Descentwise inexact proximal algorithms for smooth optimization

Marc Fuentes, Jérôme Malick, Claude Lemaréchal

coap manuscript No.
(will be inserted by the editor)

Descentwise inexact proximal algorithms for smooth optimization

Marc Fuentes · Jérôme Malick · Claude Lemaréchal

September 27, 2011

Abstract The proximal method is a standard regularization approach in optimization. Practical implementations of this algorithm require (i) an algorithm to compute the proximal point, (ii) a rule to stop this algorithm, (iii) an update formula for the proximal parameter. In this work we focus on (ii), when smoothness is present – so that Newton-like methods can be used for (i): we aim at giving adequate stopping rules to reach overall efficiency of the method.

Roughly speaking, usual rules consist in stopping inner iterations when the current iterate is close to the proximal point. By contrast, we use the standard paradigm of numerical optimization: the basis for our stopping test is a "sufficient" decrease of the objective function, namely a fraction of the ideal decrease. We establish convergence of the algorithm thus obtained and we illustrate it on some ill-conditioned functions. The experiments show that combining a standard smooth optimization algorithm with the proposed inexact proximal scheme improves numerical behaviour for those problems.

Keywords Proximal algorithm · Quasi-Newton algorithms · Convex analysis

Mathematics Subject Classification (2000) 65K05 · 90C53

1 Introduction, motivations, ideas

1.1 Proximal regularization, inner iterations

We consider the minimization problem

$$\inf f = \inf_{x\in\mathbb{R}^n} f(x)\,, \tag{1.1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. In this standard situation of optimization, Newton-like algorithms (as e.g. quasi-Newton [DS83] or generalized Newton [QS93]) are suitable to solve efficiently problem (1.1); but in case of ill-conditioning, such methods may run into difficulties, and then call for regularization.

Inria - Rhône-Alpes, 655 avenue de l'Europe, F-38334 Montbonnot

A possible approach in this situation is the proximal algorithm, whose idea is to solve at each iteration k

$$\inf_{y\in\mathbb{R}^n} \tilde{f}_k(y) := f(y) + \frac{1}{2t_k}\|y - x_k\|^2 \tag{1.2}$$

and obtain the associated *proximal point*

$$p_k := \operatorname*{argmin}_{y\in\mathbb{R}^n} \tilde{f}_k(y)\,.$$

Here $t_k > 0$ is a regularization parameter varying with k; $\|\cdot\|$ is the Euclidean norm associated to some inner product $\langle\cdot,\cdot\rangle$ in $\mathbb{R}^n$. When f is moreover convex, the proximal point is well-defined for any x_k.

Originally presented in [BKL66, Chap. 5] for a quadratic objective f, this proximal algorithm consists in solving (1.1) via the iteration $x_{k+1} = p_k$. Naturally, the proximal point is easily computed when f is quadratic convex: it is the unique solution of a linear system, well-conditioned by construction; see [MR09] among others for a motivation. In the general convex case [Mar70, Roc76], however, p_k cannot be computed explicitly: a subalgorithm is needed to solve (1.2), generating iterates y^ℓ supposed to converge to p_k. In the smooth case, some variant of Newton is advocated for this; the above-mentioned difficulties should no longer occur, to the extent that $\tilde{f}_k$ in (1.2) is now well-conditioned. Nevertheless, the question then arises of the *internal stopping test*: for which ℓ can we set $x_{k+1} = y^\ell$ in order to proceed with the outer iteration?

Several papers consider this question of inner stopping criteria for proximal methods, in various contexts:

- smooth optimization (see [HZ08] which in turn points to many references),
- nonsmooth optimization (see in particular [CL93, LS97])
- operator theory (see an overview in the introduction of [SS01]).

The seminal work of T. Rocakfellar [Roc76] also considered approximations of proximal points. Following this paper, most of the existing stopping tests directly force y^ℓ to be close to p_k: some parameter ε_k tending to 0 is explicitly used at each outer iteration and an asymptotic property of the type $\|y^\ell - p_k\| = O(\varepsilon_k)$ is enforced. In fact, assume f to be convex; then $\tilde{f}_k$ in (1.1) is strongly convex and we can write for each y

$$\tilde{f}_k(y) \geqslant \tilde{f}_k(p_k) \geqslant \tilde{f}_k(y) + \langle \nabla\tilde{f}_k(y), p_k - y\rangle + \frac{1}{2t_k}\|p_k - y\|^2\,.$$

This opens the way to computable bounds on $\|y - p_k\|$, and thus to a control of the error within ε_k. For example, the above inequality yields $\|p_k - y\| \leqslant 2t_k\|\nabla\tilde{f}_k(y)\|$; the inner algorithm can therefore be stopped when $2t_k\|\nabla\tilde{f}_k(y^\ell)\| \leqslant \varepsilon_k$.

This type of approach, however, is not fully constructive: explicit rules for the management of (ε_k) should be given which, for overall efficiency, should be given "on line". In other words, the outer parameter ε_k should not be fixed beforehand at each k but should depend on the inner iterate y^ℓ.

1.2 Decreasing the true objective value

Admittedly, the property $x_{k+1} - p_k \to 0$ is desirable to preserve the spirit of the proximal algorithm. Nevertheless, keeping in mind that our objective is to solve (1.1),

the property $x_{k+1} \simeq p_k$ appears as secondary: what is important and relevant is to *decrease the actual objective* function f. Stopping tests forcing $f(x_k) - f(x_{k+1})$ to be large are therefore more attractive; they are based on the observation that

$$f(x_k) - f(p_k) \geqslant \frac{1}{2t_k}\|p_k - x_k\|^2 \tag{1.3}$$

because the proximal point improves the objective function $\tilde{f}_k$. By continuity, iterates y^ℓ produced by a subalgorithm solving (1.2) will eventually satisfy relaxed versions of (1.3).

An immediate idea is to stop the subalgorithm and set $x_{k+1} = y^\ell$ when

$$f(x_k) - f(y^\ell) \geqslant \frac{m_1}{2t_k}\|y^\ell - x_k\|^2\,, \tag{1.4}$$

$0 < m_1 < 1$ being a fixed tolerance; say $m_1 = 0.1$. As explained for example in [CL93], this is the approach used by bundle methods for a convex nonsmooth function f. However, the descent test (1.4) is suitable only in this particular situation, where y^ℓ is generated by a very special cutting-plane algorithm. In general, both sides of (1.4) are not "homogeneous", that is, not expressed in the same units: the righthand side does not measure differences of objective function.

To obviate this inconsistency, we consider the optimality condition of (1.2)

$$\frac{x_k - p_k}{t_k} = \nabla f(p_k). \tag{1.5}$$

It allows us to replace $\|p_k - x_k\|^2$ in (1.3) by $t_k\|\nabla f(p_k)\|\,\|p_k - x_k\|$. Reproducing the above reasoning, we may replace (1.4) by

$$f(x_k) - f(y^\ell) \geqslant \frac{m_1}{2}\|\nabla f(y^\ell)\|\,\|x_k - y^\ell\|, \tag{1.6}$$

which at least is homogeneous. It is of course satisfied by $y^\ell = p_k$: any algorithm to solve (1.2) will eventually pass this test.

Indeed, (1.6) makes good sense; it is strongly related to standard techniques of numerical optimization. First, it looks like the well-accepted Armijo rule [Arm66] of standard line-searches. Second, consider the Levenberg-Marquardt technique for smooth minimization (which goes back to [Lev44] and [Mar63]): at the k^{th} iteration, it solves a problem of the form

$$\min_y \quad f_k(y) + \frac{1}{2t_k}\|y - x_k\|^2\,, \tag{1.7}$$

where f_k is a suitable (quadratic) approximation of f near x_k; depending on the test

$$f(x_k) - f(y) \geqslant m_1\langle\nabla f(x_k), y - x_k\rangle\,, \tag{1.8}$$

x_k is then moved to y (and/or t_k is updated). We now observe that an iteration of Levenberg-Marquard is just the first inner iteration of the subalgorithm solving (1.2) when the same quadratic approximation f_k is used in both cases. Then (1.8) is not much different from (1.6). Up to this difference, we can therefore say that our proposal generalizes Levenberg-Marquardt by allowing several iterations before moving the iterate x_k and/or updating the regularization coefficient t_k.

In addition, Levenberg-Marquardt is just a variant of trust region (see the excellent review [Mor83]); in fact, (1.7) amounts to solving

$$\begin{cases} \min f_k(y) \\ \|y - x_k\| \leqslant \Delta x_k \end{cases}$$

for a suitable trust-region size Δx_k. The above discussion therefore applies to trust region as well: our approach brings it more flexibility, allowing independent updates of x_k, Δx_k, f_k.

Our proposal could even be made more similar to the classical ones if (1.6) would be replaced by

$$f(x_k) - f(y^\ell) \geqslant m_1 \langle g, y^\ell - x_k \rangle \,,$$

with $g = \nabla f(x_k)$ or $g = -\nabla f(y^\ell)$. This is dangerous for a nonconvex f, though: it does not imply $f(x_k) - f(y^\ell) > 0$.

1.3 Eliminating small moves

The descent property (1.6) is not sufficient; in particular, it does not guarantee the property $y^\ell \neq x_k$. As in standard line-searches, a second test is needed, forcing y^ℓ to be "substantially different" from x_k. Giving a precise meaning to the expression "substantially different" is not straightforward, though: p_k itself may be close to x_k (if t_k is small). As a result, the design of a second test is no longer so clear.

Continuing the comparison with classical methods, the second test could be

$$\begin{cases} \langle \nabla \tilde{f}(y^\ell), y^\ell - x_k \rangle \geqslant m_2 \langle \nabla f(x_k), y^\ell - x_k \rangle & \text{(Wolfe [Wol69])} \\ \quad \text{or} \\ f(x_k) - f(y^\ell) \leqslant m_2 \langle \nabla f(x_k), y^\ell - x_k \rangle & \text{(Goldstein-Price [GP67])} \end{cases}$$

for some $m_2 \in]m_1, 1[$. None of the above tests seems adapted to the present situation, though. In fact, they assume an angle condition between $\nabla f(x_k)$ and the "direction" $y^\ell - x_k$: in fact $\langle \nabla f(x_k), y^\ell - x_k \rangle$ must be "substantially negative" – a rather irrelevant condition in the nonconvex case, when y^ℓ is the proximal point.

Another natural possibility would be to require a substantial decrease of $\nabla \tilde{f}$ with respect to its initial value $\nabla f(x_k)$, namely

$$\|x_k - y^\ell - t_k \nabla f(y^\ell)\| \leqslant m_2 t_k \|\nabla f(x_k)\|$$

for some $m_2 \in]0, 1[$. We have not seriously studied this possibility, neither theoretically nor numerically.

Actually, we do not propose here any novel idea concerning the second test. In fact, $m_2 \in]0, 1[$ being another fixed tolerance, we complete (1.6) with the following relaxation of (1.5):

$$\|x_k - y^\ell - t_k \nabla f(y^\ell)\| \leqslant m_2 \max\{t_k \|\nabla f(y^\ell)\|, \|x_k - y^\ell\|\} \,; \tag{1.9}$$

it is obviously satisfied by $y^\ell = p_k$, but not by $y^\ell = x_k$ unless $\nabla f(x_k) = 0$. Condition (1.9) has already been studied in the literature of inexact proximal methods. It was first introduced by [SS99] in the general context of proximal algorithm for variational inequalities. This latter paper initiated a number of works on inexact proximal methods; among them, we point out [SS01] and [HS05]. In the latter, (1.9) appears explicitly

(with $t_k = 1$) in the context of smooth convex optimization. Connections with the above-mentioned references will become clear in section 2.

In this paper, we thus study an inexact proximal algorithm for smooth optimization whose internal iterations are stopped when both (1.6) and (1.9) hold. The k-th iteration of our inexact proximal algorithm

1. generates a sequence y^ℓ to solve (1.2) (by some Newton-like algorithm),
2. stops this inner iteration when both (1.6) and (1.9) hold,
3. and then proceeds to the next iteration with $x_{k+1} = y^\ell$.

We therefore have at each iteration

$$\frac{m_1}{2}\,\|g_{k+1}\|\,\|\Delta x_k\| \;\leqslant\; f_k - f_{k+1}\,, \tag{1.10}$$

$$\|\Delta x_k + t_k g_{k+1}\| \;\leqslant\; m_2\,\max\{t_k\|g_{k+1}\|, \|\Delta x_k\|\}\,, \tag{1.11}$$

with the notation

$$f_k := f(x_k)\,, \quad g_k := \nabla f(x_k)\,, \quad \Delta x_k := x_{k+1} - x_k\,.$$

In the next sections, we will prove convergence of the method (section 2) and illustrate it on some test-problems from the CUTEr library (section 3). Before this, we finish this introductive section by observing that the descent test (1.10) plays a minor role in the convex case; a result essentially given in [HS05].

Proposition 1.1 *Assume f is convex. If $m_1^2/4 + m_2^2 \leqslant 1$, then* (1.11) *implies* (1.10). *Besides, the proximal point satisfies not only* (1.3) *but even*

$$f(x_k) - f(p_k) \geqslant \frac{1}{t_k}\|p_k - x_k\|^2\,, \tag{1.12}$$

so that the algorithm is still valid with $0 < m_1 < 2$ in (1.10).

Proof Using the notation

$$\mu := \max\{t_k\|g_{k+1}\|, \|\Delta x_k\|\} \quad \text{and} \quad \nu := \min\{t_k\|g_{k+1}\|, \|\Delta x_k\|\},$$

observe that

$$\mu^2 + \nu^2 = \|t_k g_{k+1}\|^2 + \|\Delta x_k\|^2 \quad \text{and} \quad \mu\nu = \|t_k g_{k+1}\|\,\|\Delta x_k\|\,.$$

Then square (1.11) and expand the lefthand side:

$$\|\Delta x_k\|^2 + 2t_k\langle \Delta x_k, g_{k+1}\rangle + \|t_k g_{k+1}\|^2 \leqslant m_2^2\,\mu^2.$$

Therefore we have

$$\begin{aligned} -2t_k\langle \Delta x_k, g_{k+1}\rangle &\geqslant \|\Delta x_k\|^2 + \|t_k g_{k+1}\|^2 - m_2^2\,\mu^2 \\ &= \big(1 - m_2^2\big)\mu^2 + \nu^2 \\ &\geqslant 2\sqrt{1 - m_2^2}\,\mu\,\nu, \end{aligned}$$

where the last line comes from the expansion of $\left(\sqrt{1 - m_2^2}\,\mu - \nu\right)^2 \geqslant 0$.

Now use the subgradient inequality at x_{k+1}:

$$f(x_k) - f(x_{k+1}) \geqslant -\langle \Delta x_k, g_{k+1}\rangle \geqslant \sqrt{1-m_2^2}\,\|g_{k+1}\|\,\|\Delta x_k\|$$

which yields (1.10) under the stated condition on m_1 and m_2.

Besides, $\tilde{f}$ is now strongly convex, so we can write

$$\frac{1}{2t_k}\|x_k - p_k\|^2 \leqslant \tilde{f}(x_k) - \tilde{f}(p_k) = f(x_k) - f(p_k) - \frac{1}{2t_k}\|x_k - p_k\|^2\,,$$

which is (1.12). Therefore, in (1.3), (1.6), (1.10), we can eliminate the factor 1/2, or take $m_1 < 2$. □

2 Convergence analysis

In this section, we develop the convergence analysis of inexact proximal algorithms satisfying (1.10) and (1.11). The convergence results lie in Theorem 2.5 and Theorem 2.7 below, which deal respectively with the two possible cases: either (x_k) has some cluster point, or $\|x_k\| \to +\infty$. The second case calls for convexity of f.

We start with the following obvious lemma, to get rid of the trivial case and assume in the rest of this section that (f_k) is bounded below.

Lemma 2.1 *If* (1.10) *holds then the sequence* (f_k) *is decreasing. If, moreover* (f_k) *is not bounded from below, then* $\lim_{k\to+\infty} f_k = \inf f = -\infty$.

We now go through some technicalities to establish the key lemma 2.4.

Lemma 2.2 *Let* $0 < m < 1$, *and set* $\gamma = \sqrt{2(1+m^2)}/(1-m^2) > 1$. *Then for all* $u, v \in \mathbb{R}^n$ *such that*

$$\|u+v\| \leqslant m\,\|v\|, \tag{2.1}$$

we have $\|u\| \leqslant \gamma\|v\|$ *and* $\|v\| \leqslant \gamma\|u\|$.

Proof Square (2.1) and expand the lefthand side to get

$$\|u\|^2 + 2\langle u, v\rangle + \|v\|^2 \leqslant m^2\|v\|^2.$$

Note also that $-2\langle u, v\rangle \leqslant \varepsilon\|u\|^2 + \|v\|^2/\varepsilon$, for all $\varepsilon > 0$. Adding the two inequalities gives

$$(1-\varepsilon)\|u\|^2 \leqslant \Big(m^2 - 1 + \frac{1}{\varepsilon}\Big)\|v\|^2. \tag{2.2}$$

Take $\varepsilon = 1/2$ in (2.2) to get $1/2\|u\|^2 \leqslant (m^2+1)\|v\|^2$; that is,

$$\|u\| \leqslant \sqrt{2(m^2+1)}\;\|v\| \leqslant \gamma\,\|v\|.$$

Take now $\varepsilon = 2/(1-m^2)$ in (2.2) to get $-(1+m^2)/(1-m^2)\|u\|^2 \leqslant -(1-m^2)/2\|v\|^2$; in other words,

$$\frac{2(1+m^2)}{(1-m^2)^2}\,\|u\|^2 \geqslant \|v\|^2\,,$$

which is just the second claimed inequality. □

Lemma 2.3 *Let t_k vary in a positive interval:*

$$0 < t_{\min} \leqslant t_k \leqslant t_{\max} < +\infty. \tag{2.3}$$

Then (1.11) *implies the existence of positive α, β such that, for all k,*

$$\frac{1}{\alpha}\,\|g_{k+1}\| \leqslant \|\Delta x_k\| \leqslant \beta \|g_{k+1}\|. \tag{2.4}$$

Proof Take γ of Lemma 2.2 with $m = m_2$. At iteration k, (1.11) means

$$\text{either} \quad \|\Delta x_k + t_k g_{k+1}\| \leqslant m_2 t_k \|g_{k+1}\|, \tag{2.5}$$

$$\text{or} \quad \|\Delta x_k + t_k g_{k+1}\| \leqslant m_2 \|\Delta\, x_k\|. \tag{2.6}$$

If (2.5) holds [resp. (2.6) holds], we apply Lemma 2.2 with $u = \Delta x_k$ and $v = t_k g_{k+1}$ [resp. $u = t_k g_{k+1}$ and $v = \Delta x_k$]. In both cases, we obtain

$$\|\Delta x_k\| \leqslant \gamma t_k \|g_{k+1}\| \quad \text{and} \quad t_k \|g_{k+1}\| \leqslant \gamma \|\Delta x_k\|.$$

This finally gives (2.4) with $\alpha = \gamma / t_{\min}$ and $\beta = \gamma\, t_{\max}$. □

Lemma 2.4 *Let* (1.10) *and* (1.11) *hold, and make assumption* (2.3)*. Then we have for all k*

$$f_0 - f_k \geqslant \frac{m_1}{2\alpha} \sum_{i=0}^{k-1} \|g_{i+1}\|^2, \tag{2.7}$$

$$f_0 - f_k \geqslant m_1 \beta \sum_{i=0}^{k-1} \|\Delta x_i\|^2. \tag{2.8}$$

It follows that, if (f_k) is bounded from below, then the two series

$$\sum_k \|g_k\|^2 \quad \textit{and} \quad \sum_k \|\Delta x_k\|^2$$

are convergent.

Proof The descent property (1.10) implies that for all $i = 0, \ldots, k$

$$f_i - f_{i+1} \geqslant \frac{m_1}{2} \|g_{i+1}\|\, \|\Delta x_i\|.$$

Summing these inequalities gives

$$f_0 - f_k \geqslant \frac{m_1}{2} \sum_{i=0}^{k-1} \|g_{i+1}\|\, \|\Delta x_i\|$$

which, using the two inequalities of Lemma 2.3, yields both (2.7) and (2.8). The rest follows directly. □

Theorem 2.5 (Convergence, bounded case) *Let* (1.10) *and* (1.11) *hold, and make assumption* (2.3)*. If (x_k) has some cluster point $\bar{x}$, then the whole sequence (f_k) converges to $f(\bar{x})$; and continuous differentiability of f implies that $\bar{x}$ is stationary. In particular, if f is inf-compact, then (f_k) converges to some stationary value (which is minimal if f is convex), while any cluster point of (x_k) is stationary.*

Proof Let $\bar{x}$ be a cluster point of (x_k). The first statement comes from monotonicity of the sequence (f_k) (Lemma 2.1). If $(x_{k'})$ is a subsequence converging to $\bar{x}$, $\|g_{k'}\| \to 0$ (Lemma 2.4) and therefore $\nabla f(\bar{x}) = 0$ by continuity of ∇f. The rest follows because $\bar{x}$ is arbitrary, and certainly exists if f is inf-compact. □

Note that convergence of an inexact proximal method satisfying just (1.11) is established in [HS05, Theorem 1] for a smooth *convex* function. When there exists a cluster point to the sequence of iterates, the above result shows that convexity can be replaced by the descent-test (1.10).

The case where no cluster point exists is more delicate, since no stationary point shows up. The only relevant property for (x_k) is then to be a minimizing sequence; but establishing such a property is hopeless without convexity. Besides, to have $\nabla f(x_k) \to 0$ is not enough for (x_k) to be a minimizing sequence: some additional property is required, for example the following one, due to M.J. Todd.

Lemma 2.6 ([Tod89, Prop. 2.2]) *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be convex (and continuously differentiable). If a sequence (x_k) of points in $\mathbb{R}^n$ satisfies the following four properties: (f_k) is a decreasing sequence (bounded from below), $g_k \to 0$, $\|x_k\| \to \infty$ and*

$$\liminf_{k\to\infty} \|g_k\|\,\|x_k\| < \infty,$$

then the whole sequence (f_k) converges to $\inf f$.

Now, when f is convex, Proposition 1.1 gives that (1.10) is superfluous. So we have the general convergence result involving only (1.11).

Theorem 2.7 *Let (1.11) hold and make assumption (2.3). If f is convex (and continuously differentiable) then $\lim_{k\to+\infty} f_k = \inf f$.*

Proof First, in view Lemma 2.1, the we have only to consider the case when $(f_k)_k$ is bounded below. Second, in view of Theorem 2.5, we have only to consider the case $\|x_k\| \to +\infty$. To use Lemma 2.6, we look for an estimation of $\|x_k\|\,\|g_k\|$. Using first the norm $\|\cdot\|_1$, we write

$$\|x_k\|_1 = \left\| x_0 + \sum_{i=0}^{k-1} \Delta x_i \right\|_1 \leqslant \sum_{i=0}^{k-1} \|\Delta x_i\|_1 + \|x_0\|_1 .$$

The Cauchy-Schwarz inequality in $\mathbb{R}^k$ gives

$$\sum_{i=0}^{k-1} \|\Delta x_i\|_1 \leqslant \sqrt{k}\sqrt{\sum_{i=0}^{k-1} {\|\Delta x_i\|_1}^2},$$

so we get for all k

$$\|x_k\|_1 \leqslant \sqrt{k}\sqrt{\sum_{i=0}^{k-1} {\|\Delta x_i\|_1}^2} + \|x_0\|_1 . \tag{2.9}$$

Now we combine the property $\|z\|_1 \leqslant \sqrt{n}\|z\|$ with (2.8) to obtain

$$f_0 - f_k \geqslant \frac{m_1\beta}{n} \sum_{i=0}^{k-1} \|\Delta x_i\|_1^2 .$$

Since f is bounded below, this yields that the series $\sum \|\Delta x_i\|_1^2$ converges. Use this property in (2.9): for some $M > 0$,

$$\|x_k\| \;\leqslant\; \|x_k\|_1 \;\leqslant\; M\sqrt{k} + \|x_0\|_1 \,,$$

and then

$$\|x_k\| \, \|g_k\| \leqslant M\sqrt{k}\|g_k\| + \|x_0\|_1 \, \|g_k\| \,. \tag{2.10}$$

Now we claim that $\liminf \sqrt{k}\|g_k\| = 0$. If it were not the case, we would have a $\kappa > 0$ such that $\sqrt{k}\|g_k\| \geqslant \kappa$ for all k. This would contradict Lemma 2.4 showing that the serie $\sum \|g_k\|^2$ converges (since the serie $\sum 1/k$ diverges).

Together with (2.10), this implies $\liminf \|x_k\| \, \|g_k\| = 0$. We also know that (f_k) is decreasing (Lemma 2.1), (g_k) tends to 0 by (2.7) and $\|x_k\|$ tends to $+\infty$ by assumption; each assumption of Lemma 2.6 is therefore satisfied. □

The conclusion of this section is that the two tests (1.10) and (1.11) are consistent as stopping conditions for inner minimization in inexact proximal algorithms for smooth minimization. We illustrate their reliability in the next section.

3 Numerical illustrations

This section presents numerical illustrations of the proximal scheme using the two inner stopping rules (1.6) and (1.9), applied to smooth optimization. We consider some test-problems from the CUTEr library [BCGT95], with smooth objective function and no constraint, for which classical minimization algorithms run into numerical difficulties because of some degeneracy (for example, the objective function may have a degenerate Hessian at some iteration).

In addition to the inner stopping rule, implementing the proximal algorithm requires an algorithm to solve the inner problem and an updating method for the proximal parameter. These two ingredients are described in the next two subsections.

3.1 Solving the internal problem

Among the various possibilities to minimize smooth functions, Newton-like methods (quasi-Newton, exact or truncated Newton) are known to be most efficient in general. The choice of a particular one depends on the available information about the objective function: we consider here problems where the gradient is available under an analytic form and we use a quasi-Newton method. More precisely, we use the software `m1qn3`[1] implementing a limited-memory BFGS update with Wolfe line-search [GL89].

Recall that this update consists in "simulating" the BFGS formula, without storing any $n \times n$ matrix. Specifically, let y^ℓ be the current iterate aimed at solving (1.2). Using the notation $g := \nabla f$, the gradient of $\tilde{f}_k$ at y^ℓ is $\tilde{g}_k^\ell = g^\ell + (y^\ell - x_k)/t_k$. The algorithm stores m differences of points and of gradients

$$\left.\begin{array}{l} \eta^i := y^i - y^{i-1} \\ \gamma_k^i := \tilde{g}_k^i - \tilde{g}_k^{i-1} \end{array}\right\} \quad i = \ell, \ell-1, \ldots, \ell-m+1 \,. \tag{3.1}$$

[1] `http://www-rocq.inria.fr/~gilbert/modulopt`

Appropriate formulae [Noc80] then allow the computation of the search-direction $d^\ell = -H^\ell g^\ell$, where H^ℓ results from m BFGS updates of some "initial" matrix – which is actually re-computed at each iteration ℓ. All of our experiments use $m = 20$, so that `m1qn3` needs to store 40 vectors of dimension n to run.

In our implementation, each new prox computation is warm-restarted, using the pairs (η^i, γ^i) from the previous cycle. In fact, observe from (3.1) that

$$\gamma_k^i = g^i - g^{i-1} + \frac{y^i - y^{i-1}}{t_k}$$

does not depend on the prox center x_k. On the other hand, when t_k is changed to t_{k+1}, no η^i is changed, while the differences of gradients are changed to

$$\gamma_{k+1}^i = \gamma_k^i + \Big(\frac{1}{t_{k+1}} - \frac{1}{t_k}\Big)\eta^i \,. \tag{3.2}$$

Easy updates of the working arrays of `m1qn3` thus allow the use of old pairs to restart the next internal algorithm, thereby ensuring persistence of available information.

3.2 Updating the proximal parameter

The prox parameter t_k is delicate to tune and serious studies on this question seem to be lacking in the literature. We have conducted some experiments with the updating rule of [LS97], they were not too convincing (note also that this rule needs convexity of f). Here we propose for simplicity to start with a fixed value $t_0 = 10^4$, and then to use a rough heuristic, based on how easily (1.2) can be solved:

1. If the number ℓ_k of internal iterations needed to reach (1.6) and (1.9) exceeds some threshold $\bar{\ell}$, more regularization seems to be needed: we decrease t_k.
2. By contrast, if ℓ_k is lower than some $\underline{\ell}$, we increase t_k to make (1.2) closer to (1.1).
3. In between ($\underline{\ell} \leqslant \ell \leqslant \bar{\ell}$), we leave t_k as it is.

Besides, `m1qn3` may fail to solving (1.2) for some reason:

- No descent can be obtained at the current iteration ℓ.
- Some scalar product $\langle \gamma_k^i, \eta^i \rangle$ is not positive. Recall that the property $\langle \gamma_k^i, \eta^i \rangle > 0$ is crucial for BFGS updates and is guaranteed by Wolfe's line-search; however, observe on (3.2) that $t_{k+1} > t_k$ may result in some negative $\langle \gamma_{k+1}^i, \eta^i \rangle$.
- The number of inner iterations reaches a maximal number allowed (fixed at 1500).

In these pathological cases, we decrease t_k as well; but we keep the same x_k, since otherwise the theory of Section 2 would not apply ((1.10), (1.11) need not hold).

Figure 3.1 summarizes our implementation; here are some comments.

- We fix $m_1 = 0.1$ in (1.6) and $m_2 = 0.9$ in (1.9); these rather extreme values aim at a quick exit of the inner algorithm.
- The overall stopping test is

$$\|\nabla f(x_k)\| \leqslant \varepsilon \quad \text{with } \varepsilon = 10^{-6} \,. \tag{3.3}$$

- Note that `m1qn3` may terminate with an optimal y^ℓ; then (1.6) and (1.9) *have to* be satisfied.
- The bounds on internal iterations are $\underline{\ell} = 0.03n$ and $\bar{\ell} = 0.07n$.
- The initial prox parameter is $t_0 = 10^4$. To increase [decrease] t_k, we multiply [divide] it by $\rho = 5$.

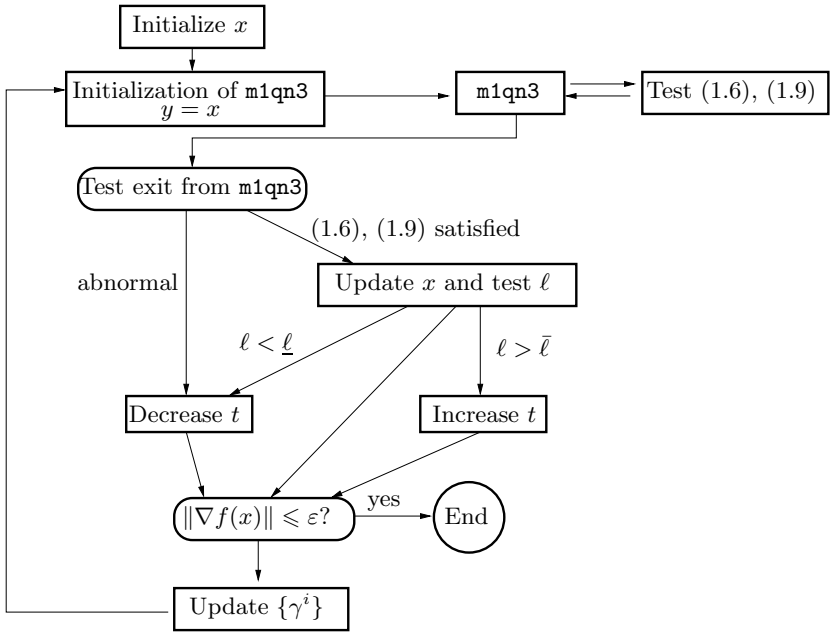

Fig. 3.1 Flow-chart of the inexact proximal algorithm

3.3 Comparative results

Our experiments illustrate the plain use of `m1qn3` and its embedding within our proximal mechanism, on a collection of unconstrained optimization problems from the CUTEr library.

name	n	# prox	# sim
DJTL	2	7	3202
BROWNDEN	4	2	40
TOINTGOR	50	5	207
SENSORS	100	3	73
NCB20	210	9	347
BDQRTIC	1000	2	50
CRAGGLVY	2000	5	248
FREUROTH	5000	3	93
BROYND7D	5000	5	1732
SINQUAD	5000	2	43
SCHMVETT	5000	11	349

Table 3.1 CUTEr test-problems for which quasi-Newton fails

Table 3.1 reports the results of the inexact proximal algorithm on a serie of instances where the pure `m1qn3` fails to reaching (3.3). More precisely, for 27 instances of CUTEr, the line-search fails to obtain a descent step at some iteration. Among these 27 instances, the 11 instances of Table 3.1 are solved by our proximal scheme (using fixed

parameters). Note also that most of the other instances are solved by our algorithm with ad hoc tunings of parameters.

The last two columns of Table 3.1 give the number of external iterations (# prox) and the total number of f, g computations (called # sim, as each such computation usually corresponds to the simulation of some physical system; note that `m1qn3` computes systematically $\nabla\tilde{f}$ together with $\tilde{f}$).

name	n	# prox	# sim	# sim QN
SPARSINE	1000	3	3175	3090
SPARSINE	2000	7	7639	7649
NONDQUAR	1000	7	469	654
NONDQUAR	500	12	558	544
EIGENALS	420	7	401	262
EIGENBLS	420	109	6473	4256
EIGENCLS	462	7	1801	1717
NCB20	510	4	421	383

Table 3.2 Proximal and pure quasi-Newton on the instances of [HZ08]

We have also used the instances selected by [HZ08]; Table 3.2 shows the results. These problems can be solved by the plain version of `m1qn3`, so Table 3.2 reports also # sim for QN in its last column. For homogeneity with [HZ08], the stopping test is now $\|\nabla f(x_k)\|_\infty \leqslant 10^{-6}$.

This table suggests that the pure and proximal versions have roughly similar behaviours on these problems. Note, however, that adjusting the various parameters appearing in our implementation (t_0, ρ, $\underline{\ell}$, $\bar{\ell}$) may entail different results. For example, taking $t_0 = 10^6$ on EIGENALS [resp. $t_0 = 10^8$ on EIGENBLS] reduces # sim from 401 to 311 [resp. from 6473 to 4267].

To close this section, we mention that our results in Table 3.2 are consistently better than those of [HZ08]; but this is normal: the latter uses conjugate gradient, known to be much less efficient than quasi-Newton.

4 Conclusions, perspectives

The contribution of this paper is a proposal to stop the inner iterations in a proximal algorithm, based on the decrease of the true objective function. We use the two tests (1.6) and (1.9), prove their consistency and illustrate them on standard problems.

Our experiments indicate a capacity of the proximal mechanism to improve a quasi-Newton algorithm: it permits to solve degenerate problems on which a direct use of the algorithm fails (Table 3.1), while there is no significant slow-down for problems on which the algorithm succeeds (Table 3.2).

These observations, however, should not be over-interpreted.

- First of all, our experiments involve relatively few instances of CUTEr, neglecting many others where a mere quasi-Newton implementation behaves correctly. Introducing a proximal term would be pointless, then.
- It should be kept in mind that the proximal algorithm is motivated by *reliability*; it is by no means aimed at being brilliant. For example, finite convergence for a quadratic f seems hard to reach.

– Actually, a proximal algorithm without a convincing rule to monitor t_k can hardly be called stable. Ours (section 3.2) is far from that; definite conclusions cannot be drawn until this question is fixed.
– Our crude use of `m1qn3` as a black box for the inner iterations might be subject to improvements. A relevant question is indeed: can one taylor Newton-like methods to take advantage of the special structure in (1.2) and of its iterative nature? This question is close to: can one devise a finitely convergent proximal algorithm to minimize a convex quadratic function? These sorts of questions might be of interest for future research.

In our opinion, the descent test (1.6) is well motivated; recall in particular some positive aspects mentioned in section 1.2. By contrast, the motivation for (1.9) does not seem to be as strong, in a context of descent iteration. By requiring a "small" $\nabla \tilde{f}_k$, it somehow goes against our initial motivation, which was to overlook closedness to the proximal point. Thus, the second test might perhaps be improved, to adequately complement the descent property. The fact that (1.9) implies (1.6) in the convex case might indicate that it is too stringent.

References

[Arm66] L. Armijo. Minimization of functions having Lipschitz continuous first partial deri vatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[BCGT95] I. Bongartz, A. R. Conn, N. Gould, and Ph. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *Transactions of the ACM on Mathematical Software*, 21(1):123–260, 1995.

[BKL66] R.E. Bellman, R.E. Kalaba, and J. Lockett. *Numerical Inversion of the Laplace Transform*, pages 143–144. Elsevier, New York, 1966.

[CL93] R. Correa and C. Lemaréchal. Convergence of some algorithms for convex minimization. *Mathematical Programming*, 62:261–275, 1993.

[DS83] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, 1983.

[GL89] J. C. Gilbert and C. Lemaréchal. Some numercial experiments with variable-storage quasi-Newton algorithms. *Mathematical Programming*, 45:407–435, 1989.

[GP67] A.A. Goldstein and J.F. Price. An effective algorithm for minimization. *Num. Math.*, 10:184–189, 1967.

[HS05] C. Humes and P.J.S. Silva. Inexact proximal point algorithms and descent methods in optimization. *Optimization and Engineering*, 6:257–271, 2005.

[HZ08] W. W. Hager and H. Zhang. Self-adaptive inexact proximal point methods. *Computational Optimization and Applications*, 39(2):161–181, 2008.

[Lev44] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[LS97] C. Lemaréchal and C. Sagastizábal. Variable metric bundle methods: from conceptual to implementable forms. *Mathematical Programming*, 76(3):393–410, 1997.

[Mar63] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963.

[Mar70] B. Martinet. Régularisation d'inéquations variationelles par approximation successives. *Revue Française d'Informatique et de Recherche Opérationelle*, R3:154–158, 1970.

[Mor83] J.J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming, the State of the Art*, pages 258–287. Springer-Verlag, Berlin, 1983.

[MR09] P. Maréchal and A. Rondepierre. A proximal approach to the inversion of ill-conditioned matrices. *C. R. Acad. Sci. Paris*, 347(23-24):1435–1438, 2009.

[Noc80] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

[QS93] L. Qi and J. Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, 1993.

[Roc76] R.T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1:97–116, 1976.

[SS99] M. V. Solodov and B. F. Svaiter. A hybrid projection-proximal point algorithm. *Journal of Convex Analysis*, 6(1):059–070, 1999.

[SS01] M. V. Solodov and B. F. Svaiter. A unified framework for some inexact proximal point algorithms. *Numerical functional analysis and optimization*, 22:1013–1035, 2001.

[Tod89] M. Todd. On Convergence Properties of Algorithms for Unconstrained Minimization. *IMA journal of Numerical Analysis*, 9:435–441, 1989.

[Wol69] P. Wolfe. Convergence conditions for ascent methods. *SIAM Rev.*, 11:226–235, 1969.