A limited memory algorithm for inequality constrained minimization

Paul ARMAND† and Philippe SÉGALAT†

October 2, 2003

Abstract: A method for solving inequality constrained minimization problems is described. The algorithm is based on a primal-dual interior point approach, with a line search globalization strategy. A quasi-Newton technique (BFGS) with limited memory storage is used to approximate the second derivatives of the functions. The method is especially intended for solving problems with a large number of variables with bound constraints and a medium number of general inequality constraints. Some numerical experiments are presented to validate our approach.

Key words: constrained optimization, interior point method, large scale optimization, limited memory method, quasi-Newton method

1 Introduction

We present a limited memory quasi-Newton algorithm for solving large scale nonlinear optimization problems of the form

$$\begin{cases} \min\ f(x), \\ c(x) \geq 0, \end{cases} \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are continuously differentiable functions. We assume that n, the number of variables, is large. A particular attention will be brought to the problems having a set of constraints defined by a function c of the form

$$c(x) = \begin{pmatrix} g(x) \\ x - l \\ u - x \end{pmatrix}, \tag{1.2}$$

where $g : \mathbb{R}^n \to \mathbb{R}^p$ is a nonlinear function with a medium number of components and where l and u are lower and upper bounds on the variables. It is assumed that the components of the lower and upper bounds satisfy $-\infty \leq l_{(i)} < u_{(i)} \leq +\infty$, for all $i = 1, \ldots, n$. The algorithm needs only to evaluate the function values and the first derivatives of f and c, the computation of the second derivatives is not required. The Hessian matrices of the functions are approximated by means of quasi-Newton techniques and only a limited amount of memory is required to store these approximation matrices.

The algorithm is an implementation of a BFGS interior point method proposed in [1] and uses a compact representation of limited memory quasi-Newton matrices

†LACO - CNRS UMR 6090 - Université de Limoges, Faculté des Sciences et Techniques, 123, avenue Albert Thomas, 87060 Limoges (France) ; e-mail: armand@unilim.fr, segalat@unilim.fr.

introduced in [2]. To store the second order information, only a small number, say r, of vector pairs of length n is kept in memory (in practice $3 \leq r \leq 20$). We will show that the storage memory requirement is of order $O(nr)+O(p(p+r))$ and the computational cost of one iteration is of order $O(n(p+r)^2)+O(p^3)+O(r^3)$.

The paper is organized as follows. In the next section, we present an outline of the algorithm and show how we use a compact representation of BFGS matrices in the framework of an interior method. In Section 3, we give the estimates of the amount of memory and of the computational cost for one iteration. Then, we give a convergence result of the algorithm in Section 4 and present some numerical tests in Section 5.

2 Outline of the algorithm

Let us denote by $\nabla f(x)$ the gradient of f at x, $\nabla c(x)^\top$ the Jacobian matrix of c at x, $\lambda \in \mathbb{R}^m$ a vector of multipliers, $C = \mathrm{diag}(c_{(1)}, \ldots, c_{(m)})$ the diagonal matrix, whose diagonal elements are the components of c and $e = (1 \ \cdots \ 1)^\top$ the vector of all ones. The Lagrangian associated with problem (1.1) is the function $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ defined by $\ell(x,\lambda) = f(x) - \lambda^\top c(x)$. The gradient and Hessian of ℓ with respect to x are given by

$$\nabla_x \ell(x,\lambda) = \nabla f(x) - \nabla c(x)\lambda \quad \text{and} \quad \nabla^2_{xx}\ell(x,\lambda) = \nabla^2 f(x) - \sum_{i=1}^{m} \lambda_i \nabla^2 c_{(i)}(x).$$

The primal-dual interior point algorithm computes iteratively approximate solutions of the perturbed optimality conditions associated to (1.1)

$$\left\{ \begin{array}{l} \nabla_x \ell(x,\lambda) = 0, \\ C(x)\lambda - \mu e = 0, \\ (c(x),\lambda) > 0, \end{array} \right. \tag{2.1}$$

for a sequence of parameters $\mu > 0$ converging to zero. The second equation means that

$$(\forall\, i = 1, \ldots, m) \quad c_{(i)}(x)\lambda_{(i)} = \mu.$$

Therefore, when μ tends to zero, any accumulation point of the sequence satisfies

$$(\forall\, i = 1, \ldots, m) \quad c_{(i)}(x) = 0 \quad \text{or} \quad \lambda_{(i)} = 0, \quad c(x) \geq 0 \quad \text{and} \quad \lambda \geq 0,$$

which corresponds, with the first equality of (2.1), to the optimality conditions of (1.1).

For each fixed value of μ, an approximate solution of (2.1) is computed by means of a sequence of Newton iterations, called inner iterations. Let us denote by $z_k := (x_k, \lambda_k)$ the vector pair of primal-dual iterates at the k-th inner iteration. The iterates are generated according to the recurrence

$$z_{k+1} = z_k + \alpha_k d_k,$$

where d_k is the Newton direction, obtained by a linearization of (2.1) at z_k and α_k is a step length. The direction $d_k := (d_k^x, d_k^\lambda)$ is the solution of the following linear system

$$\begin{pmatrix} M_k & -\nabla c(x_k) \\ \Lambda_k \nabla c(x_k)^\top & C(x_k) \end{pmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \nabla c(x_k)\lambda_k \\ C(x_k)\lambda_k - \mu e \end{pmatrix}, \tag{2.2}$$

in which M_k is a positive definite matrix approximating of the Hessian of the Lagrangian at (x_k, λ_k) and $\Lambda_k = \operatorname{diag}((\lambda_k)_{(1)}, \ldots, (\lambda_k)_{(m)})$. The step length is computed by means of a backtracking line search, performed on the primal-dual merit function

$$\psi_\mu(x, \lambda) = f(x) - \mu \sum_{i=1}^{m} \log c_{(i)}(x) + \sum_{i=1}^{m} \big(\lambda_{(i)} c_{(i)}(x) - \mu \log(\lambda_{(i)} c_{(i)}(x))\big).$$

Starting from the unit step ($\alpha = 1$), the value of the step length is progressively reduced, by using some safeguard rules, until the following sufficient decrease condition is satisfied:

$$\psi_\mu(z_k + \alpha_k d_k) \leq \psi_\mu(z_k) + \alpha_k \omega \nabla \psi_\mu(z_k)^\top d_k,$$

where $\omega \in (0, 1)$. The aim of the line search is to guarantee the global convergence of the inner iterates. At last, the approximate Hessian matrix is updated by the BFGS formula

$$M_{k+1} = M_k - \frac{M_k s_k s_k^\top M_k}{s_k^\top M_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}, \tag{2.3}$$

where s_k and y_k are defined by

$$s_k := x_{k+1} - x_k \quad \text{and} \quad y_k := \nabla_x \ell(x_{k+1}, \lambda_{k+1}) - \nabla_x \ell(x_k, \lambda_{k+1}). \tag{2.4}$$

The positive definiteness of M_{k+1} is guaranteed provided $y_k^\top s_k > 0$. Whenever $y_k^\top s_k$ is not sufficiently positive, one can use some correction formula (see e.g., [3, § 18.4]) or simply skip the update.

The main computational part rests in the solution of the linear system (2.2). By eliminating d_k^λ from the first equation in (2.2), the primal direction is the solution of the following linear system:

$$(M_k + \nabla c(x_k) \Lambda_k C(x_k)^{-1} \nabla c(x_k)^\top) d_k^x = -\nabla f(x_k) + \mu \nabla c(x_k)\, C(x_k)^{-1} e. \tag{2.5}$$

Then, the dual direction is computed with only some matrix-vector products:

$$d_k^\lambda = -\lambda_k + C(x_k)^{-1}(\mu e - \Lambda_k \nabla c(x_k)^\top d_k^x). \tag{2.6}$$

When n is large, a full representation of the $n \times n$ matrix M_k is too expansive to store in memory. Instead, limited memory techniques keep only a few number of vector pairs (s_k, y_k) of length $2n$, that allows to have an implicit representation of the matrix M_k (see [4, 5, 6]). In our case, the coefficient matrix of the linear system (2.5) is of the form $M_k + N_k$, where M_k is the BFGS approximation and N_k is symmetric and positive semidefinite. The following proposition shows that the compact representation of BFGS matrices introduced in [2], allows to represent the matrix $M_k + N_k$ in a compact form.

Proposition 2.1 *Let M_0 be a symmetric and positive definite matrix and let the $n \times k$ matrices $S_k = \begin{bmatrix} s_0 & \ldots & s_{k-1} \end{bmatrix}$ and $Y_k = \begin{bmatrix} y_0 & \ldots & y_{k-1} \end{bmatrix}$. Suppose that the k vector pairs $\{s_i, y_i\}_{i=0}^{k-1}$ satisfy $s_i^\top y_i > 0$. Let N_k be a symmetric and positive semidefinite matrix and let M_k be obtained by updating M_0 k times using the direct BFGS formula (2.3) and the pairs $\{s_i, y_i\}_{i=0}^{k-1}$. Then, the inverse of $M_k + N_k$ is given by*

$$(M_k + N_k)^{-1} = (M_0 + N_k)^{-1} + (M_0 + N_k)^{-1} U_k E_k^{-1} U_k^\top (M_0 + N_k)^{-1}, \tag{2.7}$$

where $U_k = \begin{bmatrix} M_0 S_k & Y_k \end{bmatrix}$ and $E_k = \begin{bmatrix} S_k^\top \tilde{M}_0 S_k & \tilde{L}_k \\ \tilde{L}_k^\top & -\tilde{D}_k \end{bmatrix}$, with

$$\begin{aligned} \tilde{M}_0 &= M_0 - M_0 (M_0 + N_k)^{-1} M_0, \\ \tilde{L}_k &= L_k - S_k^\top M_0 (M_0 + N_k)^{-1} Y_k, \\ \tilde{D}_k &= D_k + Y_k^\top (M_0 + N_k)^{-1} Y_k, \end{aligned}$$

where L_k is the $k \times k$ lower triangular matrix

$$(L_k)_{i,j} = \begin{cases} s_{i-1}^\top y_{j-1} & \text{if } i > j, \\ 0 & \text{otherwise,} \end{cases}$$

and $D_k = \operatorname{diag}\left(s_0^\top y_0, \ldots, s_{k-1}^\top y_{k-1}\right)$. Moreover, the Schur complement of $-\tilde{D}_k$ in E_k

$$S_k^\top \tilde{M}_0 S_k + \tilde{L}_k \tilde{D}_k^{-1} \tilde{L}_k^\top,$$

is a positive definite matrix.

Proof. Using the compact representation of M_k [2, Theorem 2.3], we have

$$M_k + N_k = M_0 + N_k - U_k \begin{bmatrix} S_k^\top M_0 S_k & L_k \\ L_k^\top & -D_k \end{bmatrix}^{-1} U_k^\top.$$

Applying the Sherman-Morrison-Woodbury formula [7] to $M_k + N_k$ we obtain

$$(M_k + N_k)^{-1} = (M_0 + N_k)^{-1} + (M_0 + N_k)^{-1} U_k E_k^{-1} U_k^\top (M_0 + N_k)^{-1},$$

where

$$E_k = \begin{bmatrix} S_k^\top M_0 S_k & L_k \\ L_k^\top & -D_k \end{bmatrix} - U_k^\top (M_0 + N_k)^{-1} U_k,$$

from which the formula of E_k follows.

To show that E_k is nonsingular, let us first remark that since the k vector pairs satisfy $s_i^\top y_i > 0$, the diagonal matrix D_k is positive definite and so is $\tilde{D}_k$. Therefore, the matrix E_k may be decomposed as

$$\begin{bmatrix} I & -\tilde{L}_k \tilde{D}_k^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} S_k^\top \tilde{M}_0 S_k + \tilde{L}_k \tilde{D}_k^{-1} \tilde{L}_k^\top & 0 \\ 0 & -\tilde{D}_k \end{bmatrix} \begin{bmatrix} I & 0 \\ -\tilde{D}_k^{-1} \tilde{L}_k^\top & I \end{bmatrix}. \tag{2.8}$$

The proof will be completed if we show that the Schur complement (the upper left block in the middle matrix) is nonsingular. Suppose that $(S_k^\top \tilde{M}_0 S_k + \tilde{L}_k \tilde{D}_k^{-1} \tilde{L}_k^\top)u = 0$ for some vector u. Writing $\tilde{M}_0 = M_0 \left(M_0^{-1} - (M_0 + N_k)^{-1}\right) M_0$, we easily see that $\tilde{M}_0$ is positive semidefinite and hence $S_k^\top \tilde{M}_0 S_k + \tilde{L}_k \tilde{D}_k^{-1} \tilde{L}_k^\top$ is positive semidefinite too. We then deduce that $\tilde{M}_0 S_k u = 0$ and $\tilde{L}_k^\top u = 0$. These two equalities imply that $S_k u = (M_0 + N_k)^{-1} M_0 S_k u$ and next

$$\begin{aligned} L_k^\top u &= Y_k^\top (M_0 + N_k)^{-1} M_0 S_k u \\ &= Y_k^\top S_k u. \end{aligned}$$

By definition of L_k, the matrix $S_k^\top Y_k - L_k$ is upper triangular with positive diagonal elements, then $u = 0$. □

The algorithm keeps in memory a fixed number, say r, of the most recent vector pairs (s_k, y_k) defined by (2.4) and such that $y_k^\top s_k > 0$. At each iteration, if the new pair satisfies $y_k^\top s_k > 0$, the oldest pair is removed from the memory and the new one is stored. The storing matrices are then of the form

$$S_k = \begin{bmatrix} s_{\max(0,k-r)} & \dots & s_{k-1} \end{bmatrix} \quad \text{and} \quad Y_k = \begin{bmatrix} y_{\max(0,k-r)} & \dots & y_{k-1} \end{bmatrix}, \tag{2.9}$$

the matrices L_k and D_k of Proposition 2.1 being modified accordingly.

At each iteration, a new initial matrix is computed and its choice is based on the following considerations. When $N_k = 0$, formula (2.7) gives the compact representation of the inverse matrix M_k^{-1}, see [2, Theorem 2.2]. For the updating the inverse M_k^{-1}, it has been shown that a relevant choice is to set the initial matrix to $(y^\top s / y^\top y)I$ where (s, y) is the most recent vector pair stored in memory (see [5, formula (4.1)]). As a consequence, the scaling factor and the initial matrix are set to

$$\sigma_k = \frac{y_{k-1}^\top y_{k-1}}{y_{k-1}^\top s_{k-1}} \quad \text{and} \quad M_0 = \sigma_k I. \tag{2.10}$$

We found that this choice performs well in practice.

A complete description of the algorithm is given in Appendix A.

3 Memory storage and computational cost

Let us estimate the amount of memory and the computational cost required for computing the direction d_k with formulæ (2.5), (2.6) and (2.7). We will distinguish two cases, depending on the structure of the constraints. When the constraints are reduced to simple bounds ($p = 0$), the matrix

$$N_k := \nabla c(x_k) \Lambda_k C(x_k)^{-1} \nabla c(x_k)^\top$$

in (2.5) is diagonal and thus the computation of $(M_0 + N_k)^{-1}$ in (2.7) is reduced to $2n$ divisions. In the general case ($p \geq 1$), the matrix N_k is no longer diagonal and the computation of d_k^x requires the factorization of some $p \times p$ matrix.

To simplify the estimation, we assume that $k > r$, so that the size of the matrices in (2.9) is $n \times r$, that there is no skipped update and that there are exactly $2n$ bound constraints. The number of operations that will be given correspond only to multiplications and divisions.

3.1 Bound constraints case

Suppose that the set of constraints of (1.1) is defined by

$$c(x) = \begin{pmatrix} x - l \\ u - x \end{pmatrix},$$

where $-\infty < l_{(i)} < u_{(i)} < +\infty$, for all $i = 1, \ldots, n$. The matrix N_k is diagonal and its diagonal elements are of the form

$$\frac{(\lambda_k)_{(i)}}{(x_k)_{(i)} - l_{(i)}} + \frac{(\lambda_k)_{(n+i)}}{u_{(i)} - (x_k)_{(i)}}.$$

The direction d_k is computed by means of the following algorithm. Suppose that, at the iteration k, the matrices S_{k-1}, Y_{k-1}, D_{k-1}, L_{k-1} and the vector pair (s_k, y_k) which satisfies $s_k^\top y_k > 0$ are available. For each step, an estimation of the number of operations is given in brackets.

Computation of d_k in the bound constraints case

1. Compute N_k. $[2n]$
2. Update S_k, Y_k, D_k and L_k. $[nr]$
3. Compute $\tilde{D}_k$, $\tilde{L}_k$ and $S_k^\top \tilde{M}_0 S_k$. $[nr(7r + 5)/2 + O(r^2)]$
4. $u := (M_0 + N_k)^{-1}(-\nabla f(x_k) + \mu \nabla c(x_k)\, C(x_k)^{-1} e)$. $[4n]$
5. $v := U_k^\top u$. $[(2n + 1)r]$
6. Solve $E_k\, w = v$. $[O(r^3)]$
7. $d_k^x := u + (M_0 + N_k)^{-1} U_k\, w$. $[n(2r + 2)]$
8. $d_k^\lambda := -\lambda_k + C(x_k)^{-1}(\mu e - \Lambda_k \nabla c(x_k)^\top d_k^x)$. $[4n]$

The computation of v at Step 5 is done by using the following block products:

$$\begin{cases} v_1 &= S_k^\top M_0 u, \\ v_2 &= Y_k^\top u. \end{cases}$$

Then, at Step 6, the linear system is solve by using the decomposition (2.8):

$$\begin{bmatrix} I & -\tilde{L}_k \tilde{D}_k^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} S_k^\top \tilde{M}_0 S_k + \tilde{L}_k \tilde{D}_k^{-1} \tilde{L}_k^\top & 0 \\ 0 & -\tilde{D}_k \end{bmatrix} \begin{bmatrix} I & 0 \\ -\tilde{D}_k^{-1} \tilde{L}_k^\top & I \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.$$

The solution requires the factorization of the two $r \times r$ matrices $\tilde{D}_k$ and $S_k^\top \tilde{M}_0 S_k + \tilde{L}_k \tilde{D}_k^{-1} \tilde{L}_k^\top$. Since these two matrices are positive definite (Proposition 2.1), one can use a Cholesky factorization.

The total cost to compute d_k with the above algorithm is then $n(7r^2+15r+24)/2+O(r^3)$ operations. The memory storage requirement of the storing matrices S_k, Y_k and of all intermediate matrices is about $2n(r+1)+O(r^2)$.

3.2 General case

We suppose now that the set of constraints of (1.1) is of the general form (1.2). Let us denote by $b(x)$ the vector of bound constraints, so that the vector of constraints and their gradient matrix are partitioned as follows:

$$c(x) = \begin{pmatrix} g(x) \\ b(x) \end{pmatrix} \quad \text{and} \quad \nabla c(x) = \begin{pmatrix} \nabla g(x) & \nabla b(x) \end{pmatrix}.$$

Using this partition, let us denote by $\lambda := (\lambda^g, \lambda^b)$ the vector of multipliers and by G, B, Λ^g, Λ^b the corresponding diagonal matrices. The matrix N_k can be written

$$N_k = \nabla b(x_k) \Lambda_k^b B(x_k)^{-1} \nabla b(x_k)^\top + \nabla g(x_k) \Lambda_k^g G(x_k)^{-1} \nabla g(x_k)^\top. \tag{3.1}$$

To compute the inverse of $M_0 + N_k$ we proceed as follows. Using (2.10) and (3.1) one has

$$M_0 + N_k = \Delta_k + \nabla g(x_k) \Lambda_k^g G(x_k)^{-1} \nabla g(x_k)^\top,$$

where $\Delta_k := \sigma_k I + \nabla b(x_k) \Lambda_k^b B(x_k)^{-1} \nabla b(x_k)^\top$ is a diagonal matrix. Applying the Sherman-Morrison-Woodbury formula, we obtain

$$\begin{aligned}(M_0 + N_k)^{-1} = \\ \Delta_k^{-1} - \Delta_k^{-1} \nabla g(x_k) [(\Lambda_k^g)^{-1} G(x_k) + \nabla g(x_k)^\top \Delta_k^{-1} \nabla g(x_k)]^{-1} \nabla g(x_k)^\top \Delta_k^{-1}.\end{aligned} \tag{3.2}$$

The inversion of M_0+N_k requires the factorization of the $p \times p$ positive definite matrix in brackets. When the number of inequality constraints is not too large, this computation is acceptable.

The algorithm to compute d_k is quite similar to the one described previously. Only the first step is different. We denote by H_k the Cholesky factor of the bracketed matrix in (3.2).

Computation of d_k in the general case

1. Compute Δ_k and H_k. $[2n + np(p+1) + O(p^3)]$
2. Update S_k, Y_k, D_k and L_k. $[nr]$
3. Compute $\tilde{D}_k$, $\tilde{L}_k$ and $S_k^\top \tilde{M}_0 S_k$. $[nr((7r+5)/2 + 2p) + O(rp(p+r))]$
4. $u := (M_0 + N_k)^{-1}(-\nabla f(x_k) + \mu \nabla c(x_k)\, C(x_k)^{-1} e)$. $[n(3p+4) + O(p^2)]$
5. $v := U_k^\top u$. $[(2n+1)r]$

6. Solve $E_k\, w = v$. $[O(r^3)]$
7. $d_k^x := u + (M_0 + N_k)^{-1} U_k\, w$. $[n(3p + 2r + 2) + O(p^2)]$
8. $d_k^\lambda := -\lambda_k + C(x_k)^{-1}(\mu e - \Lambda_k \nabla c(x_k)^\top d_k^x)$. $[n(p + 4) + 2p]$

The matrices of Step 3 are computed as follows. From equality (3.2) we can write

$$\begin{aligned} \tilde{D}_k &= D_k + Y_k^\top \Delta_k^{-1} Y_k - (\tilde{Y}_k)^\top \tilde{Y}_k, \\ \tilde{L}_k &= L_k - S_k^\top M_0 \Delta_k^{-1} Y_k + (\tilde{S}_k)^\top \tilde{Y}_k, \\ S_k^\top \tilde{M}_0 S_k &= S_k^\top M_0 S_k - S_k^\top M_0 \Delta_k^{-1} M_0 S_k + (\tilde{S}_k)^\top \tilde{S}_k, c \end{aligned}$$

where

$$\tilde{Y}_k := H_k^{-1} \nabla g(x_k)^\top \Delta_k^{-1} Y_k \quad \text{and} \quad \tilde{S}_k := H_k^{-1} \nabla g(x_k)^\top \Delta_k^{-1} S_k$$

are $p \times r$ matrices. It follows that the computing cost of the three matrices at Step 3, is increased by the one of $\tilde{Y}_k$ and $\tilde{S}_k$.

The total cost to compute the direction d_k in the general case is equal to $n(2p^2+7r^2+4pr+16p+15r+24)/2 + O(p^3) + O(r^3)$ operations. The memory storage requirement is equal to the one of the bound constraints case plus the storage of the matrices H_k, $\tilde{Y}_k$, $\tilde{S}_k$ and one vector of $\mathbb{R}^p$. The total memory storage requirement is approximately equal to $2n(r+1) + O((p+r)^2)$.

All details about these computations are given in [8].

4 Convergence analysis

In this section, we show that the sequence of inner iterates converges with a r-linear rate. This result relies on the assumption that the problem has a strong convexity property.

Assumption 4.1 *The functions f and $-c_{(i)}$ $(1 \leq i \leq m)$ are convex and differentiable from $\mathbb{R}^n$ to $\mathbb{R}$. There exists $\lambda \in \mathbb{R}^m$ and $\kappa > 0$, such that for all $(x, y) \in \mathbb{R}^n \times \mathbb{R}^n$, $(\nabla_x \ell(x, \lambda) - \nabla_x \ell(y, \lambda))^\top (x - y) \geq \kappa \|x - y\|^2$.*

Under this assumption, it can be shown that the perturbed optimality conditions (2.1) have a unique solution and that whenever M_k is updated by the BFGS formula, the sequence of inner iterates converges r-linearly to this solution (Theorem 3.4 in [1] is proved with a stronger assumption, but it is still valid under Assumption 4.1, see [9, Theorem 4.10]). The convergence result is proved thanks to a well known property of the BFGS formula [10, Theorem 2.1]. This property asserts that the BFGS formula generates a constant fraction of "good iterates". It means that if there exist constants $a_1 > 0$ and $a_2 > 0$ such that

$$y_k^\top s_k \geq a_1 \|s_k\|^2 \quad \text{and} \quad y_k^\top s_k \geq a_2 \|y_k\|^2 \tag{4.1}$$

for all $k \geq 1$, then there exist positive constants β_1, β_2, and β_3, such that

$$\beta_1 \leq \frac{s_k^\top M_k s_k}{\|M_k s_k\| \|s_k\|} \quad \text{and} \quad \beta_2 \leq \frac{\|M_k s_k\|}{\|s_k\|} \leq \beta_3 \tag{4.2}$$

for at least half of the iterates. Since our limited memory algorithm and the algorithm proposed in [1] differ only in the way of updating the BFGS matrices, the convergence of the inner iterates will be proved if we show that it generates a sequence of good iterates. This result is actually already proved in the framework of an unconstrained limited memory algorithm [5, Theorem 7.1]. To be complete, we state the result formally and give a proof.

The matrix M is said to be generated by the L-BFGS formula from an initial matrix $M^{(0)}$ and using q pairs $\{(s_i, y_i)\}_{i=0}^{q-1}$, if M is computed according to the following recurrence: for $i = 0, \ldots, q-1$ compute

$$M^{(i+1)} = M^{(i)} - \frac{M^{(i)} s_i s_i^\top M^{(i)}}{s_i^\top M^{(i)} s_i} + \frac{y_i y_i^\top}{y_i^\top s_i}, \tag{4.3}$$

then set $M = M^{(q)}$.

Lemma 4.2 *Let $\{M_k^{(0)}\}$ be a bounded sequence of positive definite matrices with bounded inverses. Let $\{(s_k, y_k)\}_{k \geq 1}$ be a sequence of vector pairs such that (4.1) holds for constants $a_1 > 0$ and $a_2 > 0$. Let q be a positive integer and let $\{M_k\}$ be a sequence of positive definite matrices such that each M_k is generated by the L-BFGS formula from the initial matrix $M_k^{(0)}$ and using the vector pairs $\{(s_{i_j}, y_{i_j})\}_{j=0}^{q_k-1}$ where $1 \leq q_k \leq q$. Then, there exist positive constants β_1, β_2, and β_3, such that (4.2) holds for any index $k \geq 1$,*

Proof. Using (4.3), the second inequality in (4.1), $q_k \leq q$ and the boundedness of the sequence $\{M_k^{(0)}\}$, the trace of M_k can be bounded above as follows:

$$\operatorname{tr}(M_k) \leq \operatorname{tr}(M_k^{(0)}) + \sum_{j=0}^{q_k-1} \frac{\|y_{i_j}\|^2}{y_{i_j}^\top s_{i_j}} \leq \operatorname{tr}(M_k^{(0)}) + q a_2^{-1} \leq K_1, \tag{4.4}$$

where K_1 is a positive constant. The formula for the determinant of M_k is given by (see [11])

$$\det(M_k) = \det(M_k^{(0)}) \prod_{j=0}^{q_k-1} \frac{y_{i_j}^\top s_{i_j}}{s_{i_j}^\top M_k^{(j)} s_{i_j}} = \det(M_k^{(0)}) \prod_{j=0}^{q_k-1} \frac{y_{i_j}^\top s_{i_j}}{s_{i_j}^\top s_{i_j}} \frac{s_{i_j}^\top s_{i_j}}{s_{i_j}^\top M_k^{(j)} s_{i_j}}.$$

Using the first inequality in (4.1), (4.4) and the boundedness of the inverses of $M_k^{(0)}$, one has

$$\det(M_k) \geq \det(M_k^{(0)})(a_1 K_1^{-1})^{q_k} \geq \det(M_k^{(0)}) \min\left(1, (a_1 K_1^{-1})^q\right) \geq K_2, \tag{4.5}$$

for some positive constant K_2.

From (4.4) and (4.5) we deduce that the matrices M_k and M_k^{-1} are uniformly bounded. We then conclude that inequalities (4.2) are satisfied. □

Theorem 4.3 *Suppose that Assumption* 4.1 *holds and that f and c are differentiable with Lipschitzian continuous derivatives. Then, the sequence of inner iterates $\{z_k\}$ converges r-linearly to the unique solution of* (2.1).

Proof. The inequalities (4.1) are consequences of the strong convexity of the Lagrangian (second part of Assumption (4.1)) and of the Lipschitz continuity of the derivatives of f and c (see [9, Lemma 4.5] for a proof). The initial matrices are of the form $\sigma_k I$, where σ_k is given by (2.10). From (4.1), one has $a_1 \leq \sigma_k \leq 1/a_2$. We can then apply Lemma 4.2 and according to the discussion made at the beginning of the section, we obtain the r-linear convergence of the inner iterates. □

5 Numerical experiments

We have tested our limited memory interior point algorithm named `NOPTIQ` (Nonlinear Optimization with interior point and quasi-Newton techniques) in the environment `CUTEr` [12] and compared our results with those obtained with `l-BFGS-B` [13] for the bound constrained problems, and with those obtained with `LANCELOT` [14] for the problems with general inequality constraints.

`NOPTIQ` is written in Fortran 77 and all numerical tests were performed on a PC with 256 MB of RAM. In order to compare the results of `NOPTIQ` with those obtained with the two other codes, we have used the same stopping criterion as the one used by `l-BFGS-B` and `LANCELOT`. Let $P_{l,u} : \mathbb{R}^n \to \mathbb{R}^n$ be the projection operator onto the rectangular box defined by the bound constraints. The projection is defined componentwise by

$$\big(P_{l,u}(x)\big)_{(i)} = \begin{cases} l_{(i)} & if \quad x_{(i)} < l_{(i)}, \\ x_{(i)} & if \quad l_{(i)} \leq x_{(i)} \leq u_{(i)}, \\ u_{(i)} & if \quad u_{(i)} < x_{(i)}. \end{cases}$$

When the problem has only bound constraints, `NOPTIQ` is terminated whenever

$$\|x_k - P_{l,u}(x_k - \nabla f(x_k))\| \leq 10^{-8}, \tag{5.1}$$

otherwise, it is terminated when

$$\|x_k - P_{l,u}(x_k - \nabla_x \ell(x_k, \lambda_k))\| \leq 10^{-8} \quad \text{and} \quad \|G(x_k)\lambda_k\| \leq 10^{-8}. \tag{5.2}$$

A limit of 10000 iterations was also imposed to the codes. The number of vectors pairs kept in memory is set to $r = 5$ for both `NOPTIQ` and `l-BFGS-B`. The second order information in `LANCELOT` is obtained by using the BFGS option. Other settings for `l-BFGS-B` and `LANCELOT` are the default values.

We report the numerical results as in [15]. For the i-th test problem, we compute the relative number of function evaluations and the relative CPU time to solve the problem:

$$q_i = -\log_2(nf_{\texttt{NOPTIQ}}/nf_{\texttt{C}}) \quad \text{and} \quad t_i = -\log_2(t_{\texttt{NOPTIQ}}/t_{\texttt{C}}),$$

where `C` stands for `l-BFGS-B` or `LANCELOT`. These quantities are computed only if the codes converge to the same solution. If `NOPTIQ` is better than the other code in term

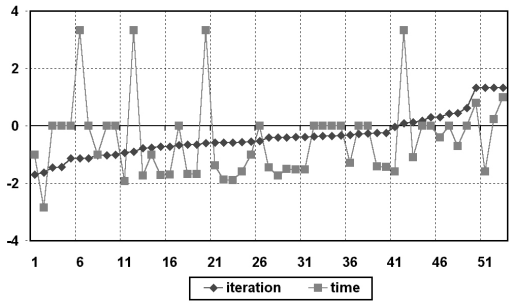

Figure 1: Relative performance of NOPTIQ and l-BFGS-B for the problems in Table 1.

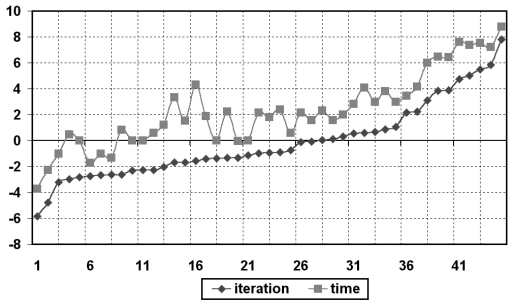

Figure 2: Relative performance of NOPTIQ and LANCELOT for problems in Table 2.

of function evaluations, then the value of q_i is positive and the number of times that NOPTIQ outperforms the other code is 2^{q_i}. In the same way, $t_i > 0$ whenever NOPTIQ is better than the other code in term of CPU time.

The numerical tests NOPTIQ versus l-BFGS-B are reported in Table 1 (Appendix B) and Figure 1. These tests include only bound constraints problems, because l-BFGS-B cannot solve problems with general inequality constraints. We notice that even if the number of function evaluations made by NOPTIQ is often greater than that of l-BFGS-B, the CPU times are often equivalent for both codes.

The numerical tests NOPTIQ versus LANCELOT are reported in Table 2 (Appendix B) and Figure 2. They include more general inequality constrained problems, but our choice was restricted because the starting point for NOPTIQ needs to be strictly feasible. We can notice that, in term of CPU time, NOPTIQ outperforms LANCELOT for almost all the problems and, in term of function evaluations, NOPTIQ is better than LANCELOT for almost half of the test problems.

We can conclude that NOPTIQ is a viable solution to solve inequality constrained problems, with a low memory storage and a low computational cost per iteration. It is of particular interest when the problem is large, with a medium number of nonlinear inequalities and when the second derivatives are unavailable. Moreover, the algorithm is simple to implement. Our preliminary experiments show that NOPTIQ has comparable results with l-BFGS-B for solving bound constrained problems. We think that its ability to solve a larger class of problems could be interesting for real-world applications. The main drawback of our implementation is the need of a strictly feasible starting point. We plan to implement a new version for handling infeasible starting points with the strategy described in [16].

The Fortran code NOPTIQ can be obtained at http://www.fist.fr.

A Algorithm

1. Choose the constant values $\omega \in (0,1)$, $0 < \xi < \xi' < 1$, $\eta > 0$ and the number r of vector pairs kept in memory. Choose the initial point $z_0^0 = (x_0^0, \lambda_0^0)$ with $c(x_0^0) > 0$ and $\lambda_0^0 > 0$, the initial value $\mu^0 > 0$ for the barrier parameter, the initial stopping tolerances $\epsilon_l^0 > 0$ and $\epsilon_c^0 > 0$. Set the outer iteration counter $j := 0$. Set the storing matrices $S_0^0 := [\,]$ and $Y_0^0 := [\,]$.

2. Set the inner iteration counter $k := 0$.

3. If the matrices S_k^j and Y_k^j are empty, then set the scaling factor $\sigma_k^j := 1$, otherwise set $\sigma_k^j := \frac{y^\top y}{y^\top s}$, where (s, y) is the most recently vector pair kept in memory.

4. Compute the primal direction $(d^x)_k^j$ by (2.5) and (2.7), with $x = x_k^j$, $\lambda = \lambda_k^j$ and $\mu = \mu^j$, the initial matrix $M_k^{(0)} = \sigma_k^j I$ and the matrices S_k^j and Y_k^j.

5. Compute the dual direction $(d^\lambda)_k^j$ by (2.6), with $x = x_k^j$, $\lambda = \lambda_k^j$ and $\mu = \mu^j$.

6. Set $\alpha = 1$. While the sufficient decrease condition
$$\psi_{\mu^j}(z_k^j + \alpha d_k^j) \leq \psi_{\mu^j}(z_k^j) + \alpha\omega\nabla\psi_{\mu^j}(z_k^j)^\top d_k^j$$
is not satisfied, choose a new trial step α in $[\xi\alpha, \xi'\alpha]$.

7. Set $z_{k+1}^j := z_k^j + \alpha d_k^j$.

8. Define the vector pair
$$s_k^j := x_{k+1}^j - x_k^j \quad \text{and} \quad y_k^j := \nabla_x \ell(x_{k+1}^j, \lambda_{k+1}^j) - \nabla_x \ell(x_k^j, \lambda_{k+1}^j).$$

9. If $(y_k^j)^\top s_k^j \geq \eta \|y_k^j\|^2$, then delete the oldest vector pair and add the new one to S_k^j and Y_k^j to form the matrices S_{k+1}^j and Y_{k+1}^j, otherwise set $S_{k+1}^j := S_k^j$ and $Y_{k+1}^j = Y_k^j$.

10. If the stopping criterion of the inner iterations
$$\|\nabla_x \ell(x_k^j, \lambda_k^j)\| \leq \epsilon_l^j \quad \text{and} \quad \|C(x_k^j)\lambda_k^j - \mu^j e\| \leq \epsilon_c^j,$$
is not satisfied, then increase k by 1 and go to step 4.

11. If the stopping criterion (5.1) or (5.2) of the outer iterations is not satisfied, then set the new barrier parameter $\mu^{j+1} := \mu^j/\beta$, the precision thresholds $\epsilon_l^{j+1} := 10\mu^{j+1}/\mu^0$ and $\epsilon_c^{j+1} := 0.999\,\mu^{j+1}$. Set the new starting iterate $z_1^{j+1} := z_{k+1}^j$ for the next outer iteration and set $S_1^{j+1} := S_{k+1}^j$ and $Y_1^{j+1} := Y_{k+1}^j$. Increase j by 1 and go to step 2.

The numerical experiments were done with the following constant values: $\omega := 10^{-4}$, $\mu^0 := 1$, $\beta := 10$, $\xi := 10^{-2}$, $\xi\prime := 0.95$ and η is set to the machine epsilon. The initial multiplier λ_0^0 is computed as the solution of $C(x_0^0)\lambda_0^0 = \mu^0 e$. For more details about the implementation, see [8].

B Tables

	Problem			NOPTIQ		l-BFGS-B	
	Name	n	nb	nf	$time$	nf	$time$
1	PALMER4	4	3	81	0,02	25	0,01
2	PALMER3A	6	2	1948	0,29	633	0,04
3	HS3	2	1	11	0,01	4	0,01
4	HS38	4	8	70	0,01	26	0,01
5	SIMBQP	2	2	11	0,01	5	0,01
6	PSPDOC	4	1	24	0,002	11	0,02
7	S368	8	16	24	0,01	11	0,01
8	HATFLDC	25	48	47	0,02	23	0,01
9	PALMER5D	8	4	49	0,01	24	0,01
10	OBSTCLAE	100	128	38	0,02	19	0,02
11	PALMER1A	6	2	1484	0,19	771	0,05
12	HS5	2	4	15	0,001	8	0,01
13	OBSTCLBM	10000	19208	189	26,94	110	8,07
14	HS1	2	1	49	0,02	29	0,01
15	TORSION1	5476	10368	279	19,97	168	6,05
16	TORSION2	5476	10368	279	19,66	168	6,05
17	HIMMELP1	2	4	27	0,02	17	0,02
18	TORSIONA	5476	10368	294	23,46	187	7,35
19	TORSIONB	5476	10368	294	23,49	187	7,33
20	HART6	6	12	29	0,001	19	0,01
21	PALMER6A	6	2	1142	0,13	759	0,05
22	TORSION5	5476	10368	178	13,42	119	3,65
23	TORSION6	5476	10368	178	13,45	119	3,63
24	PALMER2A	6	2	1260	0,18	851	0,06
25	PALMER3B	4	2	83	0,02	57	0,01
26	HS3MOD	2	1	13	0,01	9	0,01
27	MCCORMCK	5000	10000	20	1,252	15	0,46
28	TORSION3	5476	10368	240	19,8	182	5,95
29	TORSION4	5476	10368	240	16,53	182	5,85
30	TORSIONC	5476	10368	234	18,51	178	6,42
31	TORSIOND	5476	10368	234	18,42	178	6,39
32	PALMER2B	4	2	80	0,02	62	0,02
33	MDHOLE	2	1	113	0,02	88	0,02
34	PALMER2	4	3	55	0,01	43	0,01
35	LOGROS	2	2	146	0,02	115	0,02
36	NOBNDTOR	5476	10368	505	36,12	404	14,72
37	CAMEL6	2	4	17	0,01	14	0,01
38	HS2	2	1	24	0,01	20	0,01
39	TORSIONE	5476	10368	163	12,77	138	4,79
40	TORSIONF	5476	10368	163	12,8	138	4,72
41	YFIT	3	1	109	0,03	106	0,01
42	HS45	5	10	15	0,001	16	0,01
43	NONSCOMP	5000	10000	47	2,503	51	1,17
44	ALLINIT	4	3	15	0,02	17	0,02
45	PALMER8A	6	2	307	0,03	378	0,03
46	LINVERSE	1999	1000	249	4,897	308	3,70
47	HATFLDA	4	4	29	0,01	39	0,01
48	OBSTCLAE	10000	19208	341	49,1	462	30,20
49	HATFLDB	4	5	22	0,01	34	0,01
50	BDEXP	5000	5000	6	0,22	15	0,38
51	BDEXP	100	100	10	0,03	25	0,01
52	BDEXP	1000	1000	10	0,06	25	0,07
53	BDEXP	500	500	10	0,02	25	0,04

Table 1: Numerical results of NOPTIQ and l-BFGS-B. The number of bounds on the variables is denoted by nb.

	Problem				NOPTIQ		LANCELOT	
	Name	n	nb	p	nf	$time$	nf	$time$
1	CVXBQP1	10000	20000	0	115	15,8	2	1,20
2	NCVXBQP1	10000	20000	0	136	10,79	5	2,24
3	HATFLDC	25	48	0	47	0,02	5	0,01
4	PALMER1A	6	2	0	1484	0,19	190	0,26
5	HS4	2	2	0	14	0,01	2	0,01
6	NONSCOMP	5000	10000	0	47	2,503	7	0,76
7	SINEALI	1000	2000	0	51	0,41	8	0,20
8	LUKVLI3	10000	2	2	161	24,91	26	9,82
9	WOMFLET	3	3	3	301	0,04	49	0,07
10	BQP1VAR	1	2	0	10	0,01	2	0,01
11	PALMER5D	8	4	0	49	0,01	10	0,01
12	PALMER4B	4	2	0	120	0,02	25	0,03
13	PFIT3LS	3	1	0	1458	0,16	358	0,37
14	HART6	6	12	0	29	0,001	9	0,01
15	PFIT2LS	3	1	0	687	0,08	214	0,23
16	HS5	2	4	0	15	0,001	5	0,02
17	PALMER2A	6	2	0	1260	0,18	479	0,67
18	HS3MOD	2	1	0	13	0,01	5	0,01
19	PFIT4LS	3	1	0	1603	0,16	639	0,76
20	MCCORMCK	5000	10000	0	20	1,252	8	1,22
21	HS3	2	1	0	11	0,01	5	0,01
22	MDHOLE	2	1	0	113	0,02	57	0,09
23	PALMER3B	4	2	0	83	0,02	43	0,07
24	PALMER6A	6	2	0	1142	0,13	606	0,69
25	HS1	2	1	0	49	0,02	29	0,03
26	PALMER2B	4	2	0	80	0,02	74	0,09
27	CAMEL6	2	4	0	17	0,01	16	0,03
28	HATFLDA	4	4	0	29	0,01	30	0,05
29	HATFLDB	4	5	0	22	0,01	24	0,03
30	HS36	3	7	1	16	0,01	20	0,04
31	HS35	3	4	1	46	0,01	68	0,07
32	PALMER8A	6	2	0	307	0,03	466	0,51
33	HS35I	3	7	1	43	0,01	68	0,08
34	PFIT1LS	3	1	0	519	0,07	949	0,97
35	HS12	2	1	1	25	0,01	52	0,08
36	BDEXP	5000	5000	0	6	0,22	27	2,38
37	HS113	10	8	8	22	0,01	103	0,18
38	PALMER4A	6	2	0	834	0,14	7148	9,07
39	PALMER3	4	3	0	430	0,08	6134	7,14
40	HS100	7	4	4	106	0,02	1547	1,71
41	HS38	4	8	0	70	0,01	1899	1,93
42	BQPGASIM	50	100	0	44	0,01	1420	1,67
43	YFIT	3	1	0	109	0,03	4837	5,55
44	HS70	4	9	1	96	0,06	5491	8,83
45	HIMMELP4	2	7	3	18	0,01	4017	4,38

Table 2: Numerical results of NOPTIQ and LANCELOT.

References

[1] P. Armand, J. C. Gilbert and S. Jan-Jégou. A feasible BFGS interior point algorithm for solving convex minimization problems. *SIAM J. Optim.*, 11(1):199–222 (electronic), 2000.

[2] R. H. Byrd, J. Nocedal and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(2, Ser. A):129–156, 1994.

[3] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer-Verlag, New York, 1999.

[4] A. Buckley and A. Lenir. QN-like variable storage conjugate gradients. *Mathematical Programming*, 27(2):155–175, 1983.

[5] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[6] J. C. Gilbert and C. Lemaréchal. Some numerical experiments with variable-storage quasi-Newton algorithms. *Mathematical Programming*, 45(3, (Ser. B)):407–435, 1989.

[7] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original.

[8] P. Ségalat. *Méthodes de points intérieurs et de quasi-Newton*. PhD thesis, LACO, Université de Limoges (France), December 2002.

[9] P. Armand, J. C. Gilbert and S. Jan-Jégou. A BFGS-IP algorithm for solving strongly convex optimization problems with feasibility enforced by an exact penalty approach. *Mathematical Programming*, 92(3, Ser. B):393–424, 2002. ISMP 2000, Part 2 (Atlanta, GA).

[10] R. H. Byrd and J. Nocedal. A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. *SIAM Journal on Numerical Analysis*, 26:727–739, 1989.

[11] J. D. Pearson. Variable metric methods of minimisation. *The Computer Journal*, 12:171–178, 1969/1970.

[12] N. I. M. Gould, D. Ordan and P. L. Toint. General CUTEr documentation. Technical Report TR/PA/02/13, CERFACS, Toulouse (France), 2002.

[13] R. H. Byrd, P. Lu, J. Nocedal and C. Y. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995.

[14] A. R. Conn, N. I. M. Gould and P. L. Toint. *LANCELOT*, volume 17 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1992. A Fortran package for large-scale nonlinear optimization (release A).

[15] J. L. Morales. A numerical study of limited memory BFGS methods. *Appl. Math. Lett.*, 15(4):481–487, 2002.

[16] P. Armand. A quasi-Newton penalty barrier method for convex minimization problems. *Comput. Optim. Appl.*, 26(1):5–34, 2003.