MATHEMATICS OF COMPUTATION, VOLUME 35, NUMBER 151
JULY 1980, PAGES 773–782

Updating Quasi-Newton Matrices With Limited Storage

By Jorge Nocedal

Abstract. We study how to use the BFGS quasi-Newton matrices to precondition minimization methods for problems where the storage is critical. We give an update formula which generates matrices using information from the last m iterations, where m is any number supplied by the user. The quasi-Newton matrix is updated at every iteration by dropping the oldest information and replacing it by the newest information. It is shown that the matrices generated have some desirable properties.

The resulting algorithms are tested numerically and compared with several well-known methods.

1. Introduction. For the problem of minimizing an unconstrained function f of n variables, quasi-Newton methods are widely employed [4]. They construct a sequence of matrices which in some way approximate the hessian of f (or its inverse). These matrices are symmetric; therefore, it is necessary to have $n(n+1)/2$ storage locations for each one. For large dimensional problems it will not be possible to retain the matrices in the high speed storage of a computer, and one has to resort to other kinds of algorithms. For example, one could use the methods (Toint [15], Shanno [12]) which preserve the sparsity structure of the hessian, or conjugate gradient methods (CG) which only have to store 3 or 4 vectors. Recently, some CG algorithms have been developed which use a variable amount of storage and which do not require knowledge about the sparsity structure of the problem [2], [7], [8]. A disadvantage of these methods is that after a certain number of iterations the quasi-Newton matrix is discarded, and the algorithm is restarted using an initial matrix (usually a diagonal matrix).

We describe an algorithm which uses a limited amount of storage and where the quasi-Newton matrix is updated continuously. At every step the oldest information contained in the matrix is discarded and replaced by new one. In this way we hope to have a more up to date model of our function. We will concentrate on the BFGS method since it is considered to be the most efficient. We believe that similar algorithms cannot be developed for the other members of the Broyden β-class [1].

Let f be the function to be minimized, g its gradient and h its hessian. We define

$$s_k = x_{k+1} - x_k \quad \text{and} \quad y_k = g_{k+1} - g_k.$$

Received May 29, 1979.
1980 *Mathematics Subject Classifications.* Primary 65K05; Secondary 90C30.

The BFGS update formula [4] is given by

$$\bar{H} = H + \frac{ss^T}{y^T s}\left[\frac{y^T H y}{y^T s} + 1\right] - \frac{1}{y^T s}\,[sy^T H + Hys^T], \tag{1}$$

and we will write it as

$$\bar{H} = H + U(s, y, H). \tag{2}$$

It is easy to verify that if H is positive definite and $y^T s > 0$, then $\bar{H}$ is positive definite. From now on we will assume that $y_k^T s_k > 0$ for all k. In gradient-related methods this can always be done, provided the line search is sufficiently accurate. In the usual implementation of quasi-Newton methods $\bar{H}$ overwrites H. This requires in general $n^2/2 + n/2$ storage locations. In our implementation we will keep the corrections U (defined in (2)) individually. From (1) we see that every new correction of H requires the storage of two vectors, namely s and y.

Given a positive definite and diagonal matrix H_0 one constructs

$$H_1 = H_0 + U(s_0, y_0, H_0)$$

$$H_2 = H_0 + U(s_0, y_0, H_0) + U(s_1, y_1, H_1)$$

$$\vdots$$

etc.,

where $\{s_k\}$ is generated by some minimization iteration. Let m be the maximum number of correction matrices U that can be stored. Since H_0 is diagonal, this means that the maximum number of n-vectors that we can use to define the quasi-Newton matrix is $2m + 1$. Once H_m is generated we have reached the storage limit,

$$H_m = H_0 + U(s_0, y_0, H_0) + \cdots + U(s_{m-1}, y_{m-1}, H_{m-1}).$$

We would like to drop the term $U(s_0, y_0, H_0)$ and replace it by one involving s_m and y_m. (We assume that H_m is used to produce s_m and y_m.) Note that the corrections $U(s_1, y_1, H_1), \ldots, U(s_{m-1}, y_{m-1}, H_{m-1})$ depend on $U(s_0, y_0, H_0)$. Therefore, if we discard this term, we would have to change all the other ones; and that is not desirable. We could avoid this problem by storing s, $y^T H$ and $s^T y$ at every iteration. We could then drop $U(s_0, y_0, H_0)$ and the other corrections would be unaffected. However, this approach leads us to other difficulties: losing the positive definiteness of the matrices and/or obtaining iterative methods without quadratic termination.

Now we will see that using a different description for the updating a strategy for discarding updates can easily be found.

2. A Special BFGS Update Formula. The BFGS formula (1) can also be written in product form

$$\bar{H} = (I - \rho sy^T)H(I - \rho ys^T) + \rho ss^T \equiv v^T H v + \rho ss^T, \tag{3}$$

where $\rho = 1/y^T s$.

We will consider dropping a correction to be equivalent to defining $v = I$ and $\rho ss^T = 0$. To illustrate the algorithm suppose that the number of corrections stored,

m, equals 2. We write

$$\rho_i = 1/y_i^T s_i \quad \text{and} \quad v_i = (I - \rho_i y_i s_i^T),$$

H_0 is a given positive definite matrix,

$$H_1 = v_0^T H_0 v_0 + \rho_0 s_0 s_0^T,$$

$$H_2 = v_1^T v_0^T H_0 v_0 v_1 + v_1^T \rho_0 s_0 s_0^T v_1 + \rho_1 s_1 s_1^T.$$

We now discard the old information, to leave

$$\bar{H}_2 = v_1^T H_0 v_1 + \rho_1 s_1 s_1^T$$

and update this matrix

$$H_3 = v_2^T v_1^T H_0 v_1 v_2 + v_2^T \rho_1 s_1 s_1^T v_2 + \rho_2 s_2 s_2^T.$$

Similarly,

$$H_4 = v_3^T v_2^T H_0 v_2 v_3 + v_3^T \rho_2 s_2 s_2^T v_3 + \rho_3 s_3 s_3^T$$
$$\vdots$$
etc.

In general, we have for $k + 1 \leqslant m$ the usual BFGS update

$$\begin{aligned} H_{k+1} = {} & v_k^T v_{k-1}^T \cdots v_0^T H_0 v_0 \cdots v_{k-1} v_k \\ & + v_k^T \cdots v_1^T \rho_0 s_0 s_0^T v_1 \cdots v_k \\ & \vdots \\ & + v_k^T v_{k-1}^T \rho_{k-2} s_{k-2} s_{k-2}^T v_{k-1} v_k \\ & + v_k^T \rho_{k-1} s_{k-1} s_{k-1}^T v_k \\ & + \rho_k s_k s_k^T. \end{aligned} \tag{4}$$

For $k + 1 > m$ we have the special update

$$\begin{aligned} H_{k+1} = {} & v_k^T v_{k-1}^T \cdots v_{k-m+1}^T H_0 v_{k-m+1} \cdots v_{k-1} v_k \\ & + v_k^T \cdots v_{k-m+2}^T \rho_{k-m+1} s_{k-m+1} s_{k-m+1}^T v_{k-m+2} \cdots v_k \\ & \vdots \\ & + v_k^T \rho_{k-1} s_{k-1} s_{k-1}^T v_k \\ & + \rho_k s_k s_k^T. \end{aligned} \tag{5}$$

The matrices defined by (4) and (5) will be called *special BFGS matrices.*

Properties. (a) If H_0 is positive definite, it is easy to verify that the matrices defined by (4) and (5) are positive definite (provided $y_i^T s_i > 0$ for all i).

(b) Let f be a strictly convex quadratic function $f(x) = \frac{1}{2} x^T A x + b^T x$, and suppose that the vectors s_k are conjugate, i.e.

$$s_i^T A s_j = 0, \qquad i \neq j.$$

Then the matrices satisfy the quasi-Newton equation in the past m-directions

$$H_k y_j = s_j, \quad j = k-1, \ldots, k-m \ (\text{for } k > m). \tag{6}$$

To see this note that

$$v_i y_i = 0, \quad i = k, k-1, \ldots, k-m+1, \quad \text{and} \quad v_i y_j = y_j \quad \text{for } i > j. \tag{7}$$

Note that if $m = 1$ the memoryless BFGS update is obtained (see Shanno [11]).

(c) The BFGS update formula can be written in two forms ((2) and (3))

$$\bar{H} = H + U(s, y, H) \qquad \text{Sum-Form}, \tag{8}$$

$$\bar{H} = v^T H v + \rho s s^T \qquad \text{Product-Form}. \tag{9}$$

The algorithm just described used the Product-Form description. We shall now express it using the Sum-Form. Again, let m be the number of corrections stored. The first m updates will be the usual BFGS updates

$$H_{i+1} = H_i + U(s_i, y_i, H_i), \quad i = 0, 1, \ldots, m-1.$$

From (5) we have that H_{m+1} can be computed as follows:

$$\begin{aligned} &\text{Let } \hat{H}_1 = H_0. \\ &\text{For } j = 1, 2, \ldots, m, \quad \hat{H}_{j+1} = \hat{H}_j + U(s_j, y_j, \hat{H}_j). \\ &H_{m+1} = \hat{H}_{m+1}. \end{aligned}$$

In general, for any $k > m$ let $s = k + 1 - m$, then H_{k+1} is given by

$$\begin{cases} \hat{H}_s = H_0. \\ \text{For } j = s, s+1, \ldots, k, \quad \hat{H}_{j+1} = \hat{H}_j + U(s_j, y_j, \hat{H}_j). \\ H_{k+1} = \hat{H}_{k+1}. \end{cases} \tag{10}$$

Using the Sum-Form description it is necessary to recompute all the corrections at each step.

3. Gradient Related Methods. The special BFGS matrices given by (4) and (5) could be used in quasi-Newton iterations

$$d_k = -H_k g_k, \quad x_{k+1} = x_k + \alpha_k d_k, \tag{11}$$

or in preconditioned conjugate gradient iterations (PCG)

$$\begin{aligned} &\text{(a)} \quad d_0 = -H_0 g_0, \\ &\qquad\quad d_k = -H_k g_k + \beta_k d_{k-1}, \\ &\qquad\quad \text{where } \beta_k = y_{k-1}^T g_k / y_{k-1}^T d_{k-1}, \\ &\text{(b)} \quad x_{k+1} = x_k + \alpha_k d_k. \end{aligned} \tag{12}$$

The PCG method can be obtained by doing the transformation $z_k = L_k^{-1}x_k$, and then applying the conjugate gradient method in the new variables. Here L_k is the Cholesky factor of H_k: $H_k = L_k L_k^T$; see [8]. Algorithm (12) is usually restarted every n iterations by setting $\beta_k = 0$.

We now show that using the special BFGS updates in the above two iterations we obtain algorithms with quadratic termination.

Let f be a strictly convex quadratic function and suppose that exact line searches are performed. The PCG method will be implemented in the manner proposed by Nazareth [7]. Given H_0 we let

$$
\begin{aligned}
&d_0 = -H_0 g_0, \\
&x_1 = x_0 + \alpha_0 d_0, \qquad \text{(a)} \\
&H_1 = F(s_0, y_0, H_0), \\
&\text{for } i = 1, 2, \ldots, \\
&d_i = -H_{i-1} g_i + \beta_i d_{i-1}, \\
&x_{i+1} = x_i + \alpha_i d_i, \qquad \text{(b)} \\
&H_{i+1} = F(s_i, y_i, H_i), \\
&\text{where } \beta_i = y_{i-1}^T H_{i-1} g_i / y_{i-1}^T d_i;
\end{aligned}
\tag{13}
$$

and $F(s, y, H)$ denotes the special BFGS update given by (4)–(5). We will call (13) *the SCG algorithm*. In practice it should be restarted every n iterations. Note that we are not using the most recently defined matrix as preconditioner, but the one before it.

During the first m iterations the SCG will generate the usual BFGS matrices. It is not difficult to show (see [8]) that the first $m + 2$ directions are the same as those obtained by applying the PCG with fixed preconditioner H_0, i.e.,

$$
d_i(g_i, H_{i-1}, d_{i-1}) = d_i(g_i, H_0, d_{i-1}), \qquad i = 0, 1, \ldots, m + 1, \tag{14}
$$

where $d(g, H, d-)$ is defined in (13).

The conjugacy and orthogonality conditions hold (see [8])

$$
d_i^T y_j = 0, \quad i \neq j,\ i, j = 0, 1, \ldots, m + 1, \tag{15}
$$

$$
\begin{aligned}
&g_i^T H_0 g_j = 0, \quad g_i^T d_j = 0 \quad \text{for } i \neq j, \\
&\qquad\qquad\qquad\qquad i = 0, 1, \ldots, m + 2, \\
&\qquad\qquad\qquad\qquad j = 0, 1, \ldots, m + 1.
\end{aligned}
\tag{16}
$$

We will now see that (14) and, hence, (15) and (16) also hold for the rest of the iterations. From (5) we have

$$
\begin{aligned}
H_{m+1} = {} & v_m^T \cdots v_1^T H_0 v_1 \cdots v_m \\
& + v_m^T \cdots v_2^T \rho_1 s_1 s_1^T v_2 \cdots v_m \\
& \vdots \\
& + v_m^T \rho_{m-1} s_{m-1} s_{m-1}^T v_m \\
& + \rho_m s_m s_m^T.
\end{aligned}
$$

We compute d_{m+2}:

$$d_{m+2} = -H_{m+1} g_{m+2} + \beta_{m+2} d_{m+1},$$

where $\beta_{m+2} = y_{m+1}^T H_{m+1} g_{m+2} / y_{m+1}^T d_{m+1}$.

From (15) and (16) we have

$$H_{m+1} g_{m+2} = H_0 g_{m+2}, \qquad y_{m+1}^T H_{m+1} g_{m+1} = y_{m+1}^T H_0 g_{m+1}.$$

Therefore,

$$d_{m+2}(g_{m+2}, H_{m+1}, d_{m+1}) = d_{m+2}(g_{m+2}, H_0, d_{m+1});$$

and from this it now follows that

(i) $d_0, \ldots, d_{m+2}$ are conjugate,

(ii) $g_{m+3}^T H_0 g_j = 0$, $j = 0, \ldots, m+2$,

(iii) $g_{m+3}^T d_i = 0$, $i = 0, \ldots, m+2$.

We continue in the same fashion to prove that for $i > m$,

$$d_{i+1}(g_{i+1}, H_i, d_i) = d_{i+1}(g_{i+1}, H_0, d_i).$$

Therefore, for quadratic functions and exact line searches the SCG is the same as the PCG with fixed preconditioner H_0 and has quadratic termination.

Let us now consider the use of the special update in the iteration (11). H_0 is a given matrix.

$$
\begin{aligned}
& d_i = -H_i g_i, \\
& x_{i+1} = x_i + \alpha_i d_i, \\
& H_{i+1} = F(s_i, y_i, H_i),
\end{aligned}
\tag{17}
$$

where F is given by (4)–(5). This algorithm will be called the *SQN*. Using a similar argument as for the SCG, it is straightforward to show that for quadratic functions and exact line searches this algorithm is also identical to the PCG with fixed preconditioner. Hence, it has quadratic termination.

We would expect that the efficiency of the algorithms would increase as the number of corrections stored increases. Equation (6) seems to indicate so. The following argument also backs this reasoning. First we quote a result due to Fletcher [5].

THEOREM. *Let f be a strictly convex quadratic function, H_0 symmetric and positive definite. Let $\{s_k\}$ be generated by the BFGS method. Let λ_i^k, $i = 1, \ldots, n$, be the eigenvalues of $A^{1/2} H_k A^{1/2}$, where A is the hessian of f. Then*

$$\min\{\lambda_i^k, 1\} \leqslant \lambda_i^{k+1} \leqslant \max\{\lambda_i^k, 1\}. \tag{18}$$

(Note that this result does not require exact line searches.)

The theorem can be used to prove that for nonlinear functions (and under certain conditions on the line search) the BFGS quasi-Newton method generates matrices H_k with uniformly bounded condition numbers and is locally convergent (Stoer [14]).

Equation (10) tells us that any H_k obtained by means of the special update can also be obtained using the usual BFGS formula. Therefore, (18) also holds.

Let $k > m$ and define $\hat{\lambda}_i^j$, $i = 1, \ldots, n$, to be the eigenvalues of $A^{1/2}\hat{H}_j A^{1/2}$. Then for the special update

$$\begin{aligned} &\lambda_i^{k+1} \leqslant \max\{\hat{\lambda}_i^k, 1\} \leqslant \cdots \leqslant \max\{\lambda_i^s, 1\}, \\ &\lambda_i^{k+1} \geqslant \min\{\hat{\lambda}_i^k, 1\} \geqslant \cdots \geqslant \min\{\lambda_i^s, 1\}, \end{aligned} \tag{19}$$

where $s = k + 1 - m$.

We conclude that the condition number of the special matrices is bounded for quadratic functions. The larger m is, the greater the number of inequalities in (19) will be. Note that λ_i^s is an eigenvalue of $A^{1/2}H_0A^{1/2}$. In a similar way as it was done in [9] for the restarted PCG (there it is called the VSCG method) (19) can be used to show that for nonlinear functions SCG and SQN using asymptotically exact line searches generate matrices with uniformly bounded condition numbers and, therefore, are locally convergent.

4. A Recursive Formula to Compute $H \cdot g$. When using the special matrices (4)–(5) in quasi-Newton steps or in preconditioned conjugate gradient steps, the product $H \cdot g$ has to be computed. The following recursion performs this efficiently. It is essentially the same as the formula for the usual BFGS [6]. M is the number of corrections stored. ITER is the iteration number.

1) IF ITER $\leqslant M$ SET INCR = 0; BOUND = ITER ELSE SET INCR = ITER $- M$; BOUND = M

2) $q_{\text{BOUND}} = g_{\text{ITER}}$

3) FOR i = (BOUND–1), . . . , 0

$\quad j = i + \text{incr}$

$\quad \alpha_i = \rho_j s_j^T q_{i+1}$ (STORE α_i) (a)

$\quad q_i = q_{i+1} - \alpha_i y_j$ (b)

$r_0 = H_0 \cdot q_0$ (c)

FOR $i = 0, 1, \ldots,$ (BOUND–1)

$\quad j = i + \text{incr}$

$\quad \beta_j = \rho_j y_j^T r_i$ (d)

$\quad r_{i+1} = r_i + s_j(\alpha_i - \beta_i)$ (e)

This formula requires at most $4nM + 2M + n$ multiplications and $4nM + M$ additions.

5. **Numerical Results.** The limited storage BFGS formula was implemented with the quasi-Newton iteration (17) and with the preconditioned conjugate gradient iteration (13). The resulting methods are denoted by SQN and SCG, respectively. They are compared with the usual BFGS method, with Shanno's method [11] and with a standard CG method. The numbers given in Table 1 indicate the number of function evaluations.

TABLE I

	CG	SCG			SQN			BFGS	SHANNO
		MSTORE			MSTORE				
		2	4	8	3	4	8		
Fletcher n=3	75	59	53	51	47	55	44	32	49
Biggs n = 6	235	60	49	46	95	77	68	50	82
Powell n=4	165	82	76	68	122	69	83	59	150
Wood n = 4	292	146	181	155	74	67	56	45	112
Ext. Powell n = 8	168	115	93	79	116	103	83	70	150
n =16	170	113	99	92	94	92	76	66	121
n = 20	211	106	105	98	97	84	92	47	129
TRIG									
n = 10					364	271	204		
n = 15					310	271	209		
n = 20					425	413	307		

The BFGS and Shanno's method were run using the code of Shanno and Phua [13]. The SQN and SCG were tested for different values of MSTORE (the number of corrections stored). For SQN the initial matrix was scaled after the first iteration using

the formula

$$\hat{H}_0 = s_0^T y_0 / y_0^T H_0 y_0.$$

The SCG was restarted every n iterations or whenever a nondescent direction was found. This happened very rarely.

All methods employed the line search procedure given in [13]. It uses safeguarded cubic interpolation to find an α such that

$$f(x + \alpha d) < f(x) + .0001 \alpha d^T g(x)$$

and

$$|d^T g(x + \alpha d)/d^T g(x)| < .9.$$

For the SCG and SQN two values of the steplength were always tried before accepting the step.

The test problems are Fletcher's helix, Bigg's exponential, Powell's singular, Wood's, Extended Powell and the Trigonometric functions. They are all documented in [3], where the initial points are also given. The convergence criterion was $\|g\| < \epsilon$, where $\epsilon = 10^{-8}$ in all problems except for Powell's singular, where $\epsilon = 10^{-6}$. All runs were made using a Burroughs B6700 computer in double length arithmetic.

For some values of MSTORE the limited storage methods are actually using more storage than the BFGS. In practice, one would of course never do this. However, as we are concerned with the effect of storing more vectors, these runs are of interest. For the Trigonometric functions the different methods converged to different solutions; we only report the results for SQN.

Shanno's method has the same storage requirements as SQN with MSTORE = 2. It is based on Beale's method and uses Powell's restart criterion. It is considerably more efficient than the standard conjugate gradient, as Table I shows. The SQN with MSTORE = 2 does not perform well, as was noted earlier in [11]; and we do not report its results here. SQN with MSTORE = 3 is somewhat faster than Shanno's method and speeds up as MSTORE increases. The SCG performs well for small values of MSTORE and improves also as MSTORE increases. We note that there are few instances where the performance does not improve by increasing MSTORE.

6. Conclusion. A formula for updating quasi-Newton matrices based on the BFGS and which uses a variable amount of storage is presented. It is shown that it produces positive definite matrices and when used in two classes of minimization algorithms it preserves the quadratic termination property. Furthermore, the quasi-Newton equation is satisfied in the past m directions, where m is the number of updates stored. Numerical experiments indicate that the resulting algorithms are very efficient and that their performance improves consistently as the storage used increases.

Acknowledgements. The author is grateful to Professor J. Dennis for guiding comments and to L. Nazareth for his continuing advice and helpful suggestions.

IIMAS
Universidad Nacional Autónoma de México
México 20, D. F., México

1. C. G. BROYDEN, "The convergence of a class of double-rank minimization algorithms," *J. Inst. Math. Appl.*, v. 6, 1970, pp. 76–90.

2. A. G. BUCKLEY, "A combined conjugate gradient quasi-Newton minimization algorithm," *Math. Programming*, v. 15, 1978, pp. 200–210.

3. W. C. DAVIDON & L. NAZARETH, *DRVOCR–A FORTRAN Implementation of Davidon's Optimally Conditioned Method*, TM-306, Argonne National Lab., Argonne, Ill., 1977.

4. J. E. DENNIS & J. J. MORE, "Quasi-Newton methods, motivation and theory," *SIAM Rev.*, v. 19, 1977, pp. 46–89.

5. R. FLETCHER, "A new approach to variable metric algorithms," *Comput. J.*, v. 13, 1970, pp. 317–322.

6. H. MATTHIES & G. STRANG, "The solution of nonlinear finite element equations," *Internat. J. Numer. Methods Engrg.*, v. 14, 1979, pp. 1613–1626.

7. L. NAZARETH, *A Relationship Between the BFGS and Conjugate Gradient Algorithms*, ANL-AMD Tech. Memo 282 (rev.), Argonne National Lab., Argonne, Ill., 1977.

8. L. NAZARETH & J. NOCEDAL, *A Study of Conjugate Gradient Methods*, Tech. Rep. SOL 78–29, Dept. of Operations Research, Stanford University, Stanford, Calif., 1979.

9. L. NAZARETH & J. NOCEDAL, "Convergence analysis of optimization methods that use variable storage," Manuscript, 1978.

10. A. PERRY, *A Modified Conjugate Gradient Algorithm*, Discussion paper No. 229, Center for Mathematical Studies in Economics and Management Science, Northwestern University, Evanston, Ill., 1976.

11. D. SHANNO, *Conjugate Gradient Methods With Inexact Line Searches*, MIS Tech. Report 22, University of Arizona, Tucson, Ariz., 1977.

12. D. SHANNO, *On Variable-Metric Methods for Sparse Hessians*, MIS Tech. Rep. 27, University of Arizona, Tucson, Ariz., 1978.

13. D. SHANNO & K. PHUA, *A Variable Method Subroutine for Unconstrained Nonlinear Minimization*, MIS Tech. Rep. No. 28, University of Arizona, Tucson, Ariz., 1978.

14. J. STOER, "On the convergence rate of imperfect minimization algorithms in Broyden's β-class," *Math. Programming*, v. 9, 1975, pp. 313–335.

15. P. TOINT, "On sparse and symmetric updating subject to a linear equation," *Math. Comp.*, v. 31, 1977, pp. 954–961.