International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

Some Numerical Experiments with Variable Storage Quasi-Newton Algorithms

Gilbert, J.C. and Lemarechal, C.

IIASA Working Paper

WP-88-121

August 1988

WORKING PAPER

SOME NUMERICAL EXPERIMENTS WITH VARIABLE STORAGE QUASI-NEWTON ALGORITHMS

Jean Charles Gilbert
Claude Lemaréchal

August 1988
WP-88-121

IIASA
International Institute
for Applied Systems Analysis

SOME NUMERICAL EXPERIMENTS WITH VARIABLE STORAGE QUASI-NEWTON ALGORITHMS

Jean Charles Gilbert
Claude Lemaréchal

August 1988
WP-88-121

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

FOREWORD

This paper relates some numerical experiments with variable storage quasi-Newton methods for the optimization of large-scale models. The basic idea of the recommended algorithm is to start bfgs updates on a diagonal matrix, itself generated by an update formula and adjusted to Rayleigh's ellipsoid of the local Hessian of the objective function in the direction of the change in the gradient.

A variational derivation of some rank one and rank two updates in Hilbert spaces is also given.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

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)

Abbreviated title. Variable storage QN algorithms.

Key words. Conjugate Gradient, Diagonal Updates, Large-scale Problems, Limited Memory, Unconstrained Optimization, Variable Metric Algorithms, Variable Storage.

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

---

* Work supported in part by INRIA (Institut National de Recherche en Informatique et en Automatique), France and in part by the FNRS (Fonds National de la Recherche Scientifique), Belgium.

1. Introduction

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

These methods are intended for large-scale problems (that is to say, problems with a large number of variables, say, more than 500) when the Hessian of the objective function has no particular structure. In particular, in their general setting, these methods do not try to take advantage of the possible sparsity of the Hessian. It is thought that this type of algorithms may help in filling the gap between, on the one hand, conjugate gradient (CG) methods, which use few locations in memory, $O(n)$, but converge rather slowly and require exact line search, and, on the other hand, quasi-Newton (QN) or variable metric methods, which have the converse features: fast rate of convergence (theoretically superlinear), no exact line search requirement but cumbersome in memory since $O(n^2)$ storage locations are needed.

Variable storage QN algorithms are QN-like methods in the sense that 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 typically is a multiple of n. A priori, if CG and QN methods are regarded as two extremes, a small (resp. a large) amount of storage should make them resemble CG (resp. QN) methods. In any case, it seems reasonable to expect an increase in performance if more storage is used.

Among the papers dealing with variable storage methods, let us mention the works by Buckley (1978), Nocedal (1980), Buckley and Lenir (1983). The papers by Shanno (1978), Shanno and Phua (1978b) and Gill and Murray (1979) have also some connection to the subject. We shall come back in details on the methods proposed in these papers.

This study is definitely experimental in the sense that we have tried to bring improvements to an existing method by observing the effect of the implementation of some ideas on a set of test problems.

The resulting algorithms of this study have been included in the Frensh library Modulopt of INRIA under the names of m1qn2 and m1qn3. This library has the important attraction to put a battery of test problems to the disposal of optimization code writers and this, for each of the four classes of optimization problems: without constraints, with bound constraints, with linear constraints and with nonlinear constraints. These applications usually come from real-world models. This has not only

advantages. Indeed, the solutions of the problems are not exactly known, neither is the nature of the spectrum of the Hessian around a solution, the gradient of the objective function may be spoilt by rounding errors due to the large amount of computation and, last but not least, computation time and memory storage may be deterrent factors. However, those problems are the one to be solved and their large scale is not artificially obtained. Moreover, the problems from Modulopt library are written in a fully portable form (Fortran 77), which should allow comparison with further developped codes. The problems we used essentially come from physics: fluid mechanics, meteorology and crystallography.

We started with an algorithm proposed by Nocedal (1980) in order to compare its efficiency with the algorithm conmin of Shanno and Phua (1980) and the algorithm of Buckley and Lenir (1983), called mlgc3 in Modulopt library. Strictly speaking, conmin has not the "variable" storage property but, as we shall see, is rather a "two-storage" method. However, the code is intended for large-scale problems and, like variable storage methods, it is inspired by QN methods to enhance the performances of CG methods. Furthermore, the comparison of the performances of mlgc3 and Nocedal's algorithm with those of conmin will allow us to see to what extent the decreasing of the objective function can benefit from the availability of memory space.

Contrary to conmin and mlgc3, Nocedal's method has, in our opinion, the conceptually nice feature of not requiring any restart during computation, a concept inherited from CG methods. This makes the algorithm closer to QN methods. Our numerical experiments will show that, at least on our test problems, the "basic" Nocedal's method (called mlqn2.a below) generally behaves better than conmin and mlgc3.

The codes mlqn2 and mlqn3 that we validate by this study are polished versions of Nocedal's algorithm. Like this one, it builds the current approximation of the Hessian by updating m times a given matrix H^0, using the last m couples (y,s), where s is the change in the variable x and y is the corresponding change in the gradient of the objective function. A particular attention is given here in the choice of the starting matrix H^0. In mlqn2, H_0 is just a multiple of the identity matrix. In mlqn3, H^0 is a diagonal matrix, itself updated at each iteration using a diagonal update formula. We have found that the most efficient way to compute H^0 is to use a "diagonal bfgs update formula".

Without wishing to go into what will be said in the conclusion of the paper, we may say, however, that our experiments suggest that the marginal profit yielded by increasing the number m of updates is clearly decreasing with m and may become negative. If we take into account the computing time, the best algorithms for our test problems, which have dimensions n equal to 34, 403, 455, 500, 1559 and 1875, should

use between 5 and 10 updates. Increasing the number of updates does not increase significantly the performances or even decreases them and increases substantially the computing time. In other words, the algorithms do not seem to take much advantage of the possible avaibility of storage. On the other hand, a good choice of the starting (diagonal) matrix H^0 was determining for the performances of the methods. It is indeed not unusual to observe a better decrease of the objective function with m1qn2 amd m1qn3 than with bfgs algorithm.

The paper is organized as follows. In *Section 2*, we give more details on the algorithms mentioned above. In *Section 3*, we briefly describe the test problems and discuss numerical experiments made with m1gc3 and bfgs 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. Numerical experiments are related. In *Annex*, we show how to obtain some variable metric update formulae in Hilbert spaces by means of a variational formulation.

2. Some variable storage quasi-Newton methods

2.1. Notation

Let $I\!H$ be a Hilbert space over $I\!R$, equipped with a real scalar product $<\cdot,\cdot>$ and its associated norm $|.|$. We shall denote by $L(I\!H)$, the space of continuous linear operators on $I\!H$. Being given two vectors u and v in $I\!H$, we shall use the *bracket operator* $[u,v] \in L(I\!H)$ defined by $[u,v] : I\!H \longrightarrow I\!H : d \longrightarrow [u,v]\, d := <v,d>\, u$. We shall say that $B \in L(I\!H)$ is *positive* if $<Bu,u>$ is positive, for all nonzero $u \in I\!H$.

If $I\!H$ has finite dimension n and if $(e_i)_{1 \le i \le n}$ is an orthonormal basis (ONB) of $I\!H$, the Frobenius norm associated to the scalar product $<\cdot,\cdot>$ of a linear operator B is defined by

$$||B||_F := \left(\sum_{i=1}^{n} |B\, e_i|^2 \right)^{1/2} . \tag{2.1}$$

This quantity does not depend on the choice of the ONB and is actually a norm. If $||.||$ denotes the norm of $L(I\!H)$, we have clearly $||B|| \le ||B||_F$, $\forall\, B \in L(I\!H)$, and for the bracket operator, we have

$$|| [u,v] || = || [u,v] ||_F = |u|\ |v| . \tag{2.2}$$

2.2. Quasi-Newton methods in Hilbert spaces

We consider the problem of minimizing a smooth real-valued function f on $\mathbb{H}$:

$$\min \{ f(x) : x \in \mathbb{H} \} , \tag{2.3}$$

and we shall denote by x_* a local solution of this problem.

Quasi-Newton (or secant) methods generate two sequences: a sequence $(x_k) \subseteq \mathbb{H}$ of approximations of x_* and a sequence $(B_k) \subseteq L(\mathbb{H})$ of bijective approximations of $B_* := \nabla^2 f(x_*)$, the Hessian of f at x_*. We shall suppose that B_* is nonsingular and we shall note $H_* := B_*^{-1}$. Starting with a couple (x_0, B_0), the sequences are calculated by:

$$x_{k+1} := x_k - \rho_k \, B_k^{-1} \, g_k , \tag{2.4}$$

$$B_{k+1} := U (B_k , y_k , s_k) . \tag{2.5}$$

In (2.5), ρ_k is a positive step-size determined by a search on f along the direction $d_k := - B_k^{-1} g_k$ and $g_k := g(x_k)$, where $g(x) := \nabla f(x)$ is the gradient of f at x. In (2.5), 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:

$$H_{k+1} := \bar{U} (H_k , y_k , s_k) . \tag{2.6}$$

Let us make some comments to see to what extent QN theory depends on the scalar structure of $\mathbb{H}$. Formula (2.4) uses the gradient $g(x)$ of f at $x = x_k$, which is defined by Riesz theorem (see Weidmann (1980), for instance) as the unique representative in $\mathbb{H}$ of the continuous linear form $f'(x)$, the derivative of f at x:

$$f'(x)\cdot(u) = <g(x),u> , \ \forall \, u \in \mathbb{H} . \tag{2.7}$$

Formula (2.4) also uses the operator B_k, which is an approximation of the Hessian $B(x)$ of f at $x = x_*$. $B(x)$ is the unique representative in $L(\mathbb{H})$ of the continuous bilinear form $f''(x)$, the second order derivative of f at x:

$$f''(x)\cdot(u,v) = <B(x)u,v> , \ \forall \, (u,v) \in \mathbb{H}^2 . \tag{2.8}$$

Therefore, the choice of the scalar product in $\mathbb{H}$ affects the value of $B(x_*)$ that B_k should approach. So, B_k should also depend on the Hilbert structure of $\mathbb{H}$. Now, by Taylor's theorem and formulae (2.7) and (2.8), we have:

$$y_k = \int_0^1 B(x_k + t s_k) \, s_k \, dt .$$

This relation allows to understand why the basic idea of QN methods is to find update formulae U such that B_{k+1} given by (2.5) satisfies the *QN equation* (or *secant equation*):

$$y_k = B_{k+1}\, s_k \,. \tag{2.9}$$

Therefore, if a change of scalar product does not affect the secant equation (2.9), it changes B_{k+1} by changing y_k, via formula (2.7). Moreover, the form of the formulae U and $\bar{U}$ in (2.5) and (2.6) also depends on the scalar product. Indeed, the properties that characterize a given update formula are generally expressed in terms of the Hilbert structure of $I\!H$. If we take as a guideline the preservation of these properties, the form of the formulae will reflect its dependance on the scalar product. This point of vue is taken in *Annex*, where some classical rank one and rank two update formulae are derived.

One of them is the bfgs formula, which is thought to be one of the best secant updates in optimization (Dennis and Moré (1977)). For us, U will stand for the *bfgs formula*:

$$B_{k+1} := B_k + \frac{[y_k,y_k]}{<y_k,s_k>} - \frac{B_k\,[s_k,s_k]\,B_k}{<s_k,B_k s_k>} \tag{2.10}$$

and $\bar{U}$ will stand for the *inverse bfgs formula*:

$$H_{k+1} := H_k + \frac{[s_k - H_k y_k, s_k] + [s_k, s_k - H_k y_k]}{<y_k,s_k>} - \frac{<s_k - H_k y_k, y_k>}{<y_k,s_k>^2}\,[s_k,s_k] \,. \tag{2.11}$$

These formulae clearly show their dependance on the scalar product $<\cdot,\cdot>$.

Formulae (2.10) and (2.11) have the property to transmit the positivity of B_k to B_{k+1} (resp. of H_k to H_{k+1}), if and only if $<y_k,s_k>$ is positive. Having B_k positive is important to make d_k a descent direction of f at x_k: $f'(x_k)\cdot(d_k) < 0$. For this reason, the step-size ρ_k in (2.4) is generally determined such that Wolfe's (1969) conditions are satisfied:

$$f(x_k+\rho_k d_k) \le f(x_k) + \alpha_1\, \rho_k\, <g_k,d_k> \,, \tag{2.12}$$

$$<g(x_k+\rho_k d_k),d_k> \ge \alpha_2\, <g_k,d_k> \,, \tag{2.13}$$

where $0 < \alpha_1 < 1/2$ and $\alpha_1 < \alpha_2 < 1$. Clearly, inequality (2.13) implies the positivity of $<y_k,s_k>$.

In practice, $I\!H$ is a Hilbert space of finite dimension n. If the number n of variables is large, it may turn out to be impossible or too expensive to store in memory the full current approximation H_k of the inversed Hessian. Because the initial matrix H_0 takes generally little place in memory (it is most commonly a multiple of the identity matrix) and because H_k is formed from H_0 and k couples $\{ (y_i,s_i) : 0 \leq i < k \}$, it can be thought of memorizing these elements instead of H_k and of computing $H_k\, g_k$ by an appropriate algorithm. Of course, when the number of iteration increases, these pieces of information become more and more cumbersome in memory and we must think to truncate the sequence of couples $\{ (y_i,s_i) : 0 \leq i < k \}$ or to get rid of some of them. We shall say that it is 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 methods, the inverse update formula (2.11) is preferably used to the direct update formula (2.10) because the inversion of B_k may be problematic.

The variable storage QN methods we present hereafter, all 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.

2.3. The algorithm of Shanno

Motivated by the search of a conjugate gradient type method without exact line searches, Shanno (1978) recommended, on the basis of a large amount of computational results, to take $d_k := -\, H_k\, g_k$ as descent direction at iteration k, with the following formulae for H_k:

$$H_{r_k} := \bar{U}\,(\,\delta'_{r_k-1}\, I\,,\, y_{r_k-1}\,,\, s_{r_k-1}\,)\,,\ \text{for}\ \ k = r_k\,, \tag{2.14}$$

$$H_k := \bar{U}\,(\,H_{r_k}\,,\, y_{k-1}\,,\, s_{k-1}\,)\,,\ \text{for}\ \ k > r_k\,, \tag{2.15}$$

where $\bar{U}$ stands for the inverse bfgs formula (2.11), r_k is the index of the last restart iteration preceding iteration k and δ'_{r_k-1} is the evaluation at iteration r_k-1 of

$$\delta' := \frac{<y,s>}{|y|^2}\,. \tag{2.16}$$

The algorithm is restarted at iteration k when *Powell's (1977) restart criterion* is satisfied, i.e. when $|<g_k,g_{k-1}>| \geq 0.2\ |g_k|^2$. Then, r_k is set to k and formula (2.14) is used. Otherwise, r_k is set to r_{k-1} and formula (2.15) is used. The scaling factor (2.16) was experimented by Shanno and Phua (1978a) who motivated it by the self-scaling variable metric algorithms 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.

Formulae (2.14) and (2.15) are directly inspired by Beale's (1972) formulae to restart the CG method at iteration r_k. It can be proved indeed (see Shanno (1978)) that for f quadratic and exact line searches, the search directions obtained by (2.14) and (2.15) 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 then to generate automatically descent direction of f without requiring exact line searches, as long as $<y_k,s_k>$ is positive at each iteration, which can be provided by the line search.

This algorithm is a part of the code conmin, name by which we shall refer to. It requires $7n$ locations in memory: for x_k, g_k, d_k, x_{k+1}, g_{k+1}, y_{r_k} and s_{r_k}. The Euclidean scalar product is used: $<u,v> := u^T v$. The step-size ρ_k is determined to satisfy (2.12) with $\alpha_1 = 10^{-4}$ and

$$|<g(x_k+\rho_k\, d_k),d_k>| < \alpha_2\, |<g_k,d_k>| \,, \tag{2.17}$$

with $\alpha_2 = 0.9$, which is more restrictive than Wolfe's condition (2.13). At restart iterations, the first trial step-size is $\rho^0_{r_k} := 1$, whereas for nonrestart iterations, the first trial step-size is chosen to be (see Shanno and Phua (1980)):

$$\rho^0_k := \rho_{k-1}\, \frac{<g_{k-1},d_{k-1}>}{<g_k,d_k>} \,. \tag{2.18}$$

It is also important to mention that always at least two trial step-sizes are required before accepting a step-size satisfying (2.12) and (2.17). Therefore, at least one quadratic interpolation can be done at each step, which gives an exact line search in case f is quadratic.

2.4. The algorithm of Buckley and Lenir

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 would be possibly available in memory. The algorithm of Buckley and Lenir (1983) remedies to 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 occurs at iteration $r_k \leq k$. Let m be an integer with $m \geq 2$. If $k = r_k$, the algorithm takes:

$$H_{r_k} := \bar{U} \left(\delta'_{r_k-1} I , y_{r_k-1} , s_{r_k-1} \right) , \tag{2.19}$$

with δ'_{r_k-1} evaluated by (2.16). If $r_k < k \leq r_k+m-1$, the algorithm is in the *QN-part of the cycle* and takes:

$$H_k := \bar{U} \left(H_{k-1} , y_{k-1} , s_{k-1} \right) . \tag{2.20}$$

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

$$H_k := \bar{U} \left(H_{r_k+m-2} , y_{k-1} , s_{k-1} \right) . \tag{2.21}$$

In any case, the descent direction is $d_k := - H_k g_k$ at iteration k. The CG-part is so called because, if the line search is exact, $<g_k,s_{k-1}> = 0$ 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 $\{ (y_i,s_i) : r_k-1 \leq i \leq k-1 \}$ and for $k \geq r_k+m$, it uses the m couples $\{ (y_i,s_i) : r_k-1 \leq i \leq r_k+m-3 \} \bigcup \{ (y_{k-1},s_{k-1}) \}$. We also see that for $m=2$, the matrices H_k are computed just like in Sanno's algorithm.

The paper by Buckley and Lenir (1983) is not very specific about the way the step-size is determined. However, it is said that in the QN-part the unit step-size is tried first (because it is usually accepted by (2.12) and (2.13) for QN steps) and that the positivity of $<y,s>$ is assured (to keep the positive definiteness of the metrics). In the CG-part, the same strategy is followed (the positivity of $<y,s>$ is still necessary because of the use of (2.21)), except that at least two step-sizes are tried in order to take advantage of at least one interpolation. The initial trial step-size ρ_k^0 is given by (2.18).

This algorithm we have just described is called vscg by Buckley and Lenir (1980). The tests we present below have been made with a version of this algorithm, called mlgc3 in Modulopt library, on which we can be more specific. It requires $4n+2m(n+1)$ locations in memory: 4 vectors (x_k, g_k, d_k and an auxiliary vector) and for each update, 2 vectors (y and s) and 2 scalars ($|y|^2$ and $<y,s>$). The line search gives a step-size ρ_k satisfying (2.12) with $\alpha_1 = 0.001$ and (2.13) with $\alpha_2 = 0.9$ in the QN-part and $\alpha_2 = 0.001$ in the CG-part. At the first iteration, the initial trial step-size is given by

$$\rho_0^0 := \frac{2\,\Delta_0}{|g_0|^2}\,, \tag{2.22}$$

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.

2.5. The algorithm of Nocedal

The method proposed by Nocedal (1980) deserts the restart notion that the preceding algorithms inherited from the CG method and, as a result, is not cyclic anymore. 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 contained in the matrix is discarded and a new one is taken. An elegant procedure to apply H_k, which is not stored like a matrix but just represented by the m couples (y,s), to a vector is given. This procedure is based on the use of the following form of the inverse bfgs formula:

$$H_{k+1} := \left[I - \frac{[s_k,y_k]}{<y_k,s_k>}\right] H_k \left[I - \frac{[y_k,s_k]}{<y_k,s_k>}\right] + \frac{[s_k,s_k]}{<y_k,s_k>}\,. \tag{2.23}$$

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:

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

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

$$H_k := \bar{U}_{k-m}^{k-1}\,(\,H_k^0\,)\,, \tag{2.24}$$

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

Formula (2.24) clearly outlines the fact that a choice has to be done on the starting matrices H_k^0. This will be the subject of section 4, where we shall propose and test several choices.

2.6. The bfgs algorithm

In the tests below, we shall name bfgs, the following algorithm. If n is smaller than 501, it is just the classical bfgs method using $H_0 := I$, $H_1 := <y_0,s_0>/|y_0|^2\, I$ 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.

3. Test problems

The test problems we briefly describe below are taken from Modulopt library and we shall refer to them by their library name. Most of them come from real-live applications (physics, biology, ...) and their difficulty to be solved efficiently is certainly a challenge for optimization codes. Those problems are written in a fully portable way, which should allow, we hope so, new developed codes to confront them.

Some of the test problems can be modulated in order to change their conditionement or their number of variables. We shall refer to them by version numbers. Different versions can also correspond to different starting points x_0.

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 of this is that contrary to g, f decreases monotonically. We avoid in this way hazardeous realization of the stopping criterion. The value of f to reach 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. Tests have been made on a VAX-11/780.

The code **u1ts0** computes a trans-sonic flow. The number of variables is $n = 403$. The Fortran real variable $u0 \in]0,1[$ may be used to modulate the conditionement of the problem, which deteriorates when $u0$ increases. **u1ts0.1**: $u0 = 0.8$, $f(x_0) \approx 0.152\ 10^{-3}$, $\delta_0' \approx 3.70$ (see formula 2.18) and $f_{stop} = 10^{-12}$. **u1ts0.2**: $u0 = 0.9$, $f(x_0) \approx 0.257\ 10^{-3}$, $\delta_0' \approx 3.57$ and $f_{stop} = 10^{-8}$. **u1ts0.3**: $u0 = 0.95$, $f(x_0) \approx 0.322\ 10^{-3}$, $\delta_0' \approx 3.34$ and $f_{stop} = 10^{-5}$. For each version, $\Delta_0 = f(x_0)$ (see formula (2.22)).

The problem **u1mt1** comes from meteorology. The objective function is an augmented Lagragian but the multipliers are set to zero and they are not updated. The number of variables is $n = 1875$, $f(x_0) \approx 0.124\ 10^{8}$, $\delta_0' \approx 0.119\ 10^{-3}$, $\Delta_0 = f(x_0)$ and

$f_{stop} = 0.62\ 10^5$.

The problem **u1cr1** comes from crystallography: the so-called phase problem in X-ray crystallography. It consists in minimizing the opposite of the entropy of a molecular configuration when some Fourier coefficients of the unknown function (the electronic density) are supposed measured. These measurements appears as constraints. Using duality techniques, the problem is reduced to an unconstrained one in terms of the multipliers associated to the previous constraints. It is this problem that u1cr1 solves. Thus, the number of variables depends on the number of measurements taken into account. This data of the problem is controlled by the Fortran variables $drmax1 = drmax2$ and $drmin1 = drmin2$. In any version, $\Delta_0 = ||g_0||_2^2/(2||g_0||_\infty)$, where $||.||_2$ and $||.||_\infty$ are the l_2 and l_∞ norms, respectively. **U1cr1.1**: $n = 34$ (obtained with $drmax2 = 5.1$ and $drmin2 = 3.0$), $f(x_0) \approx -0.198\ 10^{-1}$, $\delta_0' \approx 0.105\ 10^4$ and $f_{stop} = -0.20033\ 10^{-1}$. **U1cr1.2**: $n = 455$ (obtained with $drmax2 = 3.1$ and $drmin2 = 1.3$), $f(x_0) \approx -0.882\ 10^{-2}$, $\delta_0' \approx 0.298\ 10^5$ and $f_{stop} = -0.8876\ 10^{-2}$. **U1cr1.3**: $n = 1559$ (obtained with $drmax2 = 20.0$ and $drmin2 = 0.8$), $f(x_0) \approx -0.106\ 10^{-1}$, $\delta_0' \approx 0.139\ 10^5$ and $f_{stop} = -0.10625\ 10^{-1}$.

The objective function of the code **vpbi** is quadratic:

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} i\ (\ x_{(i)} - 1\ )^2\ ,$$

where $x_{(i)}$ is the i-th component of x. Hence, B_* is a diagonal matrix formed with the first n integers. There are two versions of the code, corresponding to two different starting points. In each of them, $n = 500$, $\Delta_0 = f(x_0)/10$ and $f_{stop} = 10^{-5}$. **Vpbi.1**: $x_0 = 0$, hence $f(x_0) = 62625$ and $\delta_0' \approx 0.250\ 10^{-2}$. **Vpbi.2**: $x_{0(i)} = 1 + (100/i)^4$, hence $f(x_0) \approx 0.504\ 10^{16}$ and $\delta_0' \approx 0.958$.

The objective function of the code **vphi** is quadratic:

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} \frac{1}{i}\ (\ x_{(i)} - 1\ )^2\ .$$

Hence, H_* is a diagonal matrix formed with the first n integers. There are also two versions of the code, corresponding to two different starting points. In each of them, $n = 500$, $\Delta_0 = f(x_0)/10$ and $f_{stop} = 10^{-10}$. **Vphi.1**: $x_0 = 0$, hence $f(x_0) \approx 3.40$ and $\delta_0' \approx 1.11$. **Vphi.2**: $x_{0(i)} = 1 + (i/100)^4$, hence $f(x_0) \approx 0.246\ 10^5$ and $\delta_0' \approx 417$.

We can already give numerical results obtained with the codes described in Section 2. They are gathered in Table 1. For each test problem, we have put the dimension n of the problem in brackets. Results obtained with the code conmin are not given because the principle of the method is the same as the one of m1gc3 with $m = 2$.

Differences may only come from the line-search procedures and from adjustment of some parameters.

	m = 2	m = 5	m = 10	m = 20	m = 50	bfgs
ults0.1 (403)	162/82	157/81	151/81	146/82	78/63	66/64
ults0.2 (403)	330/168	227/147	179/144	249/180	156/140	142/134
ults0.3 (403)	181/93	165/111	131/105	117/102	100/92	95/91
ulmt1 (1875)	277/142	273/142	251/149	228/155	206/166	192/190 (**)
ulcr1.1 (34)	174/87	125/78	(***)	(***)	(***)	118/114
ulcr1.2 (455)	85/43	76/42	67/44	65/47	52/48	52/48
ulcr1.3 (1559)	46/24	49/33	31/23	35/28	31/29	31/29 (**)
vpbi.1 (500)	162/81	155/81	148/83	135/87	109/102	116/114
vpbi.2 (500)	253/126	307/158	290/153	302/188	295/195	515/257
vphi.1 (500)	158/81	139/83	125/82	108/86	124/117	190/188
vphi.2 (500)	148/74	138/69	112/56	100/50	94/47	94/47

Table 1 - *Performances (simul/iter) of m1gc3 and bfgs (**) by m1gc3, (***) enough storage for bfgs.*

These results enable us to recover some of the conclusions of Buckley and Lenir (1983): (1) there is a reasonable trend for the number of function evaluations simul to decrease as m increases but (2) this rule may be invalidated in some cases (see ults0.2 [$m = 20$], ulcr1.3 [$m = 5$, $m = 50$], vpbi.2 and vphi.1 [$m = 50$]). As in their test problems, we observe that (3) bfgs method has not always the best performances (see vpbi.1, vpbi.2 and vphi.1) and that (4) the number of iterations iter has rather a trend to increase with m.

However, these results do not enable us to infer on the improvement of performances that a variable storage QN method can expect by increasing m. Indeed, conclusion (1) is mainly due to the difference in the line-search options during the CG-part (at least two function evaluations are required per iteration) and the QN-part (the first trial step-size may be and is often 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 several iterations) and therefore, the ratio simul/iter decreases as m increases. Hence, even when iter increases slightly, simul decreases.

Some other interpretations can also be done on the results obtained on vpbi and vphi. First, let us remark that iter is larger for vpbi.2 than for vpbi.1 since the starting point is more distant from the optimal point in the second version. The converse is true for vphi. The results obtained by bfgs method on these problems bring out the

importance of the choice of the starting matrix. A close examination of the output shows, indeed, that the matrix H_k and, in particular, its diagonal change very slowly. This has the following consequences, which may be observed in Table 1. As the starting matrix is $H_1 = \delta_0' I$ and as δ_0' depends on the starting point, this one influences the results and notably the ratio simul/iter. In vpbi.1, most of the diagonal elements of H_* (the small one) are initially not too badly approximated by $\delta_0' \approx 0.0025$. Therefore, d_k is well scaled and the unit step-size is usually accepted: simul/iter ≈ 1. For vphi.1, the diagonal element of H_* are underestimated by $\delta_0' \approx 1.11$. Therefore d_1, and also d_k because H_k changes slowly, is too small in comparison with the optimal step. However, due to the tolerance of the criterion (2.13) with $\alpha_2 = 0.9$, the unit step-size is also generally accepted: simul/iter ≈ 1. On the other hand, for vpbi.2, the diagonal elements of H_* are noticeably overestimated by $\delta_0' \approx 0.954$. Therefore d_k is usually too large and the unit step-size is not accepted by the line-search procedure. As the objective function is quadratic and as a cubic interpolation is used in the procedure, the second step-size is usually accepted and we have simul/iter ≈ 2. The same remarks apply for vphi.2.

The reasons why iter may increase or decrease with m in vphi seem also largely due to the starting matrix and the line-search procedure. We may reason on vphi.1, as follows. When m increases, the optimizer is more and more often in its QN-part. Anyway, during the first m iterations, there is the same problem as before: d_k is too small in comparison with the optimal step, but the unit step-size is accepted. This means that d_k is not a very good step during this part of the minimization and the dicrease of f is bad. On the other hand, the step-sizes of vphi.2 are usually optimal because they are obtained by interpolation, which makes each iteration more efficient for a quadratic function. The results in Table 2 seem to confirm this analyse: we have forced the line-search procedure to try at least two step-sizes at each iteration when m1gc3 was running on vphi.1. We see that the number of iterations iter has a trend to decrease as m increases, at least for small m. The increase of iter for larger m may be due to the fact that for vphi.1, the second step-size is obtained by extrapolation during the QN-part, which seems not to give always the optimal step-size.

	m = 2	m = 5	m = 10	m = 20	m = 50	bfgs
vphi.1 (500)	166/83	135/68	117/58	117/59	141/71	155/77

Table 2 - *Performances (simul/iter) of m1gc3 and bfgs on vphi.1 when at least two step-sizes are tried per iteration.*

We see that the interpretation of results obtained by m1gc3 are sometimes difficult and complex, because of the various parameters shaping the algorithm.

4. The choice of a starting matrix

In this section, we address the problem of choosing a starting matrix H_k^0 for algorithm (2.24). This choice is particularly important in this method because it has to be done at each iteration.

All the optimization codes (versions of m1qn2 and m1qn3) tested in this section only differ by this choice of H_k^0 and have the same characteristics. We shall successively take and test H_k^0 as a multiple of the identity matrix and as a diagonal matrix. The line-search procedure is mlis0 from Modulopt, which is described in Lemaréchal (1981). At the first iteration, the first step-size to be tried, ρ_0^0, is given by (2.22) and for $k \geq 1$, $\rho_k^0 = 1$.

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

4.1. Scaling the identity

Because it is usually impossible to satisfy the QN equation, $Hy = s$, with an H of the form δI, it may be thought, a priori, to satisfy it in some direction v. Projecting the QN equation in a direction v such that $<y,v> \neq 0$ gives for δ, the value

$$\delta_v := \frac{<s,v>}{<y,v>} .$$

If v belongs to $V := \{ \alpha y + \beta s : \alpha \geq 0, \beta \geq 0, \alpha\beta \neq 0 \}$, $<y,v>$ and $<s,v>$ are positive and by Cauchy-Schwarz inequality, we have

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

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 $\{\alpha y : \alpha > 0\}$ and $\{\beta s : \beta > 0\}$, respectively.

The value δ' is used as scaling factor in Shanno's and Buckley and Lenir's methods: see (2.16). Here are some of its properties.

(P_1'): *If f is quadratic, δ' is the Rayleigh coefficient of H_* in the direction y, i.e.* $<H_*y,y>/|y|^2$. This property is usefull since $\delta' I$ has to approximate H_*.

(P_2'): *δ' minimizes in $\delta \in \mathbb{R}$, the norm* $|\delta y - s|$. Therefore, $\delta' I$ is the least square solution of the QN equation $Hy = s$, for H multiple of the identity.

(P_3'): *δ' minimizes in $\delta \in [\, \delta' , \delta'' \,]$, the condition number of $\bar{U}(\delta I, y, s)$, the inverse bfgs update.* Therefore, starting with $\delta' I$ may be a wise choice. A more general result, which is due to Oren and Spedicato (1976), claims 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.

(P_4'): *δ' is the unique solution of the following problem:*

$$\min_{\delta \in \mathbb{R}} \; \min_{H \in Q(s,y)} \; || \, \delta I - H \, ||_F \,, \tag{4.2}$$

where $Q(s,y) := \{ H \in L(\mathbb{R}^n) : Hy = s \}$. This means that $\delta' I$ is the multiple of the identity, the closest to $Q(s,y)$ for the Frobenius norm associated to the scalar product $<\cdot,\cdot>$ (see formula (2.1)). To show this, let us remark that problem (4.2) is equivalent to

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

where $H_\delta := \delta I + [s - \delta y, y]/|y|^2$ $(y \neq 0)$ is the inverse Broyden's update of δI (see problem (A.3) and formula (A.5) in annex). But, $||\delta I - H_\delta||_F = |||[s-\delta y, y]|||_F/|y|^2 = |s-\delta y|/|y|$ by (2.2). Hence property (P_4') from property (P_2'). □

(P_5'): *δ' is the unique solution of the following problem:*

$$\min_{\delta > 0} \; \min_{K \in S(s,y)} \; || \, \delta^{1/2} I - K \, ||_F \,, \tag{4.3}$$

where $S(s,y) := \{ K \in L(\mathbb{R}^n) : K^*Ky = s \}$. Note that $S(s,y) \neq \emptyset$ because $<y,s>$ is positive (see Proposition A.3 in annex). To prove (P_5'), let us remark that problem (4.3) is equivalent to

$$\min_{\delta > 0} \; || \, \delta^{1/2} I - K_\delta \, ||_F \,, \tag{4.4}$$

where, $K_\delta := \delta^{1/2} I \pm [y,s]/(|y|<y,s>^{1/2}) - \delta^{1/2} [y,y]/|y|^2$ (the reasoning is the same as in Dennis and Schnabel (1981), but for a general scalar product). Now, using an orthonormal basis $(e_i)_{i=1}^n$ of $\mathbb{R}^n$ for the scalar product $<\cdot,\cdot>$, we get:

$$|| \delta^{1/2} I - K_\delta ||_F^2 = \sum_{i=1}^{n} \left(\frac{<s,e_i> y}{|y| <y,s>^{1/2}} - \frac{\delta^{1/2} <y,e_i> y}{|y|^2} \right)^2$$

$$= \frac{|s|^2}{<y,s>} - \frac{2 \delta^{1/2} <y,s>^{1/2}}{|y|} + \delta .$$

Therefore, the minimum in (4.4) is obtained for $\delta^{1/2} = <y,s>^{1/2}/|y|$, i.e. $\delta = \delta'$. □

All these properties seem indicate that $\delta' I$ is the good starting matrix. However, by exchanging y and s, we obtain similar properties for δ''.

(P_1''): *If f is quadratic, $1/\delta''$ is the Rayleigh coefficient of B_* in the direction s, i.e.* $<B_* s,s>/|s|^2$. This property is also usefull since $1/\delta'' I$ has to approximate B_*.

(P_2''): *δ'' minimizes in $\delta \in \mathbb{R}$, the norm* $|y - s/\delta|$. Therefore, $1/\delta'' I$ is the least square solution of the QN equation $y = Bs$, for B multiple of the identity.

(P_3''): *δ'' minimizes in $\delta \in [\, \delta' , \delta'' \,]$, the condition number of $U(\delta I , s , y)$, the inverse dfp update.* This is obtained by taking $\theta = 0$ in the result of Oren and Spedicato mentioned after Property (P_3').

(P_4''): *δ'' is the unique solution of the following problem:*

$$\min_{\delta \in \mathbb{R}} \min_{B \in Q(y,s)} || 1/\delta I - B ||_F . \tag{4.5}$$

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

(P_5''): *δ'' is the unique solution of the following problem:*

$$\min_{\delta > 0} \min_{C \in S(y,s)} || 1/\delta^{1/2} I - C ||_F . \tag{4.6}$$

Because bfgs update is used in algorithm (2.24), the only property among them given above that could orientate the choice is property (P_3'), which is in favour of 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 ($H_k^0 = \delta'_{k-1}\, I$), m1qn2.b ($H_k^0 = \delta'_0\, I$ for $k \leq m$ and $H_k^0 = \delta'_{k-m}\, I$ for $k > m$) and m1qn2.c ($H_k^0 = \delta''_{k-1}\, I$) are given one above the other.

	m = 1	**m = 2**	**m = 5**	**m = 10**	**m = 20**	**m = 50**
ults0.1 (403)	145/132 145/132 179/95	117/106 121/110 171/95	97/93 97/87 160/84	76/71 86/74 135/70	76/72 78/71 127/67	70/68 68/67 118/62
ults0.2 (403)	293/254 293/254 727/348	270/235 309/268 613/277	157/143 169/148 440/178	156/142 158/141 418/154	161/142 158/142 379/160	149/140 151/135 353/155
ults0.3 (403)	112/101 112/101 252/127	112/105 124/113 248/115	100/95 110/97 197/89	104/96 108/100 188/91	102/93 104/94 183/88	102/90 98/92 167/82
ulmt1 (1875)	208/177 208/177 387/181	203/181 225/201 399/187	169/162 178/162 369/153	162/156 188/154 367/149	161/156 197/161 362/149	164/155 193/168 364/143
ulcr1.1 (34)	141/119 141/119 (*)	121/99 (*) 186/91	97/87 93/86 157/75	84/79 86/77 130/61	68/64 82/75 93/50	50/49 91/90 57/37
ulcr1.2 (455)	(*) (*) 87/54	58/52 60/52 85/46	53/49 53/49 87/47	49/46 55/46 85/45	48/44 50/45 76/41	47/44 50/49 70/38
ulcr1.3 (1559)	(*) (*) 46/23	32/27 31/27 46/23	30/26 27/24 49/23	25/23 26/24 40/20	23/21 29/27 32/16	23/21 30/29 32/16
vpbi.1 (500)	118/104 118/104 163/86	113/103 111/100 156/81	95/90 99/88 155/82	97/89 110/90 152/82	93/89 106/91 152/82	93/89 107/98 150/81
vpbi.2 (500)	327/279 327/279 373/187	263/236 263/227 369/184	235/225 248/209 393/196	204/194 265/211 373/186	199/185 272/194 353/176	170/160 285/178 295/147
vphi.1 (500)	179/153 179/153 248/139	162/138 174/144 229/128	131/123 138/122 183/101	111/100 121/107 147/80	87/80 105/99 114/64	64/61 126/122 78/45
vphi.2 (500)	153/138 153/138 193/106	141/125 151/128 182/97	117/106 126/103 169/95	102/92 117/87 144/80	87/78 117/72 109/65	66/59 110/48 74/49

Table 3 - *Performances (simul/iter) of m1qn2.a, m1qn2.b and m1qn2.c*
(*) fails to reach f_{stop}.

These results show that when m1qn2.a does not fail to reach f_{stop} (i.e. apart from ulcr1.2 [$m = 1$] and ulcr1.3 [$m = 1$]), 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 on 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 step-size. These interpolations may make each iteration more efficient and may decrease the number of iterations, like for the quadratic functions vpbi and vphi, but not enough to reduce the global cost of the runs, which is better measured by the number of simulations. On the other hand, the factor δ' gives a good scaling to the matrix as shown by a ratio simul/iter close to one.

Comparison between m1qn2.a and m1qn2.b shows that, in general, it is better to use more recent information, δ'_{k-1}, 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 sensible on vpbi.2 and vphi.2 where the phenomenon observed with m1gc3 crops up again: the matrix is initially overestimated and as no corrections are made by m1qn2.c during the first m iterations, the ratio simul/iter is close to two when m is large (for $m = 1$, m1qn2.a and m1qn2.b are the same optimizers).

Some remarks can still be made on the performances of m1qn2.a, which appears to be the best of the three optimizers. We observe that it has a general tendancy to improve with m but that, like for m1gc3, this rule is not absolute (see u1ts0.2 [$m = 20$], u1ts0.3 [$m = 10$, 20, 50], u1mt1 [$m = 50$] and vpbi.1 [$m = 10$]). Furthermore, this tendancy slows down for large m. As the computing time increases with m, the optimal number of updates for this optimizer should be placed between 5 and 10.

If we compare the performances of m1gc3 and m1qn2.a, we see that the number of function evaluations required by the latter is generally smaller, although the converse is generally true for the number of iterations, still an effect of the forced interpolation made at some iterations by the line-search procedure of m1gc3. If these results are rather in favour of m1qn2.a, this optimizer is, however, less robust and more sensitive to rounding errors than m1gc3 or bfgs method (see u1cr1). This may be due to the principle of the method itself. Using always local information to build the matrix, the search direction d_k may not be a descent direction if this local information is not very reliable, as this is often the case in the last iterations of a run. In practice, this defect could be avoided by a restart of the method in the opposite direction of the gradient.

4.2. Diagonal scaling

In this subsection, we shall suppose that the user knows an orthonormal basis $(e_i)_{1\leq i\leq n}$ of $\mathbb{R}^n$ for the scalar product $<\cdot,\cdot>$ and that he can compute easily $<u,e_i>$ for any $u \in \mathbb{R}^n$. Probably, this limits the use of the formulae below to situations where the scalar product is simple.

We shall say that a matrix D is *diagonal*, understood for the scalar product $<\cdot,\cdot>$ and the orthonormal basis $(e_i)_{1\leq i\leq n}$, if $<De_i,e_j> = 0$ for $i \neq j$. We shall note $D^{(i)} := <De_i,e_i>$. The identity matrix is diagonal. A diagonal matrix D is self-adjoint and it is positive if and only if the elements $D^{(i)}$ are positive. Therefore, we can represent and update positive diagonal matrices D_k, just by storing and updating its n positive diagonal elements $D_k^{(i)}$, $1 \leq i \leq n$. We have the following representation formula:

$$D = \sum_{i=1}^{n} D^{(i)} [e_i,e_i] , \tag{4.7}$$

which shows that it would be very interesting to have for $(e_i)_{1\leq i\leq n}$ an orthogonal basis formed by the eigenvectors of H_*, a dream. We shall denote by $\tilde{D}$, the average of the diagonal elements:

$$\tilde{D} := \frac{1}{n} \sum_{i=1}^{n} D^{(i)} . \tag{4.8}$$

If $H \in L(\mathbb{R}^n)$, its *diagonal* is the n-uple $\{ H^{(i)} := <He_i,e_i> : 1 \leq i \leq n \}$. We introduce the operator *diag* on $L(\mathbb{R}^n)$, which is such that if $H \in L(\mathbb{R}^n)$, *diag* H is the diagonal matrix defined by $(diag\ H)^{(i)} = H^{(i)}$, for $1 \leq i \leq n$. If H is positive, so are the elements of its diagonal.

In this subsection, we shall take for H_k^0 in (2.24) a positive diagonal matrix, which we shall denote by D_k (or simply D). Its inverse is also diagonal and $(D^{-1})^{(i)} = 1/D^{(i)}$. The motivations to take for H_k^0 a positive diagonal matrix are the following. First, a diagonal matrix may contain more information than just a factor of the identity, which is a particular case of diagonal matrix. Secondly, it only takes n locations in memory, i.e. half the locations taken by a couple (y,s) for an additional update. In vue of the preceding numerical results, the marginal profit of such an update is generally poor and therefore, starting with a diagonal matrix could yield some improvements. Finally, a diagonal matrix may give a good approximation of Rayleigh's ellipsoid of H_*, which is defined by $Ra(H_*) := \{ u \in \mathbb{R}^n : u \neq 0 , <H_*u,u> = |u|^3 \}$ and constitutes a full description of H_*.

We have found that the most convenient and efficient way to obtain good diagonal matrices D_k is to generate them by some formula, using the last couple (y_{k-1},s_{k-1}) to update them, just in the way matrices are generated in classical QN methods. At the beginning, we take for D_1 the diagonal matrix $\delta_0' I$, using the factor δ_0' suggested by the previous numerical experiments. Next, D_k is obtained by some formula, say $D_k := V (D_{k-1} , y_{k-1} , s_{k-1})$ or, dropping the subscripts, $D_+ := V (D , y , s)$. Of course, H_k is still obtained by using formula (2.24) with $H_k^0 = D_k$. The algorithms we shall test, only differ by the choice of the formula V.

In mlqn3.a, formula V is obtained by diagonalizing the inverse bfgs formula (2.23): $D_+ := \mathit{diag}\ \bar{U} (D , y , s)$. We have:

$$D_+^{(i)} = D^{(i)} + \left[\frac{1}{<y,s>} + \frac{<Dy,y>}{<y,s>^2} \right] <s,e_i>^2 - \frac{2\, D^{(i)} <y,e_i> <s,e_i>}{<y,s>} . \quad (4.9)$$

In mlqn3.b, formula V is obtained by diagonalizing the direct bfgs formula (2.11): $D_+^{-1} := \mathit{diag}\ U (D^{-1} , y , s)$. We have:

$$D_+^{(i)} = \left[\frac{1}{D^{(i)}} + \frac{<y,e_i>^2}{<y,s>} - \frac{(<s,e_i>/D^{(i)})^2}{<D^{-1}s,s>} \right]^{-1} . \quad (4.10)$$

In mlqn3.c, formula V is obtained by diagonalizing the inverse dfp formula: $D_+ := \mathit{diag}\ U (D , s , y)$. We have:

$$D_+^{(i)} = D^{(i)} + \frac{<s,e_i>^2}{<y,s>} - \frac{(D^{(i)}<y,e_i>)^2}{<Dy,y>} . \quad (4.11)$$

As bfgs and dfp formulae transmit positivity if and only if $<y,s>$ is positive, this will also be a sufficient condition to have D_+ positive when D is positive.

In Table 4, results with mlqn3.a, mlqn3.b and mlqn3.c are given one above the other.

	m = 1	m = 2	m = 5	m = 10	m = 20	m = 50
ults0.1 (403)	355/142 86/85 117/81	527/186 73/72 105/79	168/91 67/66 78/63	170/83 60/59 71/61	94/58 56/55 61/54	80/51 55/54 52/50
ults0.2 (403)	831/226 (*) 183/176 292/174	1001/183 (*) 145/142 279/188	485/116 (*) 128/122 251/150	538/129 (*) 134/123 230/138	693/157 (*) 136/123 229/136	1001/250 (*) 134/120 209/132
ults0.3 (403)	465/162 99/96 143/116	238/126 89/87 113/100	226/103 90/85 95/82	181/96 89/85 105/86	167/92 92/85 101/81	154/87 92/84 93/79
ulmt1 (1875)	548/164 (*) 225/209 334/235	319/78 (*) 242/240 247/177	207/77 (*) 176/174 260/184	272/89 (*) 183/181 212/160	490/126 (*) 173/171 205/161	889/179 (*) 171/169 179/148
ulcr1.1 (34)	(*) 85/83 104/73 (*)	(*) 88/86 103/71	(*) 83/82 107/69	(*) 82/81 88/60	1001/73 (*) 80/79 75/54	111/57 78/77 60/49
ulcr1.2 (455)	34/33 52/49 34/33	40/39 51/50 40/39	37/36 46/45 37/36	37/36 48/47 37/36	36/35 45/44 (*) 37/36	36/35 47/46 37/36
ulcr1.3 (1559)	27/26 38/37 26/25	28/27 34/33 28/27	26/25 30/29 27/26	26/25 30/29 27/26	26/25 30/29 26/25	26/25 34/33 26/25
vpbi.1 (500)	37/36 71/69 37/35	39/35 53/51 41/37	38/34 54/52 39/35	37/33 55/53 39/35	37/33 54/52 38/34	36/32 53/51 38/34
vpbi.2 (500)	243/71 123/51 518/162	254/75 127/54 456/149	281/83 139/60 459/148	398/123 133/57 540/173	375/126 134/60 549/176	371/124 126/56 611/204
vphi.1 (500)	104/103 696/695 106/105	91/90 209/208 98/97	91/90 250/249 97/96	90/89 206/205 96/95	90/89 183/182 95/94	89/88 183/182 95/94
vphi.2 (500)	92/41 49/24 93/42	96/44 49/24 96/45	94/43 47/23 90/42	91/42 47/23 85/40	91/42 47/23 83/39	87/40 47/23 83/40

Table 4 - *Performances (simul/iter) of m1qn3.a, m1qn3.b and m1qn3.c*
(*) fails to reach f_{stop}.

These results enable us to make the following observations: (1) performances are very much depending on the formulae used to update the diagonal matrix (see test problem vpbi.2, for instance); but (2) it is not always the same formula that gives the best results (if m1qn3.b is the best optimizer for ults0, ulmt1, vpbi.2 and vphi.2, it is the worst one for ulcr1.2, ulcr1.3, vpbi.1 and vphi.1); however (3) for each test problem, there is generally one of the three optimizers that gives better results than m1qn2.a (the only exception is for ulmt1, but the number of simulations used by m1qn3.b is rather close to the one used by m1qn2.a), which shows that obtaining a good diagonal starting matrix for algorithm (2.24) may improve the results.

Despite its good results on ulcr1.2, ulcr1.3, vpbi.1 and vphi.1, optimizer mlqn3.a (i.e. formula (4.9)) should be discarded for the following reason. The right hand side of (4.9) updates D by using two correcting terms. The first one is positive (D and $<y,s>$ 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 $<y,e_i>$ and $<s,e_i>$ have no reason to have the same sign, the diagonal elements of D_k may have a trend to increase during the minimization. This trend can be observed when the number of iterations is sufficiently large like in ulmt1 where D blows up and the run stops on overflow: for $m = 50$, we have at the last iteration, $\tilde{D}_{179} \approx 0.108\ 10^4$, while $\delta'_{179} \approx 0.178\ 10^{-3}$. The large number of simulations needed during the line-search to reduce the step-size reflects this phenomenon. Of course when the function is quadratic with a positive diagonal Hessian, $<y,e_i> = H_*^{(i)} <s,e_i>$ and $<s,e_i>$ have the same sign and the last term in (4.9) is negative. In this case, the previous argument does not apply. Anyway, the blowing up of the diagonal D_k does not occur on vpbi and vphi.

Results obtained with mlqn3.b and mlqn3.c are more difficult to comment and we do not have any convincing argument to decide between them. A partial clarification is obtained, however, by observing that δ' is a *lower attractor* for formula (4.10), whereas δ'' is an *upper attractor* for formula (4.11). Precisely, we mean by this that, denoting by D_b and D_c the diagonal matrices obtained from D by (4.10) and (4.11) respectively, we have

$$D^{(i)} \geq \delta' \,, \forall\, i \implies (D_b^{-1})\tilde{} \geq (D^{-1})\tilde{}\,, \tag{4.12}$$

where $(D^{-1})\tilde{}$ is the average value of the diagonal elements of D^{-1}, see formula (4.8), and

$$D^{(i)} \leq \delta'' \,, \forall\, i \implies \tilde{D}_c \geq \tilde{D}\,. \tag{4.13}$$

To see (4.13) (it is similar for (4.12)), we remark that $\sum (D^{(i)}<y,e_i>)^2 \leq (\max D^{(i)})$ $<Dy,y>$. Hence, from (4.11), we have

$$\tilde{D}_c \geq \tilde{D} + \frac{1}{n}\left[\frac{|s|^2}{<y,s>} - \max_{1\leq i\leq n} D^{(i)}\right],$$

from which (4.13) follows.

As $\delta'' \geq \delta'$, implications (4.12) and (4.13) suggest that formula (4.11) may have a trend to make the elements of the diagonal preconditioner D bigger than those generated by formula (4.10). We observed, indeed, this trend, which does not appear directly in Table 4: formula (4.10) has some ability to decrease D when δ' decreases and formula (4.11) has some ability to increase D when δ'' increases.

Now, going back to Table 4, the previous remarks highlight the results obtained on vpbi and vphi, in particular. Problems vpbi.1 and vphi.1 have both an initially underestimated diagonal matrix D_1. Therefore, as D increases more rapidly with formula (4.11) than with formula (4.10), it seems now more comprehensible to see that mlqn3.c works better than mlqn3.b on these problems. For example, in vphi.1, $\tilde{D}_0 \approx 1.11$ and at iteration 50, we have $\tilde{D}_{50} \approx 1.21$ (although $\delta'_{50} \approx 6.35$) when mlqn3.b $[m = 50]$ is used and $\tilde{D}_{50} \approx 6.87$ ($\delta'_{50} \approx 27.2$ and $\delta''_{50} \approx 107.$) when mlqn3.c $[m = 50]$ is used. The inverse situation occurs on vpbi.2 and vphi.2 because the initial diagonal matrix is overestimated.

For other test problems, the analysis is not so simple. First, the problems are not quadratic anymore, hence δ'_k is not a Rayleigh coefficient of a same matrix. Secondly, if for each problem δ'_0 is rather less than the values of δ' met during minimization, this underestimation of D_1 is not so pronounced as in vpbi.1 or vphi.1. Therefore, various causes may be responsible of the observed results. If we look in details on the outputs, it could be thought that much could be explained by this finally rough argument on the magnitude of the preconditioner D and on its rapid or slow evolution when using (4.10) or (4.11). However, a more detailed observation of the results shows that a finer analysis, which we do not have, is necessary to understand the respective merits of mlqn3.b and mlqn3.c. For example, if we change the stopping criterion for ults0.1 and take $f_{stop} = 10^{-9}$, we obtain the following results.

	m = 1	**m = 2**	**m = 5**	**m = 10**	**m = 20**	**m = 50**
ults0.1 (403)	47/46 79/58	48/47 59/54	41/40 45/40	40/39 47/42	41/40 40/39	40/39 41/39

Table 5 - *Performances (simul/iter) of mlqn3.b and mlqn3.c for* $f_{stop} = 10^{-9}$.

Although D_k never becomes too large with mlqn3.c (simul/iter is closer to one, unlike in Table 4 for m small), its performances remain less good as those of mlqn3.b.

The previous analysis shows the important role played by the magnitude of D_k. The temptation is great, consequently, to scale D before updating it by (4.10) or (4.11). This idea is at the root of the optimizers mlqn3.b2 and mlqn3.c2 that are versions corresponding to mlqn3.b and mlqn3.c respectively. In both of them, before updating D, we multiply it by a factor σ such that σD has the good Rayleigh coefficient in the direction y, that is to say δ' (see property (P'_1)). We find for σ:

$$\sigma = \frac{<y,s>}{<Dy,y>} .$$

With this scaling factor, formulae (4.10) and (4.11) become respectively:

$$D_+^{(i)} = \left(\frac{<Dy,y>}{<y,s>D^{(i)}} + \frac{<y,e_i>^2}{<y,s>} - \frac{<Dy,y>(<s,e_i>/D^{(i)})^2}{<y,s><D^{-1}s,s>} \right)^{-1} \tag{4.14}$$

and

$$D_+^{(i)} = \frac{<y,s>D^{(i)}}{<Dy,y>} + \frac{<s,e_i>^2}{<y,s>} - \frac{<y,s>(D^{(i)}<y,e_i>)^2}{<Dy,y>^2} . \tag{4.15}$$

Results obtained with m1qn3.b2 (formula (4.14)) and m1qn3.c2 (formula (4.15)) are given one above the other in Table 6.

	m = 1	**m = 2**	**m = 5**	**m = 10**	**m = 20**	**m = 50**
ults0.1 (403)	76/71 205/163 (*)	84/76 124/116	70/65 151/127 (*)	65/60 80/77	59/56 69/65	58/55 59/56
ults0.2 (403)	179/167 581/505 (*)	150/138 630/555 (*)	130/118 763/705 (*)	137/123 607/558 (*)	131/116 752/677 (*)	125/116 434/408
ults0.3 (403)	95/90 194/182	93/83 209/194	89/81 151/148	90/79 141/133	87/79 126/118	87/77 120/113
ulmt1 (1875)	207/184 (*)	175/152 (*)	150/145 (*)	151/144 (*)	148/141 (*)	145/139 442/414
ulcr1.1 (34)	72/68 (*) 99/89 (*)	65/57 47/39 (*)	57/54 94/87 (*)	53/51 58/53 (*)	47/45 60/54	44/42 50/46
ulcr1.2 (455)	41/38 35/34	41/37 35/33	38/36 36/33	38/35 34/32	36/34 35/33	37/35 34/32
ulcr1.3 (1559)	22/19 19/18	21/18 20/19	19/18 18/17	18/17 18/17	18/17 18/17	18/17 18/17
vpbi.1 (500)	50/45 42/36	51/47 39/35	48/44 39/37	47/43 38/34	47/43 37/33	47/43 37/33
vpbi.2 (500)	76/69 62/56	77/68 65/56	74/66 58/49	72/65 55/47	71/64 58/48	68/60 53/45
vphi.1 (500)	46/44 47/46	53/51 53/50	50/47 47/46	48/45 49/46	48/46 47/45	47/45 45/43
vphi.2 (500)	51/47 52/49	55/49 55/49	48/44 51/47	48/44 47/43	48/43 48/43	47/43 47/42

Table 6 - *Performances (simul/iter) of m1qn3.b2 and m1qn3.c2*
(*) fails to reach f_{stop}.

We see that m1qn3.b2 generally works better than m1qn3.b: exceptions are for ults0 and vphi.2, but they are minor in comparison with the better performances obtained on ulmt1 and ulcr1. On the other hand, if m1qn3.c2 sometimes improves m1qn3.c, for instance on ulcr1.2, ulcr1.3, vpbi and vphi, it has great deficiencies on ults0, ulmt1 and ulcr1.1. Therefore, formula (4.15) should not be used. After all it

comes from the diagonalization of dfp formula, whereas it is updated in (2.24) by bfgs formula, which may not be very suitable.

Now, comparing the results obtained by m1qn3.b2 with the one obtained by the previous optimizers, we see that m1qn3.b2 works always better than m1qn2.a (Table 3) and m1gc3 (Table 1), if we compare performances for a same number m of bfgs updates and that it is better than bfgs method as soon as m is greater than or equal to $1 \cdots 10$, depending on the problem.

The optimizer m1qn3.b2 is the one we should recommend.

5. Conclusion

The main merit of algorithm m1qn3.b2, which is the one that gives the best results, is precisely its performances on the test problems we have.

This is not very much, because our conclusion could be invalidated on other problems but this is inherent in any inductive processes. On the other hand, it is clear that a finer mathematical analysis of diagonal update formulae is wellcome and it is not clear whether formula (4.14) would stand up such an analysis. However, our tests have enable us to make a first sort among possible formulae. For instance, it could have been thought, a priori, that formula (4.9) obtained by diagonalization of the inverse bfgs formula is as good as formula (4.10) 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 point of vue used to obtain matrix updates, but few works well.

On the other hand, this study shows the interest that the update of diagonal matrices may have in some cases. It seems clear that this is not a general conclusion and that the success of m1qn3.b2 could be due to the fact that the eigenvectors of H_* in our test problems are not too different from the basic vectors e_i, which form, in these problems, the canonical basis of $\mathbb{R}^n$. However, when such is the case, algorithm m1qn3.b2 may do good services.

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

In this annex, we show how to extend to Hilbert spaces, the variational derivation of some rank one and rank two quasi-Newton formulae. The fact that we shall work in an infinite dimension space is rather anecdotal but does not require any particular effort, so, we did not hold back from doing this generalization. More important is the fact that the obtained formulae are valid for any scalar product and not only (in case the dimension is finite) for the Euclidean scalar product. This allows the use of the theory in various real-live situations.

For the abstract tools used in this annex, we refer the reader to the book by Weidmann (1980).

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. If these authors obtained the formulae by selecting, in the family of perturbations of rank one or two, the one that gives the desired properties (QN property, symmetry, positivity), we shall adopt the more classical but more elegant variational point of vue, showing in this way that the formulae still give least-change updates.

The basic tool to implement this strategy is the Frobenius norm (2.1) associated to a scalar product of a Hilbert space $\mathbb{H}$, or in case $\mathbb{H}$ has infinite dimension, its generalization, the Hilbert-Schmidt norm. So, let $\mathbb{H}$ be an infinite dimensional Hilbert space over $\mathbb{R}$ with a real scalar product $<\cdot,\cdot>$ and its associated norm $|.|$. Let B be in $L(\mathbb{H})$, and let $(e_i)_{i\in I}$ be an ONB of $\mathbb{H}$, where the family of indices I is not necessarily countable. The generalization of (2.1) is

$$||B||_{HS} := \left[\sum_{i \in I} |B\ e_i|^2 \right]^{1/2} . \tag{A.1}$$

The quantity of the right hand side of (A.1) makes sense if at most countably many of the Be_i are different from zero and if the series is convergent. This is not satisfied for all operators B in $L(\mathbb{H})$ (take $B = I$, for example), but if it is, the quantity does not depend on the choice of the ONB and formula (A.1) defines a norm, called the *Hilbert-Schmidt norm* of B and the operator is called a *Hilbert-Schmidt operator*. Such operators form a Hilbert space for the norm (A.1). We shall denote this space by $L_2(\mathbb{H})$. If $||.||$ denotes the norm of $L(\mathbb{H})$, $||B|| \leq ||B||_{HS}$ for $B \in L_2(\mathbb{H})$. Let us still mention that if B_1 and B_2 are continuous operators on $\mathbb{H}$ and if one of them is a Hilbert-Schmidt operator then B_1B_2 is a Hilbert-Schmidt operator.

Obviously, finite rank operators are in $L_2(\mathbb{H})$, but the converse is generally false. On the other hand, Hilbert-Schmidt operators are compact. For example, the bracket operator $[u,v]$ introduced at the beginning of section 2 is in $L_2(\mathbb{H})$ and we still have

$$|| [u,v] ||_{HS} = || [u,v] || = |u|\ |v| . \tag{A.2}$$

Note that if $R \in L(\mathbb{H})$, we have $R\,[u,v] = [Ru,v]$ and $[u,v]\,R = [u,R^*v]$, where R^* is the adjoint of R.

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 QN equation (see (2.9)):

$$y = B_+ s \ . \tag{A.3}$$

We write $B_+ = B + P$ and we restrict the perturbation operator P to be a Hilbert-Schmidt operator so that the perturbation can be measured with the norm (A.1). Hence, P is supposed to belong to

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

Let R_1 and R_2 be bijective operators in $L(\mathbb{H})$. Then, we consider the following minimization problem:

$$\min_{P \in \Pi} \ ||\ R_1\ P\ R_2\ ||_{HS}\ . \tag{A.4}$$

Note that the norm in (A.4) makes sense since if $P \in L_2(\mathbb{H})$, so does $R_1 P R_2$. If we take $R_1 = R_2 = I$ (identity), we see that B_+ is simply the operator satisfying (A.3), the closest to B for the Hilbert-Schmidt norm. The next proposition 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.4) has a unique solution P_c, given by*

$$P_c := \frac{[y-Bs,c]}{<c,s>}\ ,$$

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

Proof. Let $P \in \Pi$ and set $E := R_1\ P\ R_2$ and $E_c := R_1\ P_c\ R_2$. We have to prove that $||E_c||_{HS} \leq ||E||_{HS}$. With $z := R_2^{-1}\ s = R_2^*\ c$ and $y - Bs = Ps$, we have

$$E_c = \frac{[Ez,z]}{|z|^2}\ .$$

Hence, using (2.2), we get

$$||E_c||_{HS} = \frac{|Ez|}{|z|} \leq ||E|| \leq ||E||_{HS} .$$

Unicity comes from the fact that the Hilbert-Schmidt norm, deriving from a scalar product, is a strictly convex norm (i.e., $E_1 \neq E_2$, $||E_1||_{HS} = ||E_2||_{HS} = \tau$ and $0 < \alpha < 1$ imply $||\alpha E_1 + (1-\alpha) E_2||_{HS} < \tau$), from the fact that Π is convex (affine) and from the bijectivity of R_1 and R_2. □

The solution of problem (A.4) with $R_2 = I$ gives *Broyden's update formula*:

$$B_{Broyden} := B + \frac{[y-Bs,s]}{|s|^2} . \tag{A.5}$$

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.3). This time, the perturbation operator P is supposed to belong to

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

For $R \in L(\mathbb{H})$, bijective, we consider the following minimization problem:

$$\min_{P \in \Pi_S} || R^* P R ||_{HS} . \tag{A.6}$$

The next proposition shows that this problem has a unique solution depending on R only through $c := R^{-*}R^{-1} s$.

Proposition A.2 (Dennis and Moré). *Let B be a self-adjoint continuous operator on a Hilbert space $\mathbb{H}$ and R be a bijective continuous operator on $\mathbb{H}$. Let y and s be two vectors in $\mathbb{H}$ with $s \neq 0$. Then, problem (A.5) has a unique solution P_c, given by*

$$P_c := \frac{[y-Bs,c] + [c,y-Bs]}{<c,s>} - \frac{<y-Bs,s>}{<c,s>^2} [c,c] , \tag{A.7}$$

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

Proof. It is a straightforward adaptation of the proof of Theorem 7.3 of Dennis and Moré (1977). Let $P \in \Pi_S$ and set $E := R^* P R$ and $E_c := R^* P_c R$. We have to prove that $||E_c||_{HS} \leq ||E||_{HS}$. With $z := R^{-1} s = R^* c$, we have

$$E_c = \frac{E [z,z] + [z,z] E}{|z|^2} - \frac{<E z,z>}{|z|^4} [z,z] .$$

Hence, $E_c\, z = E\, z$ and $|E_c\, z| = |E\, z|$. Let $e_i \in I\!H$ for $i \in I$, such that $\{z/|z|\} \cup (e_i)_{i\in I}$ forms an ONB of $I\!H$ (this is possible, see Weidmann (1980), Theorem 3.10). Since $e_i \perp z$, we have

$$E_c\, e_i = \frac{[z,z]}{|z|^2}\, E\, e_i\ .$$

Then using (2.2), we deduce from this that $|E_c\, e_i| \leq |E\, e_i|$ for $i \in I$ and therefore $||E_c||_{HS} \leq ||E||_{HS}$.

Unicity comes from the same argument as in Proposition A.1. □

If we take $R = I$ in problem (A.5), we obtain the *psb update formula*:

$$B_{psb} = B + \frac{[y-Bs,s] + [s,y-Bs]}{|s|^2} - \frac{<y-Bs,s>}{|s|^4}\,[s,s]\ .$$

The question of the existence of positive self-adjoint secant (i.e. verifying the secant equation (A.3)) operators is addressed in the following proposition.

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

(i) $\exists\, B \in L(I\!H)$, *self-adjoint, positive, such that* $y = B\, s$,

(ii) $\exists\, C \in L(I\!H)$, *bijective, such that* $y = C^*C\, s$,

(iii) $<y,s>$ *is positive* .

Proof. As $(i) \Longrightarrow (iii)$ and $(ii) \Longrightarrow (i)$ are clear, it remains to prove $(iii) \Longrightarrow (ii)$. So, suppose that $<y,s>$ is positive. The following Dennis and Schnabel's (1981) factor is suitable for (ii):

$$C := I + \frac{[s,y]}{|s|\ <y,s>^{1/2}} - \frac{[s,s]}{|s|^2}\ .$$

Indeed, $y = C^*C\, s$ and because C is a finite rank pertubation of the identity, it is bijective if it is injective (Fredholm's alternative, Weidmann (1980, Theorem 6.8)), which is true because $C\, z = 0$ implies $z = \alpha\, s$ with $\alpha \in I\!R$ and $\alpha\, C\, s = 0$ implies $\alpha = 0$, hence $z = 0$. □

Now, if $<y,s>$ is positive, if C is given by Proposition A.3 (ii) and if we take $R = C^{-1}$ in Proposition A.2, then $c = y$ and formula (A.7) becomes the *dfp update formula*:

$$B_{dfp} := B + \frac{[y-Bs,y] + [y,y-Bs]}{<y,s>} - \frac{<y-Bs,s>}{<y,s>^2}\,[y,y]\,. \tag{A.8}$$

If in (A.8), we exchange y and s on the one hand and if we change B by H on the other hand, we recover bfgs formula (2.12). The positivity of B_{dfp} (if B is self-adjoint, positive and if $<y,s>$ is positive) is easily verified using the following form of formula (A.8):

$$B_{dfp} := \left[I - \frac{[y,s]}{<y,s>}\right] B \left[I - \frac{[s,y]}{<y,s>}\right] + \frac{[y,y]}{<y,s>}\,. \tag{A.9}$$

References

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

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

A. Buckley, A. Lenir (1983). QN-like variable storage conjugate gradients. Mathematical Programming 27, 155-175.

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

J.E. Dennis, R.B. Schnabel (1981). A new derivation of symmetric positive definite secant updates. Nonlinear Programming 4, 167-199. Academic Press.

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

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

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

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

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

M.J.D. Powell (1976). Some global convergence properties of a variable metric

algorithm for minimization without exact line searches. *Nonlinear Programming.* R.W. Cottle, C.E. Lemke (eds). American Mathematical Society, Providence, R.I.

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

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

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

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

D.F. Shanno, K.-H. Phua (1980). Remark on algorithm 500: minimization of unconstrained multivariate functions. ACM Transactions on Mathematical Software 6/4, 618-622.

J. Weidmann (1980). *Linear operators in Hilbert spaces.* Graduate Texts in Mathematics 68. Springer-Verlag.

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