Further Investigation of Multi-Step Quasi-Newton Methods

J.A. Ford¹ and I.A.R. Moghrabi²

The derivation and construction of multi-step quasi-Newton methods for optimization (by means of interpolating polynomials) is reviewed. We show how the numerical performance of one such method may be enhanced by use of a simple "safeguarding" mechanism designed to control the influence of "older" data and we assess the effect of this mechanism on other multi-step methods. Further multi-step algorithms, derived from conjugacy or orthogonality conditions applied to successive updating directions in the variable space, are developed and tested. On the basis of the numerical evidence, some conclusions concerning the usefulness of the interpolatory approach for constructing multi-step methods are drawn.

INTRODUCTION

In previous work [1,2], the authors introduced the concept of multi-step quasi-Newton algorithms for optimization. In these methods, the Secant (or quasi-Newton) Equation (which forms the basis of most standard quasi-Newton algorithms) is replaced by a condition which is similar in its general form, but which is derived from two or more past steps. The Hessian approximation is updated to satisfy this new condition.

Our notation is as follows: the objective function is \( f(x) \), where \( x \in \mathbb{R}^n \). \( \{x_i\} \) are the successive iterates generated by the method under consideration. The gradient and Hessian of \( f \) are denoted, respectively, by \( g \) and \( G \), while the matrix \( B_i \) is an approximation to \( G(x_i) \). \( X \) is a differentiable path \( \{x(\tau)\} \) in \( \mathbb{R}^n \), where

\[ \tau \in \mathbb{R} \text{ and } x(\tau) \text{ is an interpolating polynomial of degree } m \text{ satisfying:} \]

\[ x(\tau_k) = x_{i-m+k+1}, \quad \text{for } k = 0, 1, \ldots, m, \quad (1) \]

for given values \( \{\tau_k\}_{k=0}^m \). \( L_j(\tau) \) is the \( j^{th} \) Lagrange polynomial of degree \( m \) associated with the abscissae \( \{\tau_k\}_{k=0}^m \), so that \( L_j(\tau_j) = 1 \) and \( L_j(\tau_i) = 0 \) for \( i \neq j \). Finally:

\[ s_i \triangleq x_{i+1} - x_i, \quad (2) \]

and

\[ y_i \triangleq g(x_{i+1}) - g(x_i). \quad (3) \]

Since \( G(x_{i+1}) = G(x(\tau_m)) \) from Equation 1, we obtain (by means of the Chain Rule):

\[ G(x_{i+1}) x'(x(\tau_m)) = g'(x(\tau_m)), \quad (4) \]

1. Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, Essex, UK, CO4 3SQ.
2. Now at Natural Science Division, Beirut University College, P.O. Box 13-5053, Beirut, Lebanon.

where derivatives are taken with respect to $\tau$. Because (in quasi-Newton methods) we do not require $G$ to be available, we construct the desired Hessian approximations $\{B_j\}$ by stipulating that some approximate version of Equation 4 be satisfied:

$$B_{i+1}r_i = w_i.$$  

(5)

Here, $r_i$ is $x'(\tau_m)$ and may be computed explicitly, since we have defined the form of the curve $X$ (by specifying the degree of $x(\tau)$ and that the interpolatory conditions in Equation 1 are to be satisfied). Thus:

$$r_i = \sum_{k=0}^{m-1} \sum_{j=0}^{m-1} \sum_{k=m-j}^{m} s_{i-j} \sum_{k=m-j}^{m} L_k^m(\tau_m).$$  

(6)

(7)

On the other hand, the term $g'(x(\tau_m))$ occurring on the right-hand side of Equation 4 cannot (in general) be computed exactly without access to derivatives of the components of $g$ (that is, to $G$), and these have been assumed to be unavailable. Therefore, we estimate $g'(x(\tau_m))$ by means of a polynomial scheme based on the values $\{\tau_k\}_{k=0}^m$ and the available gradient evaluations $\{g(x_{i-m+k+1})\}_{k=0}^m$. Let $\hat{g}(\tau)$ be the polynomial vector form of degree $m$ which interpolates these gradient evaluations. Then we obtain (compare Equations 6 and 7):

$$g'(x(\tau_m)) \approx \hat{g}'(\tau_m)$$

$$= \sum_{k=0}^{m} L_k^m(\tau_m)g(x_{i-m+k+1})$$

$$= \sum_{j=0}^{m-1} \sum_{k=m-j}^{m} L_k^m(\tau_m).$$  

(8)

(9)

The derivation of Equation 5 from Equation 4 is now evident.

In the standard quasi-Newton approach for constructing the approximation $B_{i+1}$, $m$ is taken to be one, so that $r_i$ and $w_i$ are given by:

$$r_i = (\tau_i - \tau_0)^{-1}s_i; \quad w_i = (\tau_i - \tau_0)^{-1}y_i.$$  

Hence, in this case, Equation 5 turns out to be just a scaled version of the Secant Equation [3]:

$$B_{i+1}s_i = y_i.$$  

(11)

For values of $m$ greater than unity, however, $r_i$ depends not only upon $s_i$, but also upon $s_{i-1}, s_{i-2}, \ldots, s_{i-m+1}$ (compare Equation 7) and similar comments hold for $w_i$ (compare Equations 9 and 10). In addition, we remark that Equation 11, which specifies the property which $B_{i+1}$ must possess in the case when $m = 1$, is independent of the values assigned to $\{\tau_k\}_{k=0}^m$, whereas (when $m$ is greater than one) the corresponding relation (Equation 5) does not have this property. This suggests that, in such cases, the values $\{\tau_k\}_{k=0}^m$ may have to be chosen with some care, if we are to succeed in constructing effective new methods.

The obvious and most straightforward manner of obtaining a matrix $B_{i+1}$ which satisfies Equation 5 is to use a standard quasi-Newton updating formula involving $s_i$ and $y_i$ and satisfying Equation 11, and then simply replace $s_i$ and $y_i$ with $r_i$ and $w_i$, respectively. For example, the updating formula known as BFGS [4-7]:

$$B_{i+1} = B_i - \frac{B_is_is_i^T}{s_i^TB_is_i} + \frac{y_iy_i^T}{s_i^Ty_i},$$  

(12)

becomes:

$$B_{i+1} = B_i - \frac{B_ir_i^TB_i}{r_i^TB_ir_i} + \frac{w_iw_i^T}{r_i^Tw_i}.$$  

(13)

Other updating formulæ may clearly be adapted, in a similar manner, for use with these multi-step methods.

**SPECIFYING THE PARAMETERS**

In their numerical experiments, Ford and Moghrabi [2] found that methods for which
Ford and Moghrabi on Multi-step Quasi-Newton Methods

\( m = 2 \) generally out-performed (at least, for problems of dimension up to 80) methods for which \( m = 3 \). We shall, therefore, confine our attention to the case \( m = 2 \) for the rest of this paper. Ford and Moghrabi [1,2] further determined, by experiment, that the choice of the parameters \( \{ \tau_k \}_{k=0}^2 \) plays a crucial role in the numerical behavior of the resulting algorithms. Whereas, for example, the straightforward selection \( \tau_0 = -1, \tau_1 = 0, \tau_2 = +1 \) produces a modest improvement over the standard one-step BFGS formula, choices which determine \( \{ \tau_k \}_{k=0}^2 \) by reference to distances between the iterates \( x_{i-1}, x_i \) and \( x_{i+1} \) were found, in general, to yield much more substantial improvements (in this context, “distances” are measured with respect to the usual Euclidean norm, or by reference to metrics defined by the current Hessian approximations \( B_i \) or \( B_{i+1} \)). This approach produced six algorithms (called A1, A2, A3, F1, F2, F3), based on the following definitions of the intervals \( (\tau_2 - \tau_1) \) and \( (\tau_2 - \tau_0) \) or \( (\tau_1 - \tau_0) \), as convenient.

\[
F1 \quad \begin{align*}
(\tau_2 - \tau_1) &= \| s_i \|_2, \quad (17a) \\
(\tau_2 - \tau_0) &= \| s_i + s_{i-1} \|_2. \quad (17b)
\end{align*}
\]

\[
F2 \quad \begin{align*}
(\tau_2 - \tau_1) &= [-t_i s_i^T g(x_i)]^{1/2}, \quad (18a) \\
(\tau_2 - \tau_0) &= [-t_i s_i^T g(x_i) + 2s_i^T y_{i-1} + s_{i-1}^T y_{i-1}]^{1/2}. \quad (18b)
\end{align*}
\]

\[
F3 \quad \begin{align*}
(\tau_2 - \tau_1) &= [s_i^T y_i]^{1/2}, \quad (19a) \\
(\tau_2 - \tau_0) &= [s_i^T y_i + 2s_{i-1}^T y_i + s_{i-1}^T y_{i-1}]^{1/2}. \quad (19b)
\end{align*}
\]

**SAFEGUARDING THE NEW METHODS**

It is easy to show, using Equations 7, 9 and 10, that (when \( m = 2 \)) the expressions for \( r_i \) and \( w_i \) may be re-written in the form:

\[
\begin{align*}
\tau_i &= \mathcal{L}_n^m(\tau_2)[s_i - \{\delta^2/(2\delta + 1)\} s_{i-1}], \\
w_i &= \mathcal{L}_n^m(\tau_2)[y_i - \{\delta^2/(2\delta + 1)\} y_{i-1}],
\end{align*}
\]

where:

\[
\delta = (\tau_2 - \tau_1)/(\tau_1 - \tau_0). \quad (21)
\]

It is evident that the size of the term \( \delta^2/(2\delta + 1) \) is critical in determining the composition of the vectors \( r_i \) and \( w_i \). In particular, if \( \delta \) is large, we have (concentrating upon \( r_i \)) the approximation:

\[
r_i \approx s_i - (\delta/2)s_{i-1},
\]

where we have omitted the scaling factor \( \mathcal{L}_n^m(\tau_2) \), for simplicity. Hence, it would appear
to be worthwhile to place a constraint on the size of \( \delta \) in order to limit the relative contribution of the vector \( s_{i-1} \) (representing "older" data) to \( r_i \) (and of \( y_{i-1} \) to \( w_i \), of course). We are thus led to introduce a parameter \( \delta_{\text{max}} \) which restricts the permitted values of \( \delta \):

\[
\text{if } |\delta| > \delta_{\text{max}} \text{ then } \delta := \text{sign}(\delta) \cdot \delta_{\text{max}}.
\]

Note that \( \delta \) may be negative for the methods F1 to F3, although it cannot be so for A1 to A3.

To illustrate the effect of the parameter \( \delta_{\text{max}} \) on multi-step methods, we refer to the results presented in Table 1, which were derived from experiments on the third set of test problems (that is, those problems with dimensions between 46 and 80) described in [1]. This set was chosen for the experiments because the advantages of the multi-step methods are most clearly seen with higher-dimensional problems, as was shown empirically by Ford and Moghirabi. The algorithm from which these results were derived was F2 (see Equations 18a and 18b above). For comparison, the performance of the standard BFGS method on the same test set is also shown. The basic structure of all the algorithms is described in [1,2]. All execution times are given in seconds. Table 1 shows that the numerical performance of the algorithm F2 is influenced to a marked degree by the value of the parameter \( \delta_{\text{max}} \). While the precise value of \( \delta_{\text{max}} \) is not critical (a value in the interval [3.0, 4.0] appears to be indicated), it is clearly beneficial to include such a parameter in the definition of the method.

For completeness, we present in Table 2 an estimate of a suitable value of \( \delta_{\text{max}} \) for each of the methods defined by Equations 14 to 19. We stress, however, that these are not claimed to be optimal values of \( \delta_{\text{max}} \) (or even that such optimal values necessarily exist); they are merely the values which have yielded the best performances in our experiments. We also draw attention to the fact that the other five methods exhibit much less variation in performance as \( \delta_{\text{max}} \) is varied; this is indicated by the entry under "\( \delta_{\text{max}} \)" for these methods. For example, the entry "10.5(7.0+)" for the method

F1 implies that, while the best performance was obtained using the value 10.5 for \( \delta_{\text{max}} \), there was little, if any, significant variation in performance observed for values in the range 7.0 and above. The superiority of the method F2 is evident; in terms of execution time, it is nearly 30% better than the standard BFGS method and 6.6% better than the next best multi-step method.

### CONJUGATE DIRECTION METHODS

Equations 20a and 20b are evidently a particular case of a more general definition of \( r_i \) and \( w_i \) where we have again omitted the scaling factors \( L_2^2(\tau_2) \):

\[
\begin{align*}
 r_i &= s_i - \alpha_i s_{i-1} , \\
 w_i &= y_i - \alpha_i y_{i-1} .
\end{align*}
\]

(22a) (22b)

A question of some interest is whether there are other ways (apart from interpolatory polynomials) of defining the coefficient \( \alpha_i \). We shall investigate some possibilities in this section.

Given the well-known connections between (one-step) quasi-Newton minimization methods and conjugate direction methods, an attractive strategy is to choose \( \alpha_i \) such that (if possible) successive updating directions \( r_{i-1} \) and \( r_i \) are conjugate vectors. To accomplish this, we need to specify the matrix with respect to which conjugacy is to be achieved and, since we will be dealing, in general, with non-quadratic functions and therefore non-constant and unknown Hessians, the obvious candidates are \( B_i \) and \( B_{i+1} \). In each case, we can determine a unique value of \( \alpha_i \) which produces the required conjugacy property.

(i) Using \( B_i \):

\[
\begin{align*}
 0 &= r_i^T B_i r_{i-1} = r_i^T w_{i-1} ,
\end{align*}
\]

using Equation 5, with \( i \) replaced by \( i - 1 \):

\[
0 = (s_i - \alpha_i s_{i-1})^T w_i ,
\]

using Equation 22a, and thus we obtain, as the required value of \( \alpha_i \):

\[
\alpha_i^B = s_i^T w_{i-1} / s_{i-1}^T w_{i-1} .
\]

(23)
Table 1. The effect of the parameter $\delta_{\text{max}}$ on the algorithm F2.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\delta_{\text{max}}$</th>
<th>Execution Time</th>
<th>Function Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFGS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>1.0</td>
<td>719</td>
<td>18575</td>
</tr>
<tr>
<td>F2</td>
<td>2.0</td>
<td>586</td>
<td>15076</td>
</tr>
<tr>
<td>F2</td>
<td>3.0</td>
<td>531</td>
<td>13929</td>
</tr>
<tr>
<td>F2</td>
<td>3.1</td>
<td>518</td>
<td>13576</td>
</tr>
<tr>
<td>F2</td>
<td>3.2</td>
<td>513</td>
<td>13398</td>
</tr>
<tr>
<td>F2</td>
<td>3.3</td>
<td>517</td>
<td>13524</td>
</tr>
<tr>
<td>F2</td>
<td>3.4</td>
<td>516</td>
<td>13520</td>
</tr>
<tr>
<td>F2</td>
<td>3.5</td>
<td>515</td>
<td>13427</td>
</tr>
<tr>
<td>F2</td>
<td>3.6</td>
<td>514</td>
<td>13462</td>
</tr>
<tr>
<td>F2</td>
<td>3.7</td>
<td>513</td>
<td>13393</td>
</tr>
<tr>
<td>F2</td>
<td>3.8</td>
<td>512</td>
<td>13383</td>
</tr>
<tr>
<td>F2</td>
<td>3.9</td>
<td>512</td>
<td>13376</td>
</tr>
<tr>
<td>F2</td>
<td>4.0</td>
<td>514</td>
<td>13446</td>
</tr>
<tr>
<td>F2</td>
<td>4.1</td>
<td>515</td>
<td>13487</td>
</tr>
<tr>
<td>F2</td>
<td>4.2</td>
<td>518</td>
<td>13524</td>
</tr>
<tr>
<td>F2</td>
<td>4.3</td>
<td>520</td>
<td>13593</td>
</tr>
<tr>
<td>F2</td>
<td>4.4</td>
<td>523</td>
<td>13622</td>
</tr>
<tr>
<td>F2</td>
<td>4.5</td>
<td>522</td>
<td>13704</td>
</tr>
<tr>
<td>F2</td>
<td>5.0</td>
<td>534</td>
<td>13696</td>
</tr>
<tr>
<td>F2</td>
<td>8.0</td>
<td>550</td>
<td>13971</td>
</tr>
<tr>
<td>F2</td>
<td>10.0</td>
<td>558</td>
<td>14593</td>
</tr>
<tr>
<td>F2</td>
<td>15.0</td>
<td>564</td>
<td>14889</td>
</tr>
<tr>
<td>F2</td>
<td>40.0</td>
<td>594</td>
<td>15197</td>
</tr>
</tbody>
</table>

Table 2. Relative performance of six multi-step algorithms.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\delta_{\text{max}}$</th>
<th>Execution Time</th>
<th>Function Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFGS</td>
<td></td>
<td>719</td>
<td>18575</td>
</tr>
<tr>
<td>F1</td>
<td>10.5(7.0+)</td>
<td>555</td>
<td>14494</td>
</tr>
<tr>
<td>F2</td>
<td>3.8</td>
<td>512</td>
<td>13376</td>
</tr>
<tr>
<td>F3</td>
<td>30.0(9.0+)</td>
<td>553</td>
<td>14305</td>
</tr>
<tr>
<td>A1</td>
<td>12.0(8.5+)</td>
<td>561</td>
<td>14545</td>
</tr>
<tr>
<td>A2</td>
<td>14.0(5.5+)</td>
<td>548</td>
<td>15005</td>
</tr>
<tr>
<td>A3</td>
<td>1e10(4.5+)</td>
<td>574</td>
<td>14816</td>
</tr>
</tbody>
</table>
(ii) Using $B_{i+1}$:

$$0 = r_{i+1}^T B_{i+1} r_{i-1} = w_{i}^T r_{i-1}^T,$$

using Equation 5 again:

$$0 = (y_i - \alpha_i y_{i-1})^T r_i,$$

using Equation 22b, so that the desired value of $\alpha_i$ is given, this time, by:

$$\alpha_i^c = y_i^T r_{i-1} / y_{i-1}^T r_{i-1}.$$  \hspace{1cm} (24)

We note that, if the objective function is quadratic (with constant Hessian $A$, say), then Equations 23 and 24 yield the same value for $\alpha_i$, because $y_j = As_j$ and hence, $w_j = Ar_j$, for all $j$.

We now pose the question of whether such values of $\alpha_i$ could have been obtained from an interpolatory scheme of the type described above. In other words (see Equations 20a and 20b), we are asking whether there exists a (real) value of $\delta$ such that:

$$\alpha_i = \delta^2 / (2\delta + 1).$$

It is straightforward to show that such a value of $\delta$ does exist, provided that $\alpha_i \not\in (-1,0)$. The following two strategies, therefore, suggest themselves with regard to using the values of $\alpha_i$ derived in Equations 23 and 24, depending upon whether we wish to retain the connection with interpolatory schemes, or not.

**Strategy I**

Accept whatever value of $\alpha_i$ is generated by Equation 23 or 24.

**Strategy II**

Accept the value of $\alpha_i$ given by Equation 23 or 24 as appropriate, unless $\alpha_i \in (-1,0)$, in which case re-define $\alpha_i$ to be the nearer end-point of the interval [-1,0].

Combining these two strategies with the two possible values of $\alpha_i$ (Equations 23 and 24) gives four methods. A brief summary of the performance of these methods (on the same test set as before) is given in Table 3, where, as with Table 2, we have performed a series of experiments for each of the new methods in order to ascertain a suitable value for $\delta_{max}$. The notation “B/T” indicates method B, defined by Equation 23, combined with Strategy I, etc.

As a further alternative in the same vein, we consider choosing $\alpha_i$ so that successive updating directions are orthogonal (instead of conjugate). This leads to the following definition of $\alpha_i$:

$$\alpha_i^D = s_i^T r_{i-1} / s_{i-1}^T r_{i-1}.$$  \hspace{1cm} (25)

Again, this choice of $\alpha_i$ may be combined with either of Strategies I or II. A summary of results for these two algorithms is shown in Table 4.

**SUMMARY AND CONCLUSIONS**

The motivations for interpolatory multi-step quasi-Newton methods and their derivation have been presented. They may be viewed as natural extensions of the familiar one-step quasi-Newton approach in which there is additional flexibility to be exploited. This flexibility consists (i) in freedom to choose the number of steps $m$ which will be utilized, and (ii) in the ability to specify (with considerable latitude) the values of the abscissae $\{\tau_k\}_{k=0}^m$ which determine the precise shape of the interpolating curves $\{x(\tau)\}$ and $\{g(\tau)\}$.

Consideration of the nature of the terms used in the equation, which the updated Hessian approximation $B_{i+1}$ is constrained to satisfy (Equation 5), led to the introduction of the safeguarding parameter $\delta_{max}$. The results of numerical experiments were used to demonstrate the effect of this parameter in improving the performance of the new methods. In particular, the two-step fixed-point method $F2$ benefits considerably from a careful choice of this parameter and exhibits an improvement in performance of around 30% by comparison with the standard BFGS method.

Algorithms based on requiring successive updating directions $r_{i-1}$ and $r_i$ to be conjugate
Table 3. Four conjugate-direction multi-step algorithms.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\delta_{\text{max}}$</th>
<th>Execution Time</th>
<th>Function Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>B/I</td>
<td>2.0</td>
<td>561</td>
<td>17204</td>
</tr>
<tr>
<td>B/II</td>
<td>2.2</td>
<td>555</td>
<td>16259</td>
</tr>
<tr>
<td>C/I</td>
<td>2.25</td>
<td>571</td>
<td>17239</td>
</tr>
<tr>
<td>C/II</td>
<td>2.5</td>
<td>544</td>
<td>15752</td>
</tr>
<tr>
<td>BFGS</td>
<td></td>
<td>719</td>
<td>18575</td>
</tr>
<tr>
<td>F2</td>
<td>3.8</td>
<td>512</td>
<td>13376</td>
</tr>
</tbody>
</table>

Table 4. Two orthogonal-direction multi-step algorithms.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\delta_{\text{max}}$</th>
<th>Execution Time</th>
<th>Function Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>D/I</td>
<td>1.0</td>
<td>581</td>
<td>15395</td>
</tr>
<tr>
<td>D/II</td>
<td>1.3</td>
<td>569</td>
<td>15284</td>
</tr>
<tr>
<td>BFGS</td>
<td></td>
<td>719</td>
<td>18575</td>
</tr>
<tr>
<td>F2</td>
<td>3.8</td>
<td>512</td>
<td>13376</td>
</tr>
</tbody>
</table>

(or orthogonal) have also been developed. As has been demonstrated experimentally, these methods exhibit a distinct improvement over the BFGS method. However, when assessed against the earlier multi-step methods introduced in [2], their somewhat disappointing numerical performance (here, we draw attention particularly to the function evaluation counts for the various methods) suggests:

(i) That the interpolatory approach of the algorithms developed in [1] and [2] is an effective means of utilizing past data and should not be dispensed with lightly.

(ii) That, in particular, the choice of $\{\tau_k\}_{k=0}^2$ (and, therefore, the nature of $X = \{x(\tau)\}$) is a very important factor in determining numerical behavior. For the case $m = 2$, the values of $\{\tau_k\}_{k=0}^2$ defined by the metrics discussed in [2] (see Equations 14 to 19) give rise to algorithms which show a substantial improvement in performance when compared with the methods developed under Strategy II in the previous section, even though such algorithms may themselves be regarded (in some sense) as interpolatory.

ACKNOWLEDGEMENTS

One of us (I.A.M.) acknowledges, with gratitude, the assistance of the Hariri Foundation during this research. The other author (J.A.F.) is grateful for the provision of a grant by C.N.R. (Italy) which supported part of this research.

REFERENCES


3. Dennis, J.E. “On some methods based on Broyden’s secant approximation to the Hessian”, in Numerical Methods for Non-linear


