Effect of limited precision on the BFGS quasi-Newton algorithm

D. Byatt I. D. Coope C. J. Price*

(Received 8 August 2003; revised 22 December 2003)

Abstract

Some claim that updating approximate Hessian information via the BFGS formula with a Cholesky factorisation offers greater numerical stability than the more straightforward approach of performing the update directly. Others claim that no such advantage exists and that any such improvement is probably due to early implementations of the DFP formula in conjunction with low accuracy line searches. We find no discernible advantage in choosing factorised implementations (over non-factorised implementations) of BFGS methods when approximate Hessian information is available to full machine precision. However, the type of implementation may have significant effects when approximate Hessian information is only available to limited precision. Furthermore, a conjugate directions factorisation outperforms the other methods explored (including Cholesky factorisation).

*University of Canterbury, Christchurch, New Zealand.
mailto:d.byatt@math.canterbury.ac.nz, mailto:i.coope@math.canterbury.ac.nz, mailto:c.price@math.canterbury.ac.nz

See http://anziamj.austms.org.au/V45/CTAC2003/Byat for this article, © Austral. Mathematical Soc. 2004. Published May 15, 2004. ISSN 1446-8735

Contents

1 Introduction

Quasi-Newton algorithms solve the local optimisation problem $\min_{x\in\mathbb{R}^n} f(x)$ iteratively, where $f\colon \mathbb{R}^n \to \mathbb{R}$ and gradient information is available. The solution is attained when $\nabla f(x) = 0$, but in practice the usual requirement is that $\|\nabla f(x)\| \leqslant \tau$ for some (typically small) positive constant τ.

We adopt the convention of writing $f(x_k)$ as f_k and $\nabla f(x_k)$ as g_k. At iteration k of a quasi-Newton method a search direction p_k is found by solving the system of equations

$$B_k p_k = -g_k\,, \tag{1}$$

where B_k approximates, in some sense, the Hessian matrix $\nabla^2 f(x_k)$. A line search is then performed along $x_k + \alpha p_k\,,\ \alpha \in \mathbb{R}$ to find a new iterate $x_{k+1} = x_k + \alpha_k p_k$ for some α_k that satisfies the line search criteria. Information at this new point is used to generate a new approximate Hessian matrix B_{k+1}.

If B_k is positive definite then $p_k^{\mathsf{T}} g_k < 0$ so that p_k is a descent direction for f. This paper investigates the performance of four BFGS implementations on a selection of ill-conditioned test problems across a range of dimensions and line search criteria as the precision of second order information varies from 16 to two digits. The results presented in this paper support and extend those reported in [4].

2 BFGS formula

The BFGS update formula can be written as

$$B_{k+1} = \left[B + \frac{yy^{\mathsf{T}}}{s^{\mathsf{T}}y} - \frac{Bss^{\mathsf{T}}B}{s^{\mathsf{T}}Bs} \right]_k , \tag{2}$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. Note that the subscript k applies to each of the variables inside the brackets. When the inverse of B_k is denoted by H_k then application of the Sherman-Morrison-Woodbury formula gives

$$H_{k+1} = \left[H + \left(1 + \frac{y^{\mathsf{T}}Hy}{s^{\mathsf{T}}y} \right) \frac{ss^{\mathsf{T}}}{s^{\mathsf{T}}y} - \left(\frac{sy^{\mathsf{T}}H + Hys^{\mathsf{T}}}{s^{\mathsf{T}}y} \right) \right]_k . \tag{3}$$

Equation (3) allows the direct calculation of the search direction without the need to solve the system of equations (1).

2.1 Implementations

There are many ways to implement the BFGS update formulae presented in equations (2) and (3). The four implementations discussed in this paper are introduced below. For each of these implementations the initial Hessian approximation (or its inverse) is set to the identity matrix.

Bupdate. Direct update of the approximate Hessian matrices B_k using equation (2) and solving the resulting system of equations by calculating the inverse B_k^{-1}. Note this method is never recommended in practice. It is used here to provide a guideline for the worst performance that would be expected from this type of implementation. Limited numerical trials shows it performed almost identically to more preferred implementations, using Gaussian elimination for example [2].

Hupdate. Direct updates of the inverses of the approximate Hessian matrices H_k using equation (3).

Cholesky. Cholesky factorisations of the approximate Hessian matrices B_k.

Conjugate. Conjugate factorisations of the inverses of the approximate Hessian matrices H_k.

The method of conjugate factorisation used in this paper is based on [3]; however, the idea is not new, see for example [6, 7]. A brief description is given below but the reader is referred to [3] for more details.

2.2 Conjugate factorisation

The BFGS update formula in equation (3) is written in product form as $H_{k+1} = \left[(I - pq^{\mathsf{T}})H(I - pq^{\mathsf{T}})^{\mathsf{T}}\right]_k$ where

$$q_k = \left[\frac{y}{p^{\mathsf{T}}y} \pm \frac{g}{\sqrt{-p^{\mathsf{T}}g\, p^{\mathsf{T}}y/\alpha}}\right]_k .$$

If the inverse Hessian approximation matrices are factored so that $H_k = C_k C_k^{\mathsf{T}}$ then the columns of C_k are B_k-conjugate and the search direction is $p_k = -C_k d_k$ where $d_k = C_k^{\mathsf{T}} g_k$. The elements of d_k are the directional

derivatives of f at x_k in the directions of the columns of C_k. The updated conjugate factors are

$$C_{k+1} = \left[C + \frac{pz^\mathsf{T}}{d^\mathsf{T}z} \mp \frac{pd^\mathsf{T}}{\sqrt{-d^\mathsf{T}d\, d^\mathsf{T}z\,/\alpha}} \right]_k , \tag{4}$$

and

$$d_{k+1} = \left[\bar{d} - \frac{d^\mathsf{T}\bar{d}\, z}{d^\mathsf{T}z} \pm \frac{d^\mathsf{T}\bar{d}\, d}{\sqrt{-d^\mathsf{T}d\, d^\mathsf{T}z\,/\alpha}} \right]_k , \tag{5}$$

where $\bar{d}_k = C_k^\mathsf{T} g_{k+1}$ and $z_k = C_k^\mathsf{T} y_k$ is the difference in the directional derivatives at x_{k+1} and x_k. There are two obvious implementations, one for each of the $\pm$ signs in equations (4) and (5). After limited numerical trials both were found to perform very similarly. The implementation described here uses the + sign in equation (4).

2.3 Practicalities

The Bupdate implementation requires $\mathcal{O}(n^3)$ operations at each iteration to update the second order information and compute the new search direction, whereas the remaining implementations require only $\mathcal{O}(n^2)$ operations. Additionally, the Cholesky implementation easily detects loss of positive definiteness of the approximate Hessian matrices. However, with a conjugate factorisation it is extremely unlikely that the inverse approximate Hessian matrices will lose positive definiteness. The worst that can happen is that they may become positive semi-definite when an eigenvalue is identically equal to zero. However, Powell [7] comments:

> We even find that, if we let Z [the conjugate factorisation matrix] be singular initially, then in practice the rounding errors of a sequence of updating calculations remove the singularity very successfully.

Thus if positive definiteness of the inverse approximate Hessian matrices is lost then it is extremely likely it will be restored at the next iteration; or conversely, it is extremely unlikely that positive semi-definiteness will be maintained for any length of time if conjugate factors are used.

3 Numerical results

Each of the BFGS implementations described above was tested on the 25 test functions listed in Tables 1 and 2 as the precision of the approximate Hessian information varied from 16 to two digits. More details on the test functions appear in [4, 5]. The varying levels of precision are achieved by truncating the elements of the approximate Hessian matrices (possibly in factored form, or their inverses) to the desired level. For example, the elements of the matrix X are truncated to n digits with $\text{trunc}(X) = 10^{-d}\lfloor 10^d X \rfloor$ where $d = n - \lceil \log_{10}(\max(|X|)) \rceil$.

Two safeguarded, quadratically interpolating strong Wolfe line searches are used with each of the BFGS implementations. At each iteration, α_k is chosen so that $x_{k+1} = x_k + \alpha_k p_k$ satisfies $f_{k+1} \leqslant f_k + \rho\alpha_k p_k^{\mathsf{T}} g_k$ and $|p_k^{\mathsf{T}} g_{k+1}| \leqslant \sigma |p_k^{\mathsf{T}} g_k|$ where the sufficient descent parameter $\rho = 10^{-4}$ and the gradient parameter σ is set to 10^{-3} and 0.9 for what are referred to in the remainder of this paper as *strict* and *standard* line searches.

For each test problem the number of function evaluations, final function value and execution time (in seconds) are recorded. The implementations are ranked by the number of test functions successfully solved (out of a possible total of 375). A test problem is deemed to be successfully solved if the termination criterion $\|\nabla f(x)\| \leqslant 10^{-6}$ is met. If necessary, the algorithms were then sub-sorted by the mean number of function evaluations. Any ties were sub-sorted by the mean accuracy of the approximations to the minimum function values. The accuracy is measured using $\log_{10}(f - f^*)$ where f^* represents the minimum of the function and f is the final function value.

TABLE 1: Low dimension test functions.

Function	Dim.	Initial point	Min.
Rosenbrock	2	$(-1.2, 1)$	0
Powell badly scaled	2	$(0, 1)$	0
Repeated Rosenbrock	4	$(-1.2, 1, -1.2, 1)$	0
Multi-dimensional Rosenbrock	4	$(-1.2, 1, -1.2, 1)$	0
Powell singular	4	$(3, -1, 0, 1)$	0

TABLE 2: Test functions for 8, 12, 20, 40 and 60 dimensions.

Function	Initial point	Min.
Repeated Rosenbrock	$(-1.2, 1, -1.2, 1, \ldots)$	0
Multi-dimensional Rosenbrock	$(-1.2, 1, -1.2, 1, \ldots)$	0
Powell singular	$(3, -1, 0, 1, \ldots)$	0
Hilbert quadratic	$(0, 0, 0, 0, \ldots)$	0

Note that only data for the problems solved successfully are used in the sorting process. As it is the "raw" performance of each implementation that is being investigated, the algorithms are terminated whenever they ran into difficulty, rather than applying some sort of corrective procedure. All of the implementations presented were run in a MATLAB R12.1 environment on a Sun-Fire-880 multi-user machine with four 750 MHz processors and 8 Gb of RAM running Solaris 8. MATLAB's built-in functions were used where convenient.

3.1 Limited precision second order information

The performance of each implementation as the precision of the second order information varied from 16 to two digits with the strict and standard line searches is now discussed. See the results in Tables 3 and 4, where the columns labelled Succ, Fcnt, Accy and Time represent the number of successfully solved test problems, the mean number of function evaluations, the

TABLE 3: Strict line search and varying second order precision.

Ranking	Method	Succ	Fcnt	Accy	Time
1	Conjugate	332	323.1	−13.9	2.4
2	Cholesky	326	361.2	−13.8	2.8
3	Bupdate	286	336.6	−13.7	2.8
4	Hupdate	269	313.0	−13.8	2.6

mean accuracy of the solutions and the mean execution time in seconds. Note that only the data for the test functions solved successfully are presented.

Strict line search. The number of successfully solved test problems ranged from 332 for Conjugate down to 269 for Hupdate. The mean number of function evaluations ranged from 313.0 for Hupdate through to 361.2 for Cholesky.

Standard line search. The number of successfully solved test problems ranged from 331 for Conjugate down to 267 for Hupdate. The mean number of function evaluations ranged from 150.6 for Hupdate through to 171.4 for Cholesky.

The results in Table 4 are presented graphically in Figure 1. The plot for the strict line search results (Table 3) is not presented as it is very similar. Figure 1 shows that differences in performance do not become noticeable until the precision of the second order information falls below eight digits.

3.2 Quadratic termination

For any member of the Broyden family of quasi-Newton methods (which includes the BFGS method), $B_{n+1} = G$ for any n-dimensional quadratic func-

TABLE 4: Standard line search and varying second order precision.

Ranking	Method	Succ	Fcnt	Accy	Time
1	Conjugate	331	159.0	−13.1	1.6
2	Cholesky	323	171.4	−13.0	1.9
3	Bupdate	289	152.1	−12.7	1.7
4	Hupdate	267	150.6	−12.9	1.5

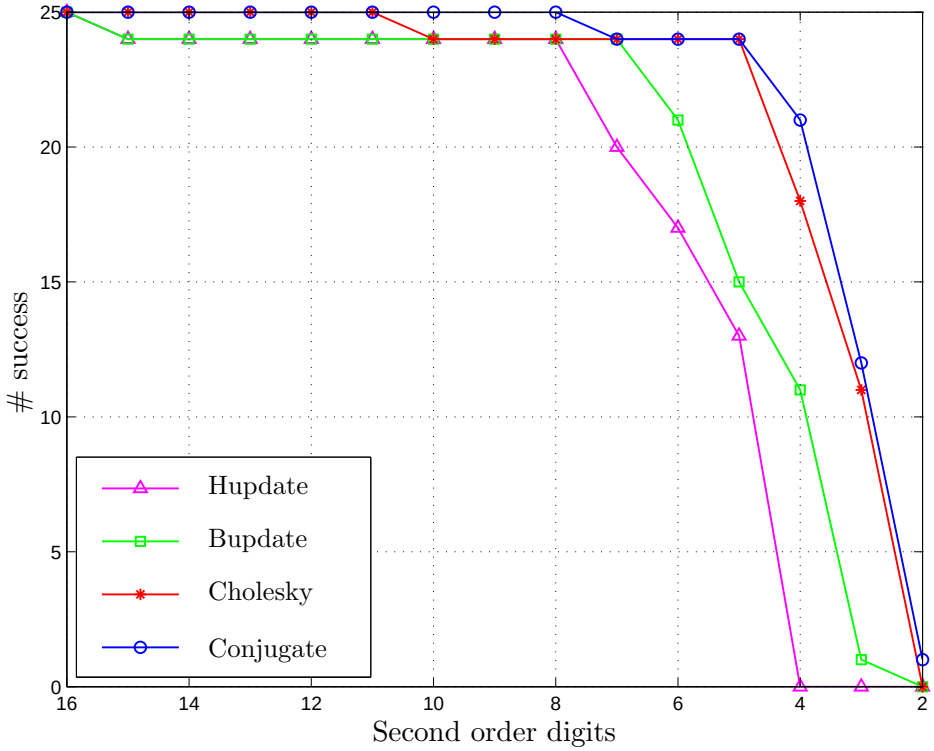

FIGURE 1: Performance of selected BFGS implementations with the standard line search and varying second order precision.

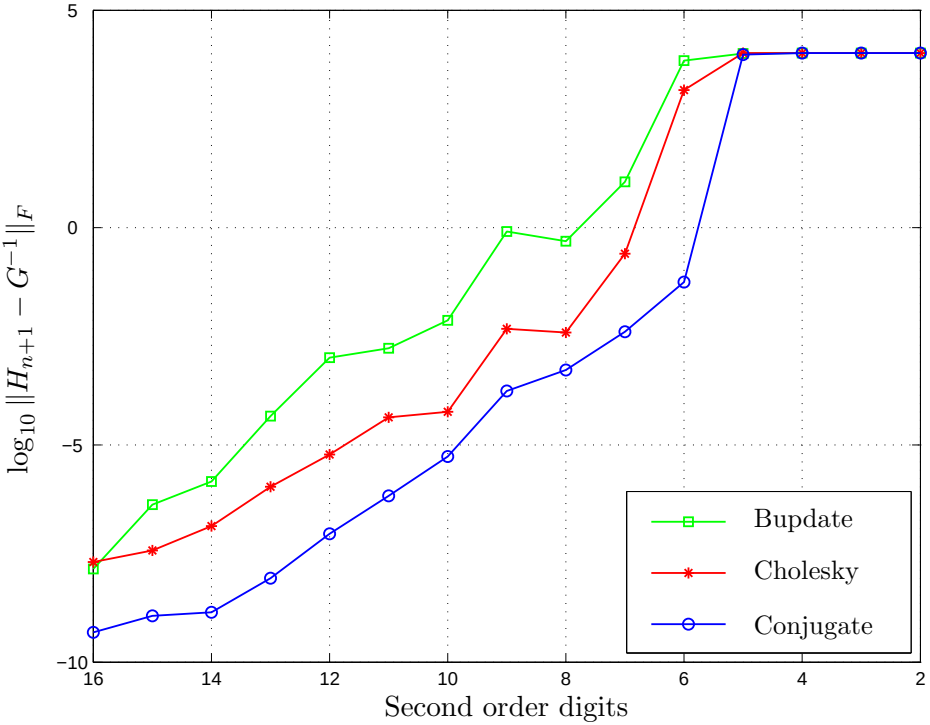

FIGURE 2: Difference in norm of approximate inverse Hessian and exact inverse Hessian after $n+1$ iterations with varying second order precision.

tion with Hessian matrix G when exact arithmetic and exact line searches are used.

Figure 2 shows $\log_{10}\|H_{n+1}-G^{-1}\|_F$ for each of the BFGS implementations with the four dimensional Hilbert quadratic and an accurate line search ($\sigma = 10^{-10}$). Note that $\|\cdot\|_F$ represents the Frobenius norm. The difference in norm of the inverse Hessian rather than the Hessian is used as the inverse Hessian allows the direct calculation of the search direction. Note also that the inverse Hessian is exact but the approximate inverse Hessian matrices H_k are truncated depending on the level of second order precision. The results for

Hupdate clutter the figure somewhat and are omitted; however, if included, the plot for Hupdate would oscillate between the lines for Cholesky and Conjugate.

Note that as the precision of the second order information falls below about five digits there is a plateau in Figure 2 with a height of about four. The height of this plateau coincides with the norm of the inverse Hessian of the four dimensional Hilbert quadratic ($\log_{10} \|G^{-1}\|_F \approx 4.0146$). Presumably once the precision of the second order information falls below a certain level there is insufficient information to approximate the inverse Hessian to any significant level. Similar results are produced with Hilbert quadratics of different dimensions. In higher dimensions the height of the plateau matches the norm of the inverse Hessian but the plateau starts at higher levels of second order precision. In lower dimensions the plateau effect is lost and the difference in the performance of each implementation is reduced.

4 Discussion and summary

The performance of four BFGS implementations on a suite of 25 test functions with two line searches (strict and standard) as the precision of second order information varied from 16 to two digits has been presented.

When second order information is available to single precision (eight digits), or better, there is no real advantage in any particular implementation. If second order information is available to reasonable precision then the straightforward inverse Hessian update of the BFGS method, Hupdate, produced results which are as accurate as those of any of the other methods considered and requires, on average, fewer function evaluations.

Figures 1–2 and Tables 3–4 clearly show the importance of a factorisation strategy as the precision of second order information is reduced. The conjugate factorisation implementation successfully solved significantly more test

problems with significantly fewer function evaluations than any of the other implementations presented here, including Cholesky factorisation. Additionally, the conjugate factorisation implementation produced better approximations to the inverse Hessian matrices of n-dimensional Hilbert quadratics when terminated after $n + 1$ iterations than any of the other methods.

A further advantage of conjugate factorisations is that grids based on these conjugate directions have useful theoretical and practical properties [1].

Acknowledgement: DB is financially supported by a Top Achiever Doctoral Scholarship.

References

[1] D. Byatt, I. D. Coope, and C. J. Price. Conjugate grids for unconstrained optimisation. Research report UCDMS2002/10, University of Canterbury, Christchurch, New Zealand, August 2002. C294

[2] D. Byatt, I. D. Coope, and C. J. Price. Performance of various BFGS and DFP implementations with limited precision second order information. Research report UCDMS2003/1, University of Canterbury, Christchurch, New Zealand, January 2003. C286

[3] I. D. Coope. A conjugate direction implementation of the BFGS algorithm with automatic scaling. *Journal of the Australian Mathematics Society Series B*, 31:122–134, 1989. C286

[4] L. Grandinetti. Factorization versus non-factorization in quasi-Newtonian algorithms for differentiable optimization. In *Third Symposium on Operations Research (University of Mannheim,*

Mannheim, 1978), pages 255–274, Konigstein, 1979. Hain. Section I. C285, C288

[5] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, March 1981. C288

[6] M. R. Osborne and M. A. Saunders. Descent methods for minimization. In R. S. Anderssen, L. D. Jennings, and D. M. Ryan, editors, *Optimization*, pages 221–237. University of Queensland Press, St. Lucia, 1972. C286

[7] M. J. D. Powell. Updating conjugate directions by the BFGS formula. *Mathematical Programming*, 38(1):29–46, 1987. C286, C287