function (Ref. 5)

\[
f = 100(x_1^2 - x_2^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 \\
+ 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1) \]  \hspace{1cm} (54)

This solution is \( f = 0 \) at the point

\[ x_1 = 1 \quad , \quad x_2 = 1 \quad , \quad x_3 = 1 \quad , \quad x_4 = 1 \]  \hspace{1cm} (55)

This function has a nonminimal stationary value \( f = 7.787 \) at the point defined by

\[ x_1 = -0.9679 \quad , \quad x_2 = 0.9471 \quad , \quad x_3 = -0.9695 \quad , \quad x_4 = 0.9512 \]  \hspace{1cm} (56)

The nominal point used for starting an algorithm is

\[ x_1 = -3 \quad , \quad x_2 = -1 \quad , \quad x_3 = -3 \quad , \quad x_4 = -1 \]  \hspace{1cm} (57)

The number of iterations for convergence N, the total number of gradient evaluations for convergence K, the total number of bisections S, and the total number of equivalent function evaluations M are given in Table 8.

**Example 8.3.** Minimize the function (Ref. 6)

\[
f = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \]  \hspace{1cm} (58)
The solution is \( f = 0 \) at the point defined by
\[
x_1 = 0 \ , \ x_2 = 0 \ , \ x_3 = 0 \ , \ x_4 = 0
\]

(59)

At this minimal point, the hessian of the function is singular. The nominal point used for starting an algorithm is
\[
x_1 = 10 \ , \ x_2 = 10 \ , \ x_3 = 10 \ , \ x_4 = -10
\]

(60)

The number of iterations for convergence \( N \), the total number of gradient evaluations for convergence \( K \), the total number of bisections \( S \), and the total number of equivalent function evaluations \( M \) are given in Table 9.

Example 8.4. Minimize the function (Ref. 7)
\[
f = (x_1 + 10x_2)^4 + 5(x_3 - x_4)^4 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4
\]

(61)

The solution is \( f = 0 \) at the point defined by
\[
x_1 = 0 \ , \ x_2 = 0 \ , \ x_3 = 0 \ , \ x_4 = 0
\]

(62)

At this minimal point, the hessian of the function is zero. The nominal point used for starting an algorithm is
\[
x_1 = 2 \ , \ x_2 = 2 \ , \ x_3 = -2 \ , \ x_4 = -2
\]

(63)

The number of iterations for convergence \( N \), the total number of gradient evaluations for convergence \( K \), the total number of bisection \( S \), and the total number of equivalent function evaluations \( M \) are given in Table 10.
Example 8.5. Minimize the function (Ref. 8)

\[ f = (e^{x_1} - x_2)^4 + 100(x_2 - x_3)^6 + \tan^4(x_3 - x_4) + x_1^8 + (x_4 - 1)^2 \]  

The solution is \( f = 0 \) at the point defined by

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

At this minimal point, the hessian of the function is singular. The nominal point used for starting an algorithm is

\[ x_1 = 1, \quad x_2 = 2, \quad x_3 = 2, \quad x_4 = 2 \]  

The number of iterations for convergence \( N \), the total number of gradient evaluations for convergence \( K \), the total number of bisections \( S \), and the total number of equivalent function evaluations \( M \) are given in Table 11.
Table 7. Example 8.1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>N</td>
</tr>
<tr>
<td>I</td>
<td>41</td>
</tr>
<tr>
<td>II</td>
<td>41</td>
</tr>
<tr>
<td>III</td>
<td>39</td>
</tr>
<tr>
<td>IV</td>
<td>36</td>
</tr>
</tbody>
</table>

Table 8. Example 8.2.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>N</td>
</tr>
<tr>
<td>I</td>
<td>160</td>
</tr>
<tr>
<td>II</td>
<td>140</td>
</tr>
<tr>
<td>III</td>
<td>127</td>
</tr>
<tr>
<td>IV</td>
<td>84</td>
</tr>
</tbody>
</table>
Table 9. Example 8.3.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>N</td>
<td>K</td>
</tr>
<tr>
<td>I</td>
<td>78</td>
</tr>
<tr>
<td>II</td>
<td>75</td>
</tr>
<tr>
<td>III</td>
<td>77</td>
</tr>
<tr>
<td>IV</td>
<td>43</td>
</tr>
</tbody>
</table>

Table 10. Example 8.4.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>N</td>
<td>K</td>
</tr>
<tr>
<td>I</td>
<td>48</td>
</tr>
<tr>
<td>II</td>
<td>45</td>
</tr>
<tr>
<td>III</td>
<td>52</td>
</tr>
<tr>
<td>IV</td>
<td>60</td>
</tr>
</tbody>
</table>
Table 11. Example 8.5

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>N K S M</td>
</tr>
<tr>
<td>I</td>
<td>76 77 2 387 76 77 2 387 35 36 33 213 29 30 25 175</td>
</tr>
<tr>
<td>II</td>
<td>88 89 2 447 88 89 2 447 29 30 22 172 30 31 30 185</td>
</tr>
<tr>
<td>III</td>
<td>60 61 4 309 60 61 4 309 33 34 74 244 29 30 26 176</td>
</tr>
<tr>
<td>IV</td>
<td>38 39 3 198 38 39 3 198 29 30 18 168 40 41 28 233</td>
</tr>
</tbody>
</table>
9. Nonquadratic Examples: Discussion

The five nonquadratic examples presented in Section 8 are of different natures. Examples 8.1 and 8.2 are characterized by the complicated surface far away from the minimal points while the hessian at the minimal point is positive-definite. Examples 8.3-8.5 are characterized by a singular or zero hessian at the minimal point while the surface far away from the minimal point is rather well behaved. It is seen from the numerical results that, regardless of the wide ranges in the choice of the stepsize factor, all algorithms exhibit satisfactory convergent property. Further comments are given in the followings.

(a) Schemes A-D for the choice of initial stepsize factor can be classified into two groups: Schemes A and B belonging to small-step processes and Schemes C and D belonging to large-step processes. The numerical results show that the small-step processes A and B lead to essentially identical results, while the large-step processes C and D display little likeness in their performances.

(b) In terms of the number of iterations for convergence N, the large-step processes C and D, in general, are superior to the small-step processes A and B. This difference is mainly due to the fact that the upper limit of the stepsize factor employed in Schemes A and B, in many occasions, is too small to yield effective reduction of the function values.

(c) In terms of the number of bisections S, the small-step processes A and B are superior to the large-step processes C and D. This difference is
due to the fact that the upper limit of the stepsize factor employed in Schemes C and D is too large such that a large number of bisections is needed in order to achieve the descent property.

(d) In terms of the number of equivalent function evaluations $M$, the performances of the small-step processes and the large-step processes do not indicate a decisive difference. This is owing to the contradictory effects of the stepsize factor on the number of iterations for convergence and the number of bisections needed.

(e) From comments (b)-(d), it is believed that large-step processes can be superior to the small-step processes in total performance by employing an improved process for the reduction of the stepsize factor instead of the simple bisection process.

(f) Comparing the performances of the four different algorithms, Algorithm IV, in general, is superior to Algorithms I-III. Moreover, Algorithm IV is less sensitive to the stepsize factor than Algorithms I-III.

(g) It is worthwhile noting that, when Schemes B and D are used, Algorithm IV is reduced to a method using only one matrix and one direction, namely, $B$ and $q$. The simplification is made possible because the sole function of the matrix $A$ and the direction $p$ is to determine the restarting point. Since restarts are not needed in Schemes B and D, the calculation of the matrix $A$ and the direction $p$ can be bypassed completely.
References


