Mathematical Programming 45 (1989) 407-435
North-Holland

SOME NUMERICAL EXPERIMENTS WITH VARIABLE-STORAGE QUASI-NEWTON ALGORITHMS

Jean Charles GILBERT*
International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria

Claude LEMARÉCHAL
Institut National de Recherche en Informatique et en Automatique, F-78153 Le Chesnay, France

This paper describes some numerical experiments with variable-storage quasi-Newton methods for the optimization of some large-scale models (coming from fluid mechanics and molecular biology). In addition to assessing these kinds of methods in real-life situations, we compare an algorithm of A. Buckley with a proposal by J. Nocedal. The latter seems generally superior, provided that careful attention is given to some nontrivial implementation aspects, which concern the general question of properly initializing a quasi-Newton matrix. In this context, we find it appropriate to use a diagonal matrix, generated by an update of the identity matrix, so as to fit the Rayleigh ellipsoid of the local Hessian in the direction of the change in the gradient.

Also, a variational derivation of some rank one and rank two updates in Hilbert spaces is given.

AMS (MOS) Subject Classifications: 49D05, 65K05.

Key words: Conjugate gradient, diagonal updates, Hilbert spaces, large-scale problems, limited memory, numerical experiments, unconstrained optimization, variable-metric algorithms, variable-storage.

1. Introduction

This paper reports on some numerical experiments with variable-storage quasi-Newton methods for finding a minimum of a smooth real-valued function f defined on $\mathbb{R}^n$.

Variable-storage quasi-Newton methods (VS methods for short) are intended for large-scale problems (that is, problems with a large number of variables, say, more than 500) when the Hessian of the objective function has no particular structure: in their general setting, these methods do not try to take advantage of the possible sparsity of the Hessian. It is thought that they may help in filling the gap between conjugate gradient (CG) and quasi-Newton (QN) methods. The former use few locations in memory, $O(n)$, but converge rather slowly and require expensive

Work supported in part by FNRS (Fonds National de la Recherche Scientifique), Belgium.

* Present address: Institut National de Recherche en Informatique et en Automatique, F-78153 Le Chesnay, France.

line-searches; on the other hand, QN methods have the converse features: fast rate of convergence (theoretically superlinear), no need for exact line-searches, but large memory requirement, namely $O(n^2)$ storage locations.

VS methods are based on the quasi-Newton principle: they use the change in the gradient to obtain information on the local Hessian of the objective function. However, they do not store any matrix of order n because this is supposed to be either impossible or too expensive. Rather, they are able to operate with a variable amount of storage, which is a *controlled* multiple of n. A small or a large amount of storage should make them resemble CG or QN methods respectively, which could thus be regarded as two extremes. The motivation is that it seems reasonable to expect an increase in the performance of a VS method if it uses more storage.

Among the papers dealing with VS methods, let us mention the works by Buckley (1978), Nazareth (1979, 1986), Nocedal (1980), Buckley and LeNir (1983) and Liu and Nocedal (1988). The papers by Perry (1976, 1977), Shanno (1978), Shanno and Phua (1978b) and Gill and Murray (1979) have also some connection to the subject.

The present study is definitely experimental and non-exhaustive: we apply a few VS methods to a few test-problems, and compare the numerical results. Our aim when starting this study was mainly a practical assessment of these methods. More precisely, we wanted to test their *actual* performance, when applied to meaningful problems (as opposed to the standard benchmarks generally used in the literature). This naturally implied first a comparison of various VS methods, between each other as well as against CG and QN.

We have selected 3 test-problems, which represent real-life applications, respectively in transonic fluid mechanics, meteorology and crystallography. Some of their internal parameters can be modified, so their dimension and/or conditioning can be varied. As a whole, 8 problems have thus been defined, with dimensions ranking from 34 to 1865 variables. In order to illustrate some particular algorithmic points, we have also used 4 purely academic problems, with 500 variables and a quadratic objective.

As for the methods, we have briefly tested that of Buckley and LeNir (1983) (which is rather close to conjugate gradient in its spirit) and its two extremes: pure QN and pure CG methods. Actually, we have focused our attention on the truly QN-like proposal of Nocedal (1980), which is based on the following simple idea: at the current iteration, a quasi-Newton direction is computed, which does not use all the information accumulated from the first iteration (there is no room for it); rather, only the most recent changes in the gradient are used. The number of such changes is kept under control, depending on the available memory. Based on this common principle, we consider several possible implementations.

In anticipation of our conclusions, we mention two phenomena, clearly exhibited by our study.

(i) An efficient implementation of Nocedal's technique requires a careful choice of the initial matrix. This is not surprising, since the influence of this "initial" matrix (which is recomputed at each descent iteration) is not damped by potentially many

updates. Accordingly, we have concentrated on this question, and some more theory should be done to complete our empirical study.

(ii) The second phenomenon is rather disappointing: when the available memory increases, the performance of a VS method does not improve much. Roughly speaking, they improve till a fairly small number of updates (say smaller than 20, apparently not depending on the number of variables); beyond that value, they stagnate, or even deteriorate.

The paper is organized as follows. In Section 2, we give some details on the algorithms mentioned above. The test-problems are briefly described in Section 3, where we also discuss some numerical experiments made with Buckley and LeNir (1983) and with QN on these problems. In Section 4, we introduce several ways of choosing an initial matrix for QN-type methods and we propose several formulae for updating diagonal matrices, compared via numerical experiments. In the Appendix, we show how to obtain some variable-metric update formulae in Hilbert spaces by means of a variational formulation.

We finish the present section with some remarks. First, our test-problems and codes are extracted from a French optimization library, called Modulopt. In their description below, they are given their library-name, in which M (minimizer) means "algorithm", U (user) means "test-problem" and 1 means "without constraints". More information, as well as copies of the programs, are available from the authors.

It is worth mentioning that using real-life test-problems does pose some difficulties: their actual solution is not exactly known (and they are of course not convex); the nature of the spectrum of the Hessian around a solution is not known either; the gradient may be inaccurate, due to rounding errors; and, last but not least, computation time and storage may be deterrent factors. However, it is these kinds of problems that are to be solved eventually, and their large scale is not artificially obtained.

Our main tool to compare speeds of convergence is the number of function-gradient evaluations (in Modulopt language, this operation is called a "simulation"). In fact, in sensible real-life problems, the computing time of a simulation largely dominates the overhead required by the algorithm itself. Furthermore, it is our experience in *all* such problems that, once the function is available, the gradient can be obtained with little additional computing time. Therefore, we have taken the point of view that function- and gradient-values are computed altogether (in the "simulator"), whenever necessary.

2. Some variable-storage quasi-Newton methods

2.1. Notation and background

Let x_* denote a local minimum of the objective function f. Quasi-Newton methods generate two sequences: a sequence $(x_k) \subset \mathbb{R}^n$ of approximations of x_* and a sequence (B_k) of bijective approximations of $B_* := \nabla^2 f(x_*)$, the Hessian of f at x_* (see, for example, Dennis and Moré, 1977). We shall suppose that B_* is positive

definite and we shall note $H_* := B_*^{-1}$. Starting with a couple (x_0, B_0), the sequences are calculated by:

$$x_{k+1} := x_k - \rho_k B_k^{-1} g_k, \tag{2.1}$$

$$B_{k+1} := U(B_k, y_k, s_k). \tag{2.2}$$

In (2.1), ρ_k is a positive stepsize determined by a search on f along the direction $d_k := -B_k^{-1} g_k$ and $g_k := g(x_k) := \nabla f(x_k)$ is the gradient of f at x_k. In (2.2), U represents an *update formula* that calculates B_{k+1} from B_k, using $y_k := g_{k+1} - g_k$ and $s_k := x_{k+1} - x_k$. If H_k is the inverse of B_k, it is generally possible to update H_k instead of B_k using the *inverse update formula* $\bar{U}$ of U: $H_{k+1} := \bar{U}(H_k, y_k, s_k)$. In that case, the direction is therefore $d_k := -H_k g_k$.

To write update formulae, we find it convenient to denote by $\langle \cdot, \cdot \rangle$ the scalar product on $\mathbb{R}^n$, $|\cdot|$ being its associated norm, and by $L(\mathbb{R}^n)$ the space of linear operators from $\mathbb{R}^n$ to $\mathbb{R}^n$. Then, we use the following *tensor product* of two vectors u and v (see Schwartz, 1981); it is the element of $L(\mathbb{R}^n)$ defined by:

$$u \otimes v : \mathbb{R}^n \to \mathbb{R}^n: \quad d \to (u \otimes v) d := \langle v, d \rangle u.$$

Remark. Usually, $\langle \cdot, \cdot \rangle$ is the standard dot-product ($\langle u, v \rangle = u^T v$), $L(\mathbb{R}^n)$ is the set of $n \times n$ matrices and $u \otimes v = uv^T$. However, more general situations are frequent in the context of large-scale optimization; see the Appendix for a motivation of our notation. For simplicity, we will still call "matrix" an operator $H \in L(\mathbb{R}^n)$ (this terminology simply implies the choice of a basis $(e_i)_{1 \leq i \leq n}$ and the identification of H with the components of the He_i's).

The BFGS *formula* is thought to be one of the best update formulae in optimization. With the preceding tensor product, it is written:

$$B_{k+1} := B_k + \frac{y_k \otimes y_k}{\langle y_k, s_k \rangle} - \frac{(B_k s_k) \otimes (B_k s_k)}{\langle s_k, B_k s_k \rangle}. \tag{2.3}$$

Throughout, U will stand for this formula and $\bar{U}$ will stand for the *inverse BFGS formula*:

$$H_{k+1} := H_k + \frac{(s_k - H_k y_k) \otimes s_k + s_k \otimes (s_k - H_k y_k)}{\langle y_k, s_k \rangle} - \frac{\langle s_k - H_k y_k, y_k \rangle}{\langle y_k, s_k \rangle^2} s_k \otimes s_k. \tag{2.4}$$

Formulae (2.3) and (2.4) have the property of transmitting the positive definiteness of B_k to B_{k+1} (resp. of H_k to H_{k+1}), if and only if $\langle y_k, s_k \rangle$ is positive. Having B_k positive definite is important to make d_k a descent direction of f at x_k. For this reason, the stepsize ρ_k in (2.1) is generally determined so that Wolfe's (1969) conditions are satisfied:

$$f(x_k + \rho_k d_k) \leq f(x_k) + \alpha_1 \rho_k \langle g_k, d_k \rangle, \tag{2.5}$$

$$\langle g(x_k + \rho_k d_k), d_k \rangle \geq \alpha_2 \langle g_k, d_k \rangle, \tag{2.6}$$

where $0<\alpha_1<\frac{1}{2}$ and $\alpha_1<\alpha_2<1$. Clearly, inequality (2.6) implies the positivity of $\langle y_k, s_k\rangle$.

In practice, the number n of variables may be large and it may turn out to be impossible or too expensive to store in memory the full current approximation H_k of the inverse Hessian. Because the initial matrix H_0 generally takes little space in memory (it is most commonly a positive multiple of the identity matrix) and because H_k is formed from H_0 and k couples $\{(y_i, s_i)\colon 0\leq i<k\}$, one can think of storing these elements instead of H_k and computing $H_k g_k$ by an appropriate algorithm. Of course, when the number of iterations increases, these pieces of information become more and more cumbersome in memory and we must get rid of some of the couples $\{(y_i, s_i)\colon 0\leq i<k\}$. A method will be called an *m-storage QN method* if only m of these couples are used to form H_k from an initial matrix. Note that in this type of method, the inverse update formula (2.4) is preferable to the direct update formula (2.3) because the inversion of B_k may be problematic.

All the VS methods we present hereafter fit into this framework and differ in the selection of the couples (y_i, s_i), in the choice of the starting matrix H_0, in the way $H_k g_k$ is computed and in the presence or absence of restarts.

2.2. *The algorithm of Shanno:* CONMIN

Motivated by the search for a conjugate gradient type method without exact line-searches, Shanno (1978) recommended, on the basis of a large amount of computational results, to use the following algorithm. It is in some way a generalization of the CG with Beale's (1972) restarts, using Perry's (1976) formulae. First, we set $r_k := 0$ (r_k will be the index of the last restart before iteration k). Suppose that we arrive at the current iteration $k\geq 1$, having on hand the couple (y_{k-1}, s_{k-1}). To complete this iteration, we must compute H_k. For this, we decide whether or not to restart.

If we restart, we set $r_k := k$ and we compute (see (2.4) for the definition of $\bar{U}$),

$$H_{r_k} := \bar{U}(\delta'_{r_k-1} I, y_{r_k-1}, s_{r_k-1}). \tag{2.7}$$

If, on the contrary, we do not restart but proceed with a "normal" iteration, then we set $r_k := r_{k-1}$ and compute

$$H_k := \bar{U}(H_{r_k}, y_{k-1}, s_{k-1}). \tag{2.8}$$

In (2.7), δ'_{r_k-1} is obtained by evaluating

$$\delta' := \langle y, s\rangle / |y|^2 \tag{2.9}$$

at $r_k - 1$. The algorithm is restarted at iteration k when *Powell's* (1977) *restart criterion* is satisfied, i.e. when $|\langle g_k, g_{k-1}\rangle| \geq 0.2|g_k|^2$. The scaling factor (2.9) was used by Shanno and Phua (1978a) who motivated it by the self-scaling ideas of Oren and Spedicato (1976). So, when $k > r_k$, the algorithm is clearly a 2-storage BFGS method using successively the couples (y_{r_k-1}, s_{r_k-1}) and (y_{k-1}, s_{k-1}) to build H_k.

It can be proved (see Shanno, 1978) that for f quadratic and exact line-searches, the search directions obtained by (2.7) and (2.8) with any scaling factor δ are identical to Beale's directions, scaled by δ. The advantage of Shanno's method over Beale's method is that it generates descent directions automatically without requiring exact line-searches, as long as $\langle y_k, s_k\rangle$ is positive at each iteration, which can be provided by the line-search.

This algorithm is a part of the code CONMIN, by which name we shall refer to it.

2.3. *The algorithm of Buckley and LeNir:* M1GC3

The algorithm of Shanno uses exactly two couples of vectors y and s to build its current approximation of the metric. Therefore, it cannot take advantage of extra locations that might be available in memory. The algorithm of Buckley and LeNir (1983) remedies this deficiency and may be seen as an extension of Shanno's method.

Following the presentation of the authors, we shall say that the algorithm is cyclic, each cycle being composed of a QN-part followed by a CG-part. The QN-part builds a preconditioner for the CG-part. The decision to restart a cycle is taken during the CG-part by using Powell's restart criterion. To be more specific, let us consider iteration k and suppose that the last restart occurred at iteration $r_k \leq k$. Let $m \geq 2$ be a fixed integer. If $k = r_k$, the algorithm takes:

$$H_{r_k} := \bar{U}(\delta'_{r_k-1} I, y_{r_k-1}, s_{r_k-1}),$$

with δ'_{r_k-1} evaluated by (2.9). If $r_k < k \leq r_k + m - 1$, the algorithm is in the *QN-part of the cycle* and takes:

$$H_k := \bar{U}(H_{k-1}, y_{k-1}, s_{k-1}).$$

If $k \geq r_k + m$, the algorithm is in the *CG-part of the cycle* and takes:

$$H_k := \bar{U}(H_{r_k+m-2}, y_{k-1}, s_{k-1}).$$

The CG-part is so called because, if the line-search is exact, $\langle g_k, s_{k-1}\rangle = 0$ and d_k is identical to the direction given by the CG formula, preconditioned by H_{r_k+m-2}.

We see that the number of couples (y, s) used to build H_k varies with k. For $r_k \leq k \leq r_k + m - 1$, the algorithm uses the $(k - r_k + 1)$ couples $\{(y_i, s_i)\colon r_k - 1 \leq i \leq k - 1\}$ and for $k \geq r_k + m$, it uses the m couples $\{(y_i, s_i)\colon r_k - 1 \leq i \leq r_k + m - 3\} \cup \{(y_{k-1}, s_{k-1})\}$. We also see from (2.8) that for $m = 2$, the matrices H_k are computed just as in Shanno's algorithm.

The line-search is briefly described in Section 2.6 and the resulting code, called M1GC3, is almost identical to the updated TOMS algorithm described in Buckley (1985, 1989).

2.4. *The algorithm of Nocedal:* M1QN2

The method proposed by Nocedal (1980) abandons the restart notion that the preceding algorithms inherited from the CG method and, as a result, is not cyclic.

If $m \geq 1$ is the desired number of updates (according to the storage available in memory), Nocedal proposes to build H_k by using always the last m couples (y, s): at each step, the oldest information (y_{k-m-1}, s_{k-m-1}) contained in the matrix is discarded and the new (y_{k-1}, s_{k-1}) is appended. An elegant procedure is given to apply H_k to a vector, using explicitly the m couples. This procedure is based on the use of the following form of the inverse BFGS formula:

$$H_{k+1} = \left(I - \frac{s_k \otimes y_k}{\langle y_k, s_k \rangle}\right) H_k \left(I - \frac{y_k \otimes s_k}{\langle y_k, s_k \rangle}\right) + \frac{s_k \otimes s_k}{\langle y_k, s_k \rangle} = \bar{U}(H_k, y_k, s_k).$$

To be more specific, H_k is obtained at iteration $k \geq m$ as follows. A matrix H_k^0 is supposed given and one computes:

$$H_k^{i+1} := \bar{U}(H_k^i, y_{k-m+i}, s_{k-m+i}), \quad 0 \leq i \leq m-1.$$

Then, $H_k := H_k^m$. For short, we shall denote this scheme by

$$H_k := \bar{U}_{k-m}^{k-1}(H_k^0), \tag{2.10}$$

to mean that H_k is obtained by updating H_k^0 using in order the m couples (y_i, s_i) for $i = k-m, \ldots, k-1$. With this notation, $H_k := \bar{U}_0^{k-1}(H_k^0)$ for $1 \leq k \leq m$.

2.5. The BFGS algorithm

In the tests below, we shall call BFGS the following algorithm. If n is smaller than 501, it is the classical BFGS method (a part of the code M1GC3) using $H_0 := I$, $H_1 := \bar{U}(\delta_0' I, y_0, s_0)$ and next $H_k := \bar{U}(H_{k-1}, y_{k-1}, s_{k-1})$ for $k \geq 2$. If n is larger than 500, it is the same algorithm but it is simulated by M1GC3 with m equal to the number of iterations.

The above expression for H_1 comes from the Oren-Spedicato (1976) preconditioning technique. It simulates the initialization $H_0 := \delta_0' I$ (remember that δ_0' is not known in advance!). Unless otherwise specified, it is the same H_1 that is used in all the methods tested below.

2.6. Line-searches

An important aspect in our tests is that all the results in the Tables below have been obtained with the same line-search procedure, outlined in Lemaréchal (1981). Starting from an initial ρ_k^0, safeguarded cubic approximation is used to find a ρ_k satisfying (2.5) and (2.6). The values for the slopes are $\alpha_1 = 0.001$ and $\alpha_2 = 0.9$. The safeguard is initialized to $\frac{1}{100}$th of the bracket and is increased (multiplied by 3) if it keeps active.

This is the general strategy for QN-like methods: M1QN2, BFGS and the QN-part of M1GC3. In this case, the initial stepsize is $\rho_k^0 = 1$ for $k \geq 1$ and $\rho_0^0 := 2\Delta_0 / |g_0|^2$, where Δ_0 is the expected decrease of f at the first iteration and is supplied by the user. This is justified when f is quadratic.

A slight complication occurs in the CG-part of M1GC3: in order to have a chance to catch the optimal stepsize when f is quadratic, at least one cubic approximation is made and then the general strategy is used.

3. The test-problems

As already mentioned, we have used a number of real-life problems, and some synthetic ones. Incidentally, it is interesting to note that they are essentially least-squares problems, so they do have a structure that could perhaps be exploited (not by a Gauss-Newton method, though: it would suffer the same memory difficulty). The corresponding programs are written in a fully portable Fortran, which should allow them to serve as benchmarks for new codes (actually, the reason for developing the Modulopt library lies here).

Although the final aim of an optimization code is to find a point of zero gradient, we shall not use a stopping criterion expressed in terms of the gradient, but rather we shall ask for a sufficient decrease of the objective function. The reason is that, contrary to $|g|$, f decreases monotonically. The value of f to be reached will be denoted by f_{stop}.

For all the codes, each time the function f is computed, so is its gradient. The number of function/gradient calls, i.e. the number of *simulations*, will be denoted by "simul" in the tables; "iter" will denote the number of iterations. The tests have been made on a SUN 3/60 (except those with UITSO, made on a PYRAMID 8920), using optimization codes in single precision.

Table 1 gathers the main characteristics of the test-problems, which are described with more details below: δ'_0 is the value δ' of formula (2.9) at $k=0$.

3.1. A problem in transonic fluid mechanics: UITSO

Our first problem is that of simulating the flow around a given object (say, the cross-section of an aircraft's wing). The mathematical model is: compute $\phi : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ solution of

$$\operatorname{div}[\rho(|\nabla\phi(\omega)|)\nabla\phi(\omega)] = 0, \quad \omega \in \Omega, \tag{3.1}$$

$$\rho(|\nabla\phi(\omega)|)\frac{\partial\phi(\omega)}{\partial n} = 0, \quad \omega \in \operatorname{bd}\Omega, \tag{3.2}$$

$$\nabla\phi(\omega) = u, \quad \omega \text{ at infinity.} \tag{3.3}$$

Here, Ω is the outside of the wing; the vector $\nabla\phi(\omega)$ is the speed of the fluid at $\omega \in \Omega$; $\partial/\partial n$ is the normal derivative and ρ is a given function from $\mathbb{R}$ to $\mathbb{R}$; u is also given in $\mathbb{R}^2$.

Table 1
The test-problems

Problems	n	$f(x_0)$	f_{stop}	δ_0'
U1TS0.1	403	$0.152\ 10^{-3}$	10^{-12}	3.70
U1TS0.2	403	$0.257\ 10^{-3}$	10^{-12}	3.57
U1TS0.3	403	$0.322\ 10^{-3}$	10^{-5}	3.34
U1TS0.4	403	$0.322\ 10^{-3}$	10^{-12}	3.34
U1MT1	1875	$0.124\ 10^{8}$	$0.62\ 10^{5}$	$0.119\ 10^{-3}$
U1CR1.1	34	$-0.198\ 10^{-1}$	$-0.20033\ 10^{-1}$	$0.105\ 10^{4}$
U1CR1.2	455	$-0.882\ 10^{-2}$	$-0.8876\ 10^{-2}$	$0.298\ 10^{5}$
U1CR1.3	1559	$-0.106\ 10^{-1}$	$-0.10625\ 10^{-1}$	$0.139\ 10^{5}$
EDEVB.1	500	$62.625\ 10^{3}$	10^{-5}	0.0025
EDEVB.2	500	$0.504\ 10^{16}$	10^{-5}	0.958
EDEVH.1	500	3.40	10^{-10}	1.11
EDEVH.2	500	$0.246\ 10^{5}$	10^{-10}	417

The problem is actually formulated as a least-squares one: to minimize an adequate norm (which acts as a preconditioner) of the left-hand side of (3.1), among all the functions ϕ satisfying (3.2) and (3.3). After discretization, there are 403 unknowns, which are the values of ϕ on the discretized Ω. See Bristeau et al. (1985) for more details.

The difficulty is that (3.1) is elliptic (resp. hyperbolic) if $|\nabla\phi| < 1$ (resp. > 1), and each case has its own integration scheme. Yet, the given U0 $= |u|$ (the Mach-number at infinity) is supposed to be slightly less than 1, so $|\nabla\phi|$ definitely crosses 1 at some (unknown!) points in Ω. As a result, the integration scheme depends on the unknowns, and this introduces unknown effects on the smoothness of f. From the numerical results below, however, these effects do not seem too troublesome.

What can be said is that the conditioning of the problem deteriorates when U0 increases. Accordingly, we have studied a number of instances, listed U1TS0.1 to U1TS0.4, in which U0 has the respective values 0.8, 0.9, 0.95, 0.95. Thus, U1TS0.3 and U1TS0.4 are the same model. Their difference lies in the value of f_{stop}, see Table 1.

Finally, we mention that f and ∇f are computed in double precision (otherwise, any algorithm produces a very inaccurate solution).

3.2. *A problem in meteorology:* U1MT1

Let now $\Omega \subset \mathbb{R}^2$ be a region on the surface of the earth, with points denoted by $(x, y) \in \Omega$. The coordinates u, v of the wind-speed and the atmospheric pressure p satisfy the model equation for $(x, y) \in \Omega$:

$$u_t' + uu_x' + vu_y' - v + p_x' = 0, \qquad v_t' + uv_x' + vv_y' + u + p_y' = 0. \tag{3.4}$$

Furthermore, a number of meteorological observations are known: the three functions (u, v, p) should satisfy

$$(u, v, p)(x_i, y_i, t_i) = U_i \in \mathbb{R}^3, \quad i \in I, \tag{3.5}$$

I being a finite set.

To obtain the problem called UIMTI, least-squares are used: one minimizes a weighted sum of the squared residuals in (3.4) and (3.5). With Ω discretized in 625 points, the number of variables is here $3 \times 625 = 1875$.

Remark. We are not claiming that the present approach gives the best meteorological forecast. Actually, UIMTI is an experimental model, already obsolete. For more details, see Nouailler (1987). Most convincing results are given in Courtier (1987), when Ω is the whole earth. Unfortunately, the corresponding model is of little interest for nonlinear programming codes: the objective function behaves pretty much like a quadratic.

3.3. A problem in crystallography: UICRI

Our last real-life problem is the so-called phase problem in X-ray crystallography. Without giving details on its physical origin, let us just say that it consists of computing the positions in the space of the atoms making up a given crystal (say a frozen protein) (see Hauptman and Karle, 1953; Klug, 1958).

Very roughly speaking, the function $f: \mathbb{R}^n \to \mathbb{R}$ to minimize is

$$f(x) := \int_0^1 p(x, \omega) \log p(x, \omega)\, d\omega + \sum_{k=1}^{m} [a_k^2(p) + b_k^2(p) - m_k^2]^2,$$

where $p: \mathbb{R}^n \times]0, 1[\to \mathbb{R}$ is a given function, namely

$$p(x, \omega) := \exp\left[-\sum_{i=1}^{n} x_{(i)} \cos(i\omega) \right],$$

the m_k are given and

$$a_k(p) := \int_0^1 p(x, \omega) \cos(k\omega)\, d\omega, \qquad b_k(p) := \int_0^1 p(x, \omega) \sin(k\omega)\, d\omega.$$

(Actually, ω is a 3-dimensional variable but here, the notations are simplified.) The atoms are computed with an accuracy depending on the number n of variables. We have used three cases, with $n = 34, 455, 1559$.

Remark. A difficulty is that f is highly oscillating, has a combinatorial number of stationary points, and varies in a fairly narrow range (see Table 1). Hence, we do not know whether f-values provide a good filter for optimization codes. Unfortunately, $|g|$-values are not so good either. Also, the model is such that computing f and g in double precision would be hazardous.

3.4. Two synthetic problems: EDEVB and EDEVH

Finally, we have taken as last test-problems, two quadratic functions with diagonal Hessian, namely:

$$f(x)=\tfrac{1}{2}\sum_{i=1}^{n} i(x_{(i)}-1)^2, \quad n=500, \qquad \text{for EDEVB,}$$

$$f(x)=\tfrac{1}{2}\sum_{i=1}^{n} \frac{1}{i}(x_{(i)}-1)^2, \quad n=500, \qquad \text{for EDEVH.}$$

Hence, the eigenvalues of the inverse Hessian in EDEVB are clustered near 0.002 in the interval [0.002, 1], while for EDEVH, they are equally distributed in [1, 500]. The converse is true for the Hessian B_*. Thus, we have two rather different structures in the Hessian. A second motivation for choosing these problems was to examine the influence of the initial matrix, i.e. δ_0' of formula (2.9). For this, we took two starting points: $x_0=0$ for EDEVB.1 and EDEVH.1 (then, δ_0' underestimates the eigenvalues of H_*). The starting points of EDEVB.2 ($x_{0(i)}=1+(100/i)^4$) and EDEVH.2 ($x_{0(i)}=1+(i/100)^4$) are such that δ_0' is now an overestimate of these eigenvalues, see Table 1.

3.5. First numerical results

In Table 2, we give numerical results obtained with the codes described in Section 2. The results obtained with CONMIN are not given because the principle of the method is the same as the one of MIGC3 with $m=2$. Differences may only come

Table 2

Performance (simul/iter) of MIGC3 and BFGS

	$m=2$	$m=5$	$m=10$	$m=20$	$m=50$	BFGS
U1TS0.1	161/82	158/81	150/80	144/82	78/63	66/64
U1TS0.2	551/281	324/195	327/215	311/234	335/243	198/187
U1TS0.3	209/105	145/104	127/99	106/95	97/93	96/91
U1TS0.4	458/232	369/219	313/201	293/201	249/202	197/192
U1MT1	277/143	273/142	254/149	232/153	208/167	192/190(**)
U1CR1.1	171/90	130/86	(***)	(***)	(***)	117/114
U1CR1.2	87/44	80/42	62/45	63/46	52/48	52/48
U1CR1.3	49/25	38/28	31/23	35/28	31/29	31/29(**)
EDEVB.1	162/81	155/81	148/83	135/87	109/102	116/114
EDEVB.2	251/125	307/158	270/143	304/189	295/195	515/257
EDEVH.1	162/83	137/77	125/82	108/86	124/117	190/188
EDEVH.2	149/75	136/68	111/56	98/49	94/47	94/47

(**) by MIGC3, (***) enough storage for BFGS.

from the line-search procedures and from adjustment of some parameters. We observed very similar results, indeed. We tested also a pure conjugate gradient method (VA14 from the Harwell library). Its only essential difference with CONMIN is that the direction is computed in a traditional CG form (not using Perry's, 1976, formulae): $d_k = -g_k + \alpha_k d_{k-1} + \beta_k d_{r_k}$. The results were similar to or worse than those of M1GC3 with $m = 2$.

The results in Table 2 enable us to recover some of the conclusions of Buckley and LeNir (1983): (i) there is a reasonable trend for the number of simulations simul to decrease as m increases but (ii) this rule may be invalidated in some cases. As in their test-problems, we observe that (iii) the BFGS method is not always the best and that (iv) the number of iterations iter has a tendency to increase with m.

However, these results do not enable us to infer that the performance of VS methods can be expected to improve by increasing m. Indeed, conclusion (i) is mainly due to the difference in the line-search options during the CG-part (at least two simulations are required per iteration) and the QN-part (the first trial stepsize may be and often is accepted) of algorithm M1GC3. As m increases, the algorithm is more and more often in the QN-part (because it takes m iterations per cycle and the CG-part usually lasts only a few iterations) and therefore, the ratio simul/iter decreases as m increases. Hence, even when iter increases slightly, simul decreases.

Remark. The results obtained by the BFGS method on the problems EDEVB and EDEVH bring out the importance of the choice of the starting matrix; especially since a close examination of the output has shown that the matrix H_k (more precisely its diagonal) changes very slowly. Consequently, when the initial scaling given by δ'_0 is large with respect to the possible values of δ' at the solution, the step $|d_k|$ remains too large during all the run. This can be observed on EDEVB.2 (and EDEVH.2), for which $\delta'_0 \approx 0.958$ is large in $[0.002, 1]$. As a result, the ratio simul/iter is close to 2 because the unit stepsize is rejected by inequality (2.5). On the contrary, this argument disappears when δ'_0 is small because inequality (2.6) with $\alpha_2 = 0.9$ is not very constricting. This shows that the scaling by δ' may give bad results if the initial point happens to be unfortunate.

On the other hand, the influence of the initial scaling factor δ'_0 for M1GC3 is rather difficult to interpret and would require a detailed study. A comprehensive explanation should take into account the difference in the line-search policies during the CG and QN-parts, the presence of restarts (also observed for quadratic functions) that rescale the matrices by readapting the factor δ', and the structure of the spectrum of the Hessian.

4. Experiments with the method of Nocedal

The compressed form (2.10) of Nocedal's algorithm outlines the need of a choice for the starting matrices H_k^0. This choice is common to any QN method, but here

it can be and has to be made at each iteration. We shall try to take advantage of this flexibility by adapting H_k^0 to the information contained in the current couple (y, s), at each iteration. Doing this avoids situations described in the remark above, where an inappropriate initial scaling condemns all the run. We shall successively take and test H_k^0 as a multiple of the identity matrix and as a diagonal matrix. All the optimization codes (versions of M1QN2 and M1QN3) tested in this section only differ by this choice of H_k^0.

Throughout this section, y and s will denote two vectors in $\mathbb{R}^n$ with $\langle y, s\rangle$ positive, y being the change in the gradient of f for a displacement s.

4.1. Scaling the identity

Some preliminary tests clearly showed that taking $H_k^0 = I$ may be very unsuitable. We do not report these results here. Instead, we consider first the cheap choice of taking H_k^0 as a positive multiple of the identity.

Because it is usually impossible to satisfy the QN equation, $Hy = s$, with an H of the form δI, one may consider to satisfy it in some direction v. Projecting the QN equation in a direction v such that $\langle y, v\rangle \neq 0$ gives for δ, the value $\delta_v := \langle s, v\rangle / \langle y, v\rangle$. If v belongs to $V := \{\alpha y + \beta s\colon \alpha \geq 0, \beta \geq 0, \alpha\beta \neq 0\}$, $\langle y, v\rangle$ and $\langle s, v\rangle$ are positive and by the Cauchy-Schwarz inequality, we have

$$\delta' := \frac{\langle y, s\rangle}{|y|^2} \leq \delta_v \leq \frac{|s|^2}{\langle y, s\rangle} =: \delta''. \tag{4.1}$$

Note that realizing the QN equation in norm corresponds to taking $\delta = \delta_v$ with $v = y/|y| + s/|s|$ in V. From (4.1), δ_v reaches its minimum value δ' and its maximum value δ'' in V along the rays $\{\alpha y\colon \alpha > 0\}$ and $\{\beta s\colon \beta > 0\}$, respectively.

The value δ' is used as scaling factor in Shanno's and Buckley and LeNir's methods. Here are some of its properties.

Property $\mathbf{P}_1'$. *δ' is the Rayleigh quotient of $\bar{H}$ in the direction y, i.e. $\delta' = \langle \bar{H}y, y\rangle / |y|^2$, where*

$$\bar{H} := \left(\int_0^1 \nabla^2 f(x + ts)\, \mathrm{d}t \right)^{-1}. \qquad \square \tag{4.2}$$

Indeed, with $\bar{B} := \bar{H}^{-1}$, we have $y = \bar{B}s$ by Taylor's theorem. This property is useful since $\delta' I$ has to approximate $\bar{H}$.

Property $\mathbf{P}_2'$. *δ' minimizes in $\delta \in \mathbb{R}$, the norm $|\delta y - s|$.* $\square$

Therefore, $\delta' I$ is the least-squares solution of the QN equation $Hy = s$, for H multiple of the identity.

Property P_3'. *δ' minimizes for $\delta \geq 0$, the 2-norm condition number of $\bar{U}(\delta I, y, s)$, the inverse BFGS update.* □

Therefore, starting with $\delta' I$ may be a wise choice. This result can be found in Oren and Spedicato (1976), for δ restricted to $[\delta', \delta'']$. The result still holds for nonnegative δ. In fact, Oren and Spedicato give a more general result, which states that $\delta^\theta := [\theta/\delta' + (1-\theta)/\delta'']^{-1}$ minimizes in $\delta \in [\delta', \delta'']$, the condition number of $\bar{U}_\theta(\delta I, y, s) := \theta \bar{U}(\delta I, y, s) + (1-\theta) U(\delta I, s, y)$, the θ-Broyden's update of δI.

Property P_4'. *δ' is the unique solution of the following problem:*

$$\min_{\delta \in \mathbb{R}} \min_{H \in Q(s,y)} \|\delta I - H\|_F, \tag{4.3}$$

where $Q(s, y) := \{H \in L(\mathbb{R}^n): Hy = s\}$.

Proof. This property means that $\delta' I$ is the multiple of the identity that is closest to $Q(s, y)$ for the Frobenius norm

$$\|B\|_F := \left(\sum_{i=1}^{n} |Be_i|^2 \right)^{1/2}$$

(where $(e_i)_{1 \leq i \leq n}$ is an orthonormal basis of $\mathbb{R}^n$ for the scalar product $\langle \cdot, \cdot \rangle$, see the Appendix). To show this, let us remark that problem (4.3) is equivalent to

$$\min_{\delta \in \mathbb{R}} \|\delta I - H_\delta\|_F,$$

where $H_\delta := \delta I + (s - \delta y) \otimes y / |y|^2$ $(y \neq 0)$ is the inverse Broyden update of δI (see Proposition A.1 in the Appendix). Since $\|\delta I - H_\delta\|_F = \|(s - \delta y) \otimes y\|_F / |y|^2 = |s - \delta y| / |y|$, the result follows from Property P_2'. □

Property P_5'. *δ' is the unique solution of the following problem:*

$$\min_{\delta > 0} \min_{K \in S(s,y)} \|\delta^{1/2} I - K\|_F, \tag{4.4}$$

where $S(s, y) := \{K \in L(\mathbb{R}^n): K^* K y = s\}$.

Proof. Note that $S(s, y) \neq \emptyset$ because $\langle y, s \rangle$ is positive (see Proposition A.3 in the Appendix). To prove the property, let us remark that problem (4.4) is equivalent to

$$\min_{\delta > 0} \|\delta^{1/2} I - K_\delta\|_F, \tag{4.5}$$

where, $K_\delta := \delta^{1/2} I \pm y \otimes s / (|y| \langle y, s \rangle^{1/2}) - \delta^{1/2} y \otimes y / |y|^2$ (see Dennis and Schnabel, 1981). Now, using an orthonormal basis $(e_i)_{1 \leq i \leq n}$, we get:

$$\|\delta^{1/2} I - K_\delta\|_F^2 = \sum_{i=1}^{n} \left(\frac{\langle s, e_i \rangle}{\langle y, s \rangle^{1/2}} \pm \frac{\delta^{1/2} \langle y, e_i \rangle}{|y|} \right)^2 = \frac{|s|^2}{\langle y, s \rangle} \pm \frac{2\delta^{1/2} \langle y, s \rangle^{1/2}}{|y|} + \delta.$$

Therefore, the minimum in (4.5) is obtained for $\delta^{1/2} = \pm \langle y, s \rangle^{1/2} / |y|$, i.e. $\delta = \delta'$. □

All these properties seem to indicate that δ' is the good scaling factor. However, by exchanging y and s, we obtain similar properties for δ''.

Property P_1''. *$1/\delta''$ is the Rayleigh quotient of $\bar{B}$ in the direction s, i.e. $1/\delta'' = \langle \bar{B}s, s\rangle/|s|^2$, where $\bar{B}$ is defined in Property P_1'.* □

This property is also useful since $1/\delta'' I$ has to approximate $\bar{B}$.

Property P_2''. *δ'' minimizes in $\delta \in \mathbb{R}$, $\delta \neq 0$, the norm $|y - s/\delta|$.* □

Therefore, $1/\delta'' I$ is the least-squares solution of the QN equation $y = Bs$, for B multiple of the identity.

Property P_3''. *δ'' minimizes in $\delta \geq 0$, the 2-norm condition number of $U(\delta I, s, y)$, the inverse DFP update.* □

Property P_4''. *δ'' is the unique solution of the following problem:*

$$\min_{\substack{\delta \in \mathbb{R} \\ \delta \neq 0}} \min_{B \in Q(y,s)} \|1/\delta I - B\|_F. \qquad \square$$

This means that $1/\delta'' I$ is the multiple of the identity that is closest to $Q(y, s)$ for the Frobenius norm.

Property P_5''. *δ'' is the unique solution of the following problem:*

$$\min_{\delta > 0} \min_{C \in S(y,s)} \|1/\delta^{1/2} I - C\|_F. \qquad \square$$

Because the BFGS update is used in algorithm (2.10), the only property among those given above that can help to choose a scaling factor δ is Property P_3', which favors δ'. However, the argument is decidedly slim and some numerical experiments are welcome. They are shown in Table 3, where the results of M1QN2.A ($H_k^0 = \delta'_{k-1} I$ for $k \geq 1$), M1QN2.B ($H_k^0 = \delta'_0 I$ for $1 \leq k \leq m$ and $H_k^0 = \delta'_{k-m} I$ for $k > m$) and M1QN2.C ($H_k^0 = \delta''_{k-1} I$ for $k \geq 1$) are given one above the other.

These results show that when M1QN2.A reaches f_{stop}, it is always better than M1QN2.C for the number of function evaluations. Therefore, the choice of the scaling factor δ' is more suitable than δ''. This seems due to the fact that δ'', which is larger than δ', is generally too large, which is revealed in Table 3 by a ratio simul/iter close to two for M1QN2.C: one or two interpolations are often necessary to reduce the initial unit stepsize. These interpolations may make each iteration more efficient and may decrease the number of iterations, as for the quadratic functions EDEVB and EDEVH, but not enough to reduce the global cost of the runs, which is better

Table 3

Performance (simul/iter) of M1QN2.A, M1QN2.B and M1QN2.C

	$m=1$	$m=2$	$m=5$	$m=10$	$m=20$	$m=50$
U1TS0.1	122/115	104/102	82/78	77/72	74/71	69/67
	122/115	114/107	107/95	82/73	77/70	69/68
	179/95	175/93	125/71	144/80	122/67	115/63
U1TS0.2	482/459	488/459	308/294	265/256	223/210	215/204
	482/459	562/515	291/270	287/258	228/207	210/195
	1579/804	1148/586	582/243	631/268	490/221	509/211
U1TS0.3	152/147	118/114	104/99	99/96	102/93	96/90
	152/147	129/113	107/99	108/99	100/95	97/93
	285/145	238/127	203/96	172/90	174/88	161/81
U1TS0.4	355/339	310/300	250/242	243/236	225/215	207/200
	355/339	327/298	244/222	266/243	222/205	210/202
	713/369	628/336	511/256	472/232	382/192	352/177
U1MT1	223/208	205/191	165/161	163/157	159/155	165/156
	223/208	214/196	180/160	188/160	183/162	190/172
	410/206	450/220	355/163	357/147	359/146	344/145
U1CR1.1	(*)	104/96	87/82	91/83	64/62	51/50
	(*)	107/91	95/86	89/81	74/71	91/90
	219/108	184/93	154/76	123/66	92/50	59/38
U1CR1.2	67/62	57/53	52/48	50/47	48/46	47/45
	67/62	57/51	54/49	56/47	50/46	50/49
	86/48	87/47	88/48	80/43	78/42	70/38
U1CR1.3	(*)	30/28	30/26	26/23	23/21	23/21
	(*)	35/30	28/25	25/24	29/27	30/29
	49/24	46/23	49/24	40/20	32/16	32/16
EDEVB.1	109/106	109/106	100/93	92/91	93/88	93/88
	109/106	109/101	97/90	104/92	103/92	105/99
	177/108	158/88	157/86	168/93	151/82	149/81
EDEVB.2	365/343	333/313	222/211	211/202	210/198	177/167
	365/343	328/297	293/259	282/228	303/226	279/175
	515/276	432/216	504/252	468/234	398/199	326/163
EDEVH.1	207/184	164/149	140/125	109/101	87/82	63/61
	207/184	177/151	131/124	119/108	106/99	126/123
	234/142	204/121	186/105	140/85	103/65	63/46
EDEVH.2	145/129	131/124	119/109	105/94	85/78	71/63
	145/129	135/116	125/105	120/88	112/73	109/49
	190/112	169/106	165/95	142/83	117/66	82/48

(*) fails to reach f_{stop}.

measured by the number of simulations. On the other hand, the factor δ' gives a good scaling of the matrix as shown by the fact that the ratio simul/iter is close to one.

Comparison between M1QN2.A and M1QN2.B shows that, in general, it is better to use the more recent information, δ'_{k-1}, rather than older one, δ'_{k-m}, even though it is the latter that can be interpreted in terms of Property P'_3. The difference is particularly noticeable on EDEVB.2 and EDEVH.2 where the phenomenon observed with the BFGS method crops up again: the matrix is initially overestimated and, as no corrections are made by M1QN2.B during the first m iterations, the ratio simul/iter is closer and closer to two when m becomes large (for $m=1$, M1QN2.A and M1QN2.B are the same algorithm).

Finally, we observe that M1QN2.A has a general tendency to improve with m but that, as for M1GC3, this rule may be violated. Furthermore, this tendency decreases for large m.

If we compare the performance of M1GC3 and M1QN2.A, we see that the number of function evaluations required by the latter is almost always smaller, although the converse is generally true for the number of iterations, which can be attributed to the forced interpolation made at some iterations by the line-search procedure of M1GC3. These results are clearly in favor of M1QN2.A, particularly when m is small.

Remark. When $m=1$, M1QN2.A has sometimes difficulties in reaching f_{stop}. A star in Table 3 means that, at some k, the line-search procedure was not able to find a stepsize satisfying (2.5) and (2.6). This phenomenon is exclusively due to (roundoff) errors in f and g, and we believe that chance plays a large part in an algorithm failing for this cause. However, when m is small, the matrix H_k of M1QN2 is built by using only the last few couples (y, s), which are not very reliable when roundoff prevails. So, it may be argued in this case that the direction d_k may not be a good search direction: if high accuracy is required, M1QN2 should not use too small values for m.

4.2. *Diagonal starting matrices*

In this subsection, we suppose that the user knows an orthonormal basis $(e_i)_{1\leq i\leq n}$ of $\mathbb{R}^n$ for the scalar product $\langle\cdot,\cdot\rangle$ and that he can easily pass between this basis and the canonical one.

We shall say that a matrix D is *diagonal* with respect to the scalar product $\langle\cdot,\cdot\rangle$ and the orthonormal basis $(e_i)_{1\leq i\leq n}$, if $\langle De_i, e_j\rangle = 0$ for $i\neq j$. For any matrix H, we note $H^{(i)} := \langle He_i, e_i\rangle$ (note that a diagonal corresponds to our visual concept only when $\langle\cdot,\cdot\rangle$ is the familiar dot-product). A diagonal matrix D is self-adjoint, and it is positive definite if and only if the elements $D^{(i)}$ are positive. To represent and update diagonal matrices D_k, only the n diagonal elements $(D_k^{(i)})_{1\leq i\leq n}$ need be stored and updated.

Remark. It is interesting to note that the representation formula

$$D = \sum_{i=1}^{n} D^{(i)} e_i \otimes e_i$$

shows that it would be very useful to have for $(e_i)_{1 \leq i \leq n}$ an orthogonal basis formed by the eigenvectors of $\bar{H}$ (see (4.2)): in this case, taking $D^{(i)} = \langle s, e_i \rangle / \langle y, e_i \rangle$ would give $D = \bar{H}$.

For $H \in L(\mathbb{R}^n)$, diag H is the diagonal matrix defined by $(\text{diag } H)^{(i)} = H^{(i)}$, for $1 \leq i \leq n$. If H is positive definite, so is diag H. In this subsection, we will take for H_k^0 in (2.10) a positive definite diagonal matrix, which we will denote by D_k (or simply D). Its inverse is also diagonal and $(D^{-1})^{(i)} = 1/D^{(i)}$.

The numerical results of Section 4.1 strongly suggest that the marginal profit of an additional update is typically poor. This is our main motivation for taking a diagonal initial matrix: with n locations in memory, it may be a definitely better improvement than an additional update (which needs $2n$ more locations). Furthermore, a diagonal matrix may give a good approximation of the Rayleigh ellipsoid of $\bar{H}$, which constitutes a complete description of $\bar{H}$.

Our strategy for defining D_k will be as follows. Instead of the double sequence $(x_k, H_k)_{k \geq 1}$, we now construct the triple sequence $(x_k, D_k, H_k)_{k \geq 1}$. To define D_k, we use a special QN update (preserving diagonal property), say $D_k := V(D_{k-1}, y_{k-1}, s_{k-1})$, starting from $D_1 := \delta_0' I$. Once D_k is computed, H_k can be computed via (2.10) from $H_k^0 := D_k$. We have tested several possibilities, only differing by the choice of the formula V.

In M1QN3.A, we diagonalize the inverse BFGS formula (2.4): $D_+ := \text{diag } \bar{U}(D, y, s)$. Thus,

$$D_+^{(i)} = D^{(i)} + \left(\frac{1}{\langle y, s \rangle} + \frac{\langle Dy, y \rangle}{\langle y, s \rangle^2} \right) \langle s, e_i \rangle^2 - \frac{2 D^{(i)} \langle y, e_i \rangle \langle s, e_i \rangle}{\langle y, s \rangle}. \tag{4.6}$$

In M1QN3.B, formula V is obtained by diagonalizing the direct BFGS formula (2.3): $D_+^{-1} := \text{diag } U(D^{-1}, y, s)$. We have:

$$D_+^{(i)} = \left(\frac{1}{D^{(i)}} + \frac{\langle y, e_i \rangle^2}{\langle y, s \rangle} - \frac{(\langle s, e_i \rangle / D^{(i)})^2}{\langle D^{-1} s, s \rangle} \right)^{-1}. \tag{4.7}$$

In M1QN3.C, it is the inverse DFP formula that is diagonalized: $D_+ := \text{diag } U(D, s, y)$, so

$$D_+^{(i)} = D^{(i)} + \frac{\langle s, e_i \rangle^2}{\langle y, s \rangle} - \frac{(D^{(i)} \langle y, e_i \rangle)^2}{\langle Dy, y \rangle}. \tag{4.8}$$

These formulae were inspired by an idea of Gill and Murray (1979), who proposed to diagonalize the direct BFGS update after having done the substitution $Bs = -\rho g$. This substitution gives, however, a formula different from (4.7) and it is not clear

Table 4

Performance (simul/iter) of M1QN3.A, M1QN3.B and M1QN3.C

	$m=1$	$m=2$	$m=5$	$m=10$	$m=20$	$m=50$
U1TS0.1	236/112	194/105	158/88	150/81	97/59	80/51
	83/82	73/72	67/66	60/59	56/55	55/54
	131/96	92/77	65/60	71/61	59/54	51/50
U1TS0.2	(*)	(*)	(*)	(*)	(*)	(*)
	386/382	257/254	229/225	181/171	173/166	170/163
	676/398	479/292	463/256	419/233	333/195	306/180
U1TS0.3	280/144	261/129	222/108	174/98	156/91	150/98
	107/105	89/87	89/84	90/85	90/86	89/84
	145/122	119/113	94/84	97/86	93/83	88/80
U1TS0.4	(*)	(*)	1429/354	682/254	492/216	411/207
	274/272	203/201	209/204	185/180	171/167	168/163
	407/296	318/248	272/206	235/184	209/177	200/174
U1MT1	(*)	(*)	(*)	(*)	(*)	(*)
	201/196	241/239	176/174	183/181	173/171	171/169
	365/277	276/218	241/191	195/173	196/167	173/153
U1CR1.1	(*)	(*)	(*)	825/87	961/75	124/61
	95/94	90/89	84/83	82/81	80/79	78/77
	127/86	103/73	98/68	85/60	74/54	59/49
U1CR1.2	34/33	40/39	37/36	37/36	36/35	36/35
	(*)	51/50	46/45	48/47	(*)	47/46
	34/33	40/39	37/36	37/36	37/36	37/36
U1CR1.3	27/26	28/27	26/25	26/25	26/25	26/25
	38/37	34/33	30/29	30/29	30/29	31/30
	26/25	28/27	27/26	27/26	26/25	26/25
	35/34	36/35	35/34	34/33	33/32	33/32
	70/69	51/50	53/52	54/53	52/51	52/51
	36/35	37/36	36/35	36/35	35/34	35/34
EDEVB.2	248/73	246/74	426/132	453/145	399/134	393/132
	122/53	130/58	137/62	138/63	131/60	135/62
	556/176	469/154	566/185	667/220	651/217	603/202
EDEVH.1	104/103	91/90	91/90	90/89	90/89	89/88
	696/695	209/208	250/249	206/205	183/182	189/182
	106/105	98/97	97/96	96/95	95/94	95/94
EDEVH.2	92/42	96/45	94/44	90/42	90/42	86/40
	48/24	48/24	46/23	46/23	46/23	46/23
	91/42	92/44	86/41	84/40	82/39	82/39

(*) fails to reach f_{stop}.

whether it transmits positive definiteness from D to D_+, so some safeguard is necessary. On the other hand, the sole standard conditions (positive definiteness of D and positivity of $\langle y, s\rangle$) suffice for (4.6)-(4.8) to imply positive definiteness of D_+.

In Table 4, results with M1QN3.A, M1QN3.B and M1QN3.C are given one above the other.

These results enable us to make the following observations: (i) performance depends very much on the formulae used to update the diagonal matrix (see EDEVB.2, for instance); but (ii) it is not always the same formula that gives the best results (if M1QN3.B is the best minimizer for EDEVB.2 and EDEVH.2, it is the worst one for EDEVB.1 and EDEVH.1); however (iii) for each test-problem, there is generally one of the three minimizers that gives better results than M1QN2.A, which shows that obtaining a good diagonal starting matrix for algorithm (2.10) may improve the results.

Despite some occasional good results, M1QN3.A should be discarded for the following reason (a similar argument has been given by Byrd, Nocedal and Yuan, 1987, concerning the trace of the DFP formula). The right hand side of (4.6) updates D by using two correcting terms. The first one is positive (D and $\langle y, s\rangle$ are supposed positive), while the sign of the second one depends on the sign of the components of y and s in the basis (e_i). As $\langle y, e_i\rangle$ and $\langle s, e_i\rangle$ have no reason to have the same sign, the diagonal elements of D_k may have a trend to increase during the minimization. The large number of simulations needed during the line-search to reduce the stepsize reflects this phenomenon (recall the remark at the end of Section 3.5). Of course when the function is quadratic with a positive definite diagonal Hessian, then $\langle y, e_i\rangle$ and $\langle s, e_i\rangle$ have the same sign and the last term in (4.6) is negative. In this case, the previous argument does not apply: we see that results obtained by M1QN3.A on EDEVB and EDEVH are good.

Results obtained with M1QN3.B and M1QN3.C are more difficult to interpret and we do not have any convincing argument to decide between them. Let us just mention that we have observed with formula (4.7) a definite ability to decrease the elements of D when δ' decreases, while formula (4.8) has the ability to increase them when δ'' increases. This observation allows us to understand the difference in the results obtained with the two formulae (at least on EDEVB and EDEVH): M1QN3.B (resp. M1QN3.C) has better results when D_1 overestimates (resp. underestimates) H_*.

4.3. *Scaling the starting diagonal*

The previous analysis shows that, as with the BFGS method, the diagonal updates (4.7)-(4.8) suffer from the inability to modify rapidly a (diagonal) matrix. The temptation is great, therefore, to scale D before updating it. This ideas is at the root of M1QN3.B2 and M1QN3.C2, which are versions corresponding to M1QN3.B and M1QN3.C respectively. In both of them, before updating D, we multiply it by a factor σ such that σD has the good Rayleigh quotient in the direction y, i.e. δ'; in other words, $\sigma = \langle y, s\rangle / \langle Dy, y\rangle$.

With this scaling factor, formulae (4.7) and (4.8) become respectively:

$$D_+^{(i)} = \left(\frac{\langle Dy, y\rangle}{\langle y, s\rangle D^{(i)}} + \frac{\langle y, e_i\rangle^2}{\langle y, s\rangle} - \frac{\langle Dy, y\rangle (\langle s, e_i\rangle / D^{(i)})^2}{\langle y, s\rangle \langle D^{-1}s, s\rangle} \right)^{-1} \tag{4.9}$$

and

$$D_+^{(i)} = \frac{\langle y, s\rangle D^{(i)}}{\langle Dy, y\rangle} + \frac{\langle s, e_i\rangle^2}{\langle y, s\rangle} - \frac{\langle y, s\rangle (D^{(i)}\langle y, e_i\rangle)^2}{\langle Dy, y\rangle^2}. \tag{4.10}$$

Results obtained with M1QN3.B2 (formula (4.9)) and M1QN3.C2 (formula (4.10)) are given one above the other in Table 5.

Table 5

Performance (simul/iter) of M1QN3.B2 and M1QN3.C2

	$m=1$	$m=2$	$m=5$	$m=10$	$m=20$	$m=50$
U1TS0.1	79/78	86/79	70/65	63/60	59/56	58/55
	294/269	167/156	103/100	80/77	68/65	58/56
U1TS0.2	254/246	256/243	181/170	208/195	166/161	166/156
	(*)	(*)	(*)	(*)	(*)	(*)
U1TS0.3	97/94	88/85	86/81	88/80	88/80	83/77
	166/160	207/199	141/135	134/129	128/120	113/106
U1TS0.4	237/228	219/208	195/188	189/177	167/158	159/153
	(*)	(*)	549/533	429/414	354/343	266/257
U1MT1	227/214	174/165	158/151	148/143	144/140	143/140
	(*)	(*)	(*)	(*)	(*)	347/334
U1CR1.1	(*)	61/57	57/56	52/51	46/45	43/42
	(*)	(*)	(*)	77/68	68/64	54/48
U1CR1.2	41/39	41/37	38/36	37/35	37/35	37/35
	33/32	37/36	35/33	34/32	34/32	34/32
U1CR1.3	21/20	20/19	19/18	18/17	18/17	18/17
	19/18	20/19	18/17	18/17	18/17	18/17
EDEVB.1	52/50	46/44	46/44	46/43	46/43	45/42
	38/37	42/39	40/36	37/34	37/35	36/34
EDEVB.2	83/79	78/72	66/61	67/62	67/61	66/61
	67/62	59/56	55/49	53/48	52/47	53/47
EDEVH.1	46/44	54/52	48/47	47/45	47/46	47/45
	47/46	53/50	47/46	48/46	47/46	46/44
EDEVH.2	54/51	58/52	49/46	49/46	49/45	48/45
	50/47	54/48	48/44	47/43	47/43	46/42

(*) fails to reach f_{stop}.

We see that M1QN3.B2 generally works better than M1QN3.B. On the other hand, if M1QN3.C2 sometimes improves M1QN3.C, it has great deficiencies on some test-problems. Therefore, formula (4.10) should not be used.

Remark. For $m = 50$, consider the runs where iter $\leqslant 50 = m$. Then, any of our VS methods reduces to BFGS, except that at each iteration the whole sequence of updates is recomputed from the very beginning because H_1 is changed at each k (except for M1QN2.B, and barring roundoff errors). A comparison of Table 5, say, with Table 2 shows how this "updated initialization" can be beneficial; see the quadratic problems, in particular!

To conclude this section, we shall say that M1QN2.A and M1QN3.B2 seem to be the variants that work best. Table 6 compares them, as well as M1GC3, to BFGS: each entry is the total number of simulations required for the 8 real-life test-problems, divided by the corresponding number of simulations required by BFGS (namely 949, obtained from Table 2).

Table 6
VS methods vs. the BFGS method for the number of simulations

	$m = 2$	$m = 5$	$m = 10$	$m = 20$	$m = 50$
M1GC3	2.07	1.60	1.46	1.37	1.23
M1QN2.A	1.49	1.14	1.07	0.97	0.92
M1QN3.B2	1.00	0.85	0.85	0.76	0.74

5. Discussion and conclusion

This study had two primary aims. First, we wanted to determine whether optimization methods could really be useful for real-life large-scale applications. This question is important because the frequency of such problems is growing quite fast, even in the nonlinear world. Our selected test-examples may not be considered as fully convincing from a physical point of view (simplified models, moderately large numbers of variables, problems not exactly in the scope of nonlinear programming, ...); but we believe that our experiments do demonstrate the viability of QN-like methods in this context. Some other, more significant, demonstrations can be cited, for example in fluid mechanics: global models for mid-term meteorological forecast, Courtier (1987); integration of 3-dimensional Navier-Stokes equations, Bristeau et al. (1987); turbulence, Ortegón Gallego (1988) (the latter is particularly spectacular: 10^5 variables, function-gradient computed in a quarter of an hour on

a Cray-1). We also mention some still unpublished works done at Michelin in the tire industry, which are of extreme difficulty.

Second, we wanted to test Nocedal's proposal against its alternative, Buckley and LeNir's algorithm (we had no pretension to exhaustivity; for example, other possibilities like in Nazareth (1986) have not been considered). Here, we feel that several important conclusions must be drawn.

(i) In Nocedal's approach, an initial matrix H_k^0 is needed at each iteration k. For rather obvious reasons, its role is overwhelming (at least when the number of updates is not large). As a result, standard initializations used for classical quasi-Newton methods are not sophisticated enough: some more is mandatory.

This paper has investigated a few possibilities, among which M1QN3.B2 has given best results. Of course, the experience is limited so far; on a purely empirical basis, it can certainly not be claimed that the ideal method is on hand. Indeed, the results obtained by Zimmermann (1989) on some examples in molecular biology do not allow to decide clearly between M1QN2.A and M1QN3.B2. A finer mathematical analysis of diagonal update formulae is wanted, and it is not clear whether formula (4.9) would stand up such an analysis. The merit of our study was to raise the question and to check some possible initial choices.

For instance, it could have been thought that formula (4.6), obtained by diagonalization of the inverse BFGS formula, would be as good as formula (4.7), obtained by diagonalization of the direct BFGS formula. Numerical experiments have shown that this is not true. Other diagonal update formulae have also been tested, some by taking the variational viewpoint used to obtain matrix updates, but few work well.

(ii) The phenomenon (i) is even perceivable in a pure QN method: knowing that BFGS can be worse than a VS method, one is bound to conclude that "something must be done". In contradiction with the admitted policy, this "something" might well be the preconditioning of Oren and Spedicato (1976).

(iii) The above observations are relevant for CG methods as well: they are particular instances of VS methods, and they are also variants of QN methods. Thus, it might be interesting for example to investigate the effect of a preconditioning technique in a CG code like CONMIN. More generally, and more importantly, still the same technique might improve a Buckley and LeNir's method, which is really CG-like.

(iv) This last point is rather important because it exhibits an unfair aspect of our comparison M1GC3 versus M1QN2: in fact, we have spent much work on improving the latter, without a counterpart on the former. Yet, if the preconditioning technique suggested in (iii) were inserted in M1GC3, the differences would probably be changed. We have not made any experiment along this line, however: to obtain definitive results, a definitive (but still unknown) diagonal scaling should be on hand. We add for completeness that Nocedal's variant has been incorporated as an option in Buckley's code, see Buckley (1989).

(v) The role of an increased memory on the efficiency of a VS method is far from what could be expected beforehand. To say the least, the performance does

not increase significantly when more information is used. To eliminate this rather frustrating observation, one needs to return to the basics and question the whole quasi-Newton principle itself.

For example, could it be that all the differences of gradients are not equally important to compute the quasi-Newton direction? Then, how can we recognize the more beneficial such differences, if any? How can we use them best? This kind of question is underlying in Fletcher (1988).

Another point is that most QN-like methods modify the approximation of the Hessian by low rank corrections. As a result, when $H_* - H_k$ is a high rank matrix, a few QN updates have little effect on the improvement of H_k. On the other hand, scalings, which are full rank corrections, have a determining importance on the performance. In other words, least-change updates may be too shy in the present context and should perhaps be replaced by high rank stable updates.

Acknowledgement

We are indebted to J. Nocedal and to anonymous referees, who made several interesting observations, in particular concerning points (ii), (iii) and (v) in the conclusion.

Appendix. A derivation of least-change secant update formulae in Hilbert space

Here, we show how to extend to Hilbert spaces the variational derivation of some rank one and rank two quasi-Newton formulae. What we want is to obtain formulae valid for any scalar product, and not only for the usual dot-product of $\mathbb{R}^n$. There are several motivations for this generalization; one is that optimization theory is independent of any system of coordinates (while a dot-product does depend on such a system). More importantly, the formulae then become more suitable for various real-life situations entering the framework of this paper.

For example, a large-scale problem, say

$$\min\{f(x): x \in \mathbb{R}^n\} \tag{A.1}$$

is often the discretized version of a continuous one, say

$$\min\{J(u): u \in \mathbb{H}\}. \tag{A.2}$$

Admitting that (A.2) is well-posed, it is then strongly advised to use for $\mathbb{R}^n$ (i.e. the discretized $\mathbb{H}$) a scalar product consistent with that of $\mathbb{H}$. Otherwise, the conditioning of (A.1) is likely to deteriorate when n increases. A typical example is when $\mathbb{H}$ is some Sobolev space, involving derivatives of u; then the norm on $\mathbb{R}^n$ has to involve differences such as $x_{(i+1)} - x_{(i)}$. Note in particular that so is the case with UITSO of Section 3.1.

More generally, even if it is purely finite-dimensional, the actual problem to be solved may have a natural preconditioner estimating $\nabla^2 f$. The latter can be introduced via a change of coordinates, or via a suitable scalar product. Both ways are equivalent, but one may be more practical for the user.

In practice, all this means that the user of an optimization code may wish to provide two subroutines: one to compute $f(x)$ and $\nabla f(x)$ for given x, knowing that there holds

$$f(x+h)=f(x)+\langle \nabla f(x), h\rangle + o(|h|).$$

The scalar product in this expression is computed in the second subroutine provided by the user.

The formulae we shall derive are not new. They can, indeed, be found in the book by Gruver and Sachs (1980). However, our approach is different. While these authors obtain the formulae by selecting, in the family of perturbations of rank one or two, one that gives the desired properties (QN property, symmetry, positive definiteness), we shall adopt the more classical and more elegant variational point of view, showing that the formulae still give least-change updates.

Thus, let $\mathbb{H}$ be a Hilbert space over $\mathbb{R}$ with a real scalar product $\langle \cdot, \cdot \rangle$ and its associated norm $|\cdot|$. For $B \in L(\mathbb{H})$, the space of linear continuous operators on $\mathbb{H}$, consider the norm

$$\|B\|_{HS} := \left(\sum_{i \in I} |Be_i|^2 \right)^{1/2} \tag{A.3}$$

where $(e_i)_{i \in I}$ is an orthonormal basis of $\mathbb{H}$.

Remark. There is no difficulty in assuming that $\mathbb{H}$ is infinite-dimensional, so there is no reason to hold back from this setting—even though it is rather anecdotal when numerical algorithms are concerned. In other words, I may be an infinite set in (A.3). However, (A.3) makes sense only if at most countably many of the Be_i are nonzero and if the series is convergent. This is not satisfied for all operators B in $L(\mathbb{H})$ (take $B=I$!), but if it is, the sum does not depend on the choice of the orthonormal basis. The resulting norm is called the *Hilbert-Schmidt* (HS) norm of B, which generalizes the Frobenius norm. An operator with finite HS norm is called a *Hilbert-Schmidt operator.* It is continuous (and even compact). The norm (A.3) actually defines a scalar product and the set of HS operators is a Hilbert space, which we shall denote by $L_2(\mathbb{H})$.

Observe for example that the tensor product $u \otimes v$ introduced at the beginning of Section 2 is in $L_2(\mathbb{H})$ (it has finite rank!). Also, if B_1 and B_2 are linear continuous on $\mathbb{H}$ and if one of them is an HS operator, then $B_1 B_2$ is HS as well.

For this elementary material, see for example the book by Weidmann (1980). Of course, if $\mathbb{H}$ is finite-dimensional, $L(\mathbb{H})$ and $L_2(\mathbb{H})$ are trivially identical.

We note some useful relations concerning HS operators ($\|\cdot\|$ denotes the usual operator-norm of $L(\mathbb{H})$, R^* is the adjoint of R):

$$\begin{aligned} &\|B\| \leq \|B\|_{\mathrm{HS}}, \\ &\|u \otimes v\|_{\mathrm{HS}} = \|u \otimes v\| = |u||v|, \\ &R(u \otimes v) = (Ru) \otimes v \quad \text{and} \quad (u \otimes v)R = u \otimes (R^* v). \end{aligned} \tag{A.4}$$

The problem we consider is the following. Being given two vectors y and s in $\mathbb{H}$ and $B \in L(\mathbb{H})$, we look for an updated operator $B_+ \in L(\mathbb{H})$, the closest to B in some sense and verifying the *secant equation*

$$y = B_+ s. \tag{A.5}$$

We write $B_+ = B + P$ and we restrict the perturbation P to be a Hilbert-Schmidt operator, so that the perturbation can be measured with the norm (A.3). Hence, P is supposed to belong to

$$\Pi := \{P \in L_2(\mathbb{H}): y = (B+P)s\}.$$

Then, we consider the following minimization problem:

$$\min_{P \in \Pi} \|R_1 P R_2\|_{\mathrm{HS}} \tag{A.6}$$

where R_1 and R_2 are fixed in $L(\mathbb{H})$ (then the norm in (A.6) makes sense), and bijective. The next result shows that this problem has a unique solution, independent of R_1 and depending on R_2 only through $c := R_2^{-*} R_2^{-1} s$.

Proposition A.1. *Let $\mathbb{H}$ be a Hilbert space and B, R_1 and R_2 be operators in $L(\mathbb{H})$ with R_1 and R_2 bijective. Let y and s be two vectors in $\mathbb{H}$ with $s \neq 0$. Then, problem* (A.6) *has a unique solution P_c, given by*

$$P_c := (y - Bs) \otimes c / \langle c, s \rangle,$$

where $c := R_2^{-*} R_2^{-1} s$.

Proof. It is straightforward to check that $P_c \in \Pi$. Also, knowing that R_1 and R_2 are bijective, uniqueness is classical for the projection onto an affine subspace Π of a Hilbert space $L_2(\mathbb{H})$. For $P \in \Pi$, set $E := R_1 P R_2$ and $E_c := R_1 P_c R_2$. We have to prove that $\|E_c\|_{\mathrm{HS}} \leq \|E\|_{\mathrm{HS}}$. With $z := R_2^{-1} s = R_2^* c \neq 0$ and $y - Bs = Ps$, we write

$$E_c = (Ez) \otimes z / |z|^2$$

and it follows from (A.4),

$$\|E_c\|_{\mathrm{HS}} = |Ez|/|z| \leq \|E\| \leq \|E\|_{\mathrm{HS}}. \qquad \square$$

The solution of problem (A.6) with $R_2 = I$ gives *Broyden's update formula*:

$$B_{\mathrm{Broyden}} := B + (y - Bs) \otimes s / |s|^2.$$

We consider now the case where B is self-adjoint, i.e. $B = B^*$, and we look for an updated self-adjoint operator $B_+ \in L(\mathbb{H})$ satisfying (A.5). This time, the perturbation operator P is supposed to belong to

$$\Pi_s := \{P \in L_2(\mathbb{H}): P = P^*, y = (B+P)s\}.$$

In this framework, we find it convenient to particularize R_1 and R_2 of (A.6), and our minimization problem becomes:

$$\min_{P \in \Pi_s} \|R^*PR\|_{\mathrm{HS}}, \tag{A.7}$$

where $R \in L(\mathbb{H})$ is bijective. Then, the situation is quite similar to the nonsymmetric case.

Proposition A.2 (Dennis and Moré). *Let B and R be two linear continuous operators on a Hilbert space $\mathbb{H}$, with B self-adjoint and R bijective. Let y and s be two vectors in $\mathbb{H}$ with $s \neq 0$. Then, problem* (A.7) *has a unique solution P_c, given by*

$$P_c := \frac{(y-Bs)\otimes c + c\otimes(y-Bs)}{\langle c, s\rangle} - \frac{\langle y-Bs, s\rangle}{\langle c, s\rangle^2} c\otimes c, \tag{A.8}$$

where $c := R^{-*}R^{-1}s$.

Proof. It is a straightforward adaptation of the proof of Theorem 7.3 of Dennis and Moré (1977). We proceed as in Proposition A.1: for $P \in \Pi_s$, we set $E := R^*PR$ and $E_c := R^*P_cR$. We have to prove that $\|E_c\|_{\mathrm{HS}} \leq \|E\|_{\mathrm{HS}}$. With $z := R^{-1}s = R^*c$, write

$$E_c = \frac{E(z\otimes z) + (z\otimes z)E}{|z|^2} - \frac{\langle Ez, z\rangle}{|z|^4} z\otimes z.$$

Choose $e_i \in \mathbb{H}$ for $i \in J$, such that $\{z/|z|\} \cup (e_i)_{i\in J}$ forms an orthonormal basis of $\mathbb{H}$ (this is possible, see Weidmann, 1980, Theorem 3.10). First, we have $E_cz = Ez$, so $|E_cz| = |Ez|$. Also, for $i \in J$,

$$E_ce_i = \frac{z\otimes z}{|z|^2} Ee_i,$$

which implies with (A.4) that $|E_ce_i| \leq |Ee_i|$.

Piecing together, the desired result follows from the very definition (A.3). □

If we take $R = I$ in problem (A.7), we obtain the *PSB update formula*:

$$B_{\mathrm{PSB}} = B + \frac{(y-Bs)\otimes s + s\otimes(y-Bs)}{|s|^2} - \frac{\langle y-Bs, s\rangle}{|s|^4} s\otimes s.$$

We turn now to the positive definite case: can we add the constraint "$B+P$ positive definite" in the definition of Π_s? The following proposition gives a general existence result.

Proposition A.3. *Let y and s be two nonzero vectors in a Hilbert space $\mathbb{H}$. Then, the following statements are equivalent:*

(i) *there exists $B_+ \in L(\mathbb{H})$, self-adjoint, positive definite, such that $y = B_+s$;*

(ii) *there exists* $C \in L(\mathbb{H})$, *bijective, such that* $y = C^*Cs$;
(iii) $\langle y, s\rangle$ *is positive.*

Proof. As (i)$\Rightarrow$(iii) and (ii)$\Rightarrow$(i) are clear, it remains to prove (iii)$\Rightarrow$(ii). So, suppose that $\langle y, s\rangle$ is positive. We follow Dennis and Schnabel (1981) and take in (ii):

$$C := I + \frac{s \otimes y}{|s|\langle y, s\rangle^{1/2}} - \frac{s \otimes s}{|s|^2}.$$

Clearly, $y = C^*Cs$. As a finite rank perturbation of the identity, C is bijective if it is injective (Fredholm's alternative, Weidmann, 1980, Theorem 6.8); but the latter is true because $Cz = 0$ implies $z = \alpha s$ with $\alpha \in \mathbb{R}$ and $\alpha Cs = 0$ implies $\alpha = 0$, hence $z = 0$. □

Thus, if $\langle y, s\rangle$ is positive, and if B is self-adjoint, positive definite, then there is $P \in \Pi_s$ such that $B + P$ is positive definite. Indeed, with C as in the proof of Proposition A.3, take $R = C^{-1}$ in Proposition A.2. The resulting c is just y and formula (A.8) yields the *DFP update formula*:

$$B_{\mathrm{DFP}} := B + \frac{(y - Bs) \otimes y + y \otimes (y - Bs)}{\langle y, s\rangle} - \frac{\langle y - Bs, s\rangle}{\langle y, s\rangle^2}\, y \otimes y. \tag{A.9}$$

Its positive definiteness is proved by verifying that (A.9) is equivalent to

$$B_{\mathrm{DFP}} = \left(I - \frac{y \otimes s}{\langle y, s\rangle}\right) B \left(I - \frac{s \otimes y}{\langle y, s\rangle}\right) + \frac{y \otimes y}{\langle y, s\rangle}$$

and that the latter is positive definite.

If in (A.9), we exchange y and s on the one hand and if we change B by H on the other hand, we recover the BFGS formula (2.3).

References

E.M.L. Beale, "A derivation of conjugate gradients," in: F. Lootsma, ed., *Numerical Methods for Non-linear Optimization* (Academic Press, London, 1972) pp. 39–43.

M.O. Bristeau, O. Pironneau, R. Glowinski, J. Périaux, P. Perrier and G. Poirier , "On the numerical solution of nonlinear problems in fluid dynamics by least squares and finite element methods (II). Application to transonic flow simulations," *Computer Methods in Applied Mechanics and Engineering* 51 (1985) 363–394.

M.O. Bristeau, R. Glowinski and J. Périaux, "Numerical methods for the Navier–Stokes equations," in: R. Gruber, ed., *Finite elements methods in physics, Computer Physics Report, Vol. 6* (North-Holland, Amsterdam, 1987).

A. Buckley, "A combined conjugate gradient quasi-Newton minimization algorithm," *Mathematical Programming* 15 (1978) 200–210.

A. Buckley, "Algorithm 630: BBVSCG—A variable-storage algorithm for function minimization," *ACM Transactions on Mathematical Software* 11 (1985) 103–119.

A. Buckley, "Remark on algorithm 630," *ACM Transactions on Mathematical Software* 15 (1989) 262–274.

A. Buckley and A. LeNir, "QN-like variable storage conjugate gradients," *Mathematical Programming* 27 (1983) 155–175.

R.H. Byrd, J. Nocedal and Y.-X. Yuan, "Global convergence of a class of quasi-Newton methods on convex problems," *SIAM Journal on Numerical Analysis* 24 (1987) 1171–1190.

P. Courtier, "Application du contrôle optimal à la prévision numérique en météorologie," Thesis, University of Paris VI (Paris, 1987).

J.E. Dennis and J. J. Moré, "Quasi-Newton methods, motivation and theory," *SIAM Review* 19 (1977) 46-89.

J.E. Dennis and R.B. Schnabel, "A new derivation of symmetric positive definite secant updates," in: O.L. Mangasarian, R.R. Meyer and S.M. Robinson, eds., *Nonlinear Programming, Vol. 4* (Academic Press, New York, 1981) pp. 167-199.

R. Fletcher, "Low storage methods for unconstrained optimization," NA Report 117, University of Dundee (Dundee, UK, 1988).

P.E. Gill and W. Murray, "Conjugate gradient methods for large scale nonlinear optimization," Technical Report SOL 79-15, Department of Operations Research, Stanford University (Stanford, CA, 1979).

W.A. Gruver and E. Sachs, *Algorithmic Methods in Optimal Control, Research Notes in Mathematics, Vol. 47* (Pitman, London, 1980).

H. Hauptman and J. Karle, *Solution of the phase-problem. I: The centrosymmetric crystal, American Crystallographic Association, Monograph 3* (Polycrystal Book Service, Pittsburgh, PA, 1953).

A. Klug, "Joint probability distributions of structure factors and the phase problem," *Acta Crystallographica* 11 (1958) 515-543.

C. Lemaréchal, "A view of line-searches," in: A. Auslender, W. Oettli and J. Stoer, eds., *Optimization and optimal control, Lecture Notes in Control and Information Science, Vol. 30* (Springer, Heidelberg, 1981) pp. 59-78.

D.C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," Technical Report NAM 03, Department of Electrical Engineering and Computer Science, Northwestern University (Evanston, IL, 1988).

J.L. Nazareth, "A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms," *SIAM Journal on Numerical Analysis* 16 (1979) 794-800.

J.L. Nazareth, "Conjugate gradient methods less dependent on conjugacy," *SIAM Review* 28/4 (1986) 501-511.

J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Mathematics of Computation* 35/151 (1980) 773-782.

A. Nouailler, "Assimilation de données à petite échelle: technique de contrôle optimal," Thesis, University of Clermont Ferrand (Clermont Ferrand, 1987).

S. Oren and E. Spedicato, "Optimal conditioning of self-scaling variable metric algorithms," *Mathematical Programming* 10 (1976) 70-90.

F. Ortegón Gallego, "Estudio de un modelo matemático de turbulencia obtenido mediante técnicas de homogeneización," Thesis, University of Sevilla (Sevilla, 1988).

A. Perry, "A modified conjugate gradient algorithm," Discussion Paper 229, Center for Mathematical Studies in Economics and Management Science, Northwestern University (Evanston, IL, 1976).

A. Perry, "A class of conjugate gradient algorithms with a two-step variable metric memory," Discussion Paper 269, Center for Mathematical Studies in Economics and Management Science, Northwestern University (Evanston, IL, 1977).

M.J.D. Powell, "Restart procedures for the conjugate gradient method," *Mathematical Programming* 12 (1977) 241-254.

L. Schwartz, *Les Tenseurs* (Hermann, Paris, 2nd ed., 1981).

D.F. Shanno, "Conjugate gradient methods with inexact searches," *Mathematics of Operations Research* 3/3 (1978) 244-256.

D.F. Shanno and K.-H. Phua, "Matrix conditioning and nonlinear optimization," *Mathematical Programming* 14 (1978a) 149-160.

D.F. Shanno and K.-H. Phua, "Numerical comparison of several variable metric algorithms," *Journal of Optimization Theory and Applications* 25 (1978b) 507-518.

J. Weidmann, *Linear operators in Hilbert spaces, Graduate Texts in Mathematics, Vol. 68* (Springer, Heidelberg, 1980).

P. Wolfe, "Convergence conditions for ascent methods," *SIAM Review* 11/2 (1969) 226-235.

K. Zimmermann, "ORAL: All-purpose molecular mechanics simulator & energy minimizer," in preparation (1989).