RUTCOR
RESEARCH
REPORT

Proximal Methods for Nonlinear Programming: Double Regularization and Inexact Subproblems

Jonathan Eckstein [a] Paulo J. S. Silva [b]

RRR 17-2008,

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]Department of Management Science and Information Systems and RUTCOR, 640 Bartholomew Road, Busch Campus, Rutgers University, Piscataway NJ 08854 USA, jeckstei@rci.rutgers.edu.

[b]Department of Computer Science, University of São Paulo, Brazil. pjssilva@ime.usp.br

Rutcor Research Report
RRR 17-2008,

Proximal Methods for Nonlinear Programming: Double Regularization and Inexact Subproblems

Jonathan Eckstein Paulo J. S. Silva

Abstract. This paper describes the first phase of a project attempting to construct an efficient general-purpose nonlinear optimizer using an augmented Lagrangian outer loop with a relative error criterion, and an inner loop employing a state-of-the art conjugate gradient solver. The outer loop can also employ double regularized proximal kernels, a fairly recent theoretical development that leads to fully smooth subproblems. We first enhance the existing theory to show that our approach is globally convergent in both the primal and dual spaces when applied to convex problems. We then present an extensive computational evaluation using the CUTE test set, showing that some aspects of our approach are promising, but some are not. These conclusions in turn lead to addditional computational experiments suggesting where to next focus our theoretical and computational efforts.

Acknowledgements: Jonathan Eckstein's work was partially supported by Rutgers Business School Research Resources Committee grants. Paulo J. S. Silva carried out part of this research while visiting RUTCOR and IMECC-UNICAMP. He was upported by CNPq (grant 303030/2007-0), FAPESP (grant 2008/03823-0), and PRONEX–Optimization.

1 Introduction and motivation

Over the past 15 years, there has been significant theoretical activity in the field of proximal point algorithms for convex programming and monotone variational inequalities (see [24, 25] and references therein). Two developments of particular note are

- A much more satisfactory theory of nonquadratic regularizations and corresponding augmented Lagrangian methods: early work in this area include [9, 31, 12], and recent developments have focused on "double" regularizations [5, 4, 26] that combine the Fejér monotonicity of the classic quadratic regularization with the barrier and smoothing properties of coercive regularizations. For inequality constraints, the augmented Lagrangians corresponding to these methods retain the full level of smoothness of the given problem, unlike the classic quadratic penalty of [24].

- Improved understanding of approximate solution of subproblems. In particular, works such as [28, 29, 30, 8] have explored "relative" error criteria that allow for very loose solution of early subproblems and a constructive technique for tightening subproblem accuracy.

On the other hand, attempts to apply these ideas toward the development of general-purpose, high quality solvers have been rare: we are aware of two examples, [7] for nonlinear optimization and [26] for complementarity problems. Both study the use of nonquadratic kernels, although the different choice of problem domains led to rather different conclusions. Moreover, we know of only one application of a relative error criterion to a general-purpose solver for nonlinear optimization [17]. This study was limited to the quadratic penalty and presented only simplistic numerical experiments.

This paper describes the current status of a project attempting to capitalize on both nonquadratic kernels and relative error criteria, in conjunction with recent advances is nonlinear unconstrained and box-constrained optimizers, to create a practical class of nonlinear optimization algorithms having different properties from the popular and currently dominant Newton barrier methods. Our motivation was to find methods that do not require construction of Hessian matrices or the highly accurate solution of systems of linear equations, particularly by direct methods, at every iteration. While Newton barrier methods have many attractive properties and well deserve their current popularity, these features make them difficult to apply in some settings. Our plan was as follows:

- Use a "double-regularized" proximal method, applied to a primal-dual formulation of the problem, as the main loop of the method. Our work on complementarity problems in [26] suggested this would be a robust and relatively efficient approach.

- Instead of solving proximal subproblems exactly, use a form of relative error criterion inspired by [28, 29, 30, 8]. With such a criterion, effort would not be wasted obtaining highly accurate solutions to subproblems occurring early in a run, far from the eventual solution.

- When solving subproblems, take advantage of the most recent advances in unconstrained nonlinear conjugate gradient methods, as embodied in work like [15, 16].

A method constructed in this manner could solve problems with only function evaluations, gradient evaluations, and very elementary linear algebra operations. With reasonably competitive performance, these features would make it a desirable alternative to more established methods in some situations.

The remainder of this paper lays out the theoretical foundations of our proposed method, and the results of our first round of computational experiments. Obtaining global convergence for convex problems while combining double regularizations with a relative error criterion requires extensions to existing theoretical results for general maximal monotone operators, which are outlined in Section 2. Next, Section 3 shows how to apply these results to general convex programming problems. With this foundation, Section 4 describes a prototype nonlinear optimizer built on the foundation provided by Sections 2 and 3, and our computational experience with it, using (mostly nonconvex) standard test problems from the CUTE test set.

The results in Section 4 confirm some of our expectations about our approach, but suggest reconsideration of others. In particular, it appears, in accordance with [7] — at least in the context of optimization, rather than complementarity — that the flurry of recent interest in coercive proximal methods may have been misplaced, and it may well be sufficient to concentrate on the classical quadratic regularization. On the other hand, it appears that relative error criteria, which have not been the subject of prior rigorous experiments, can be very beneficial. Further results suggest where we should focus continued theoretical and experimental effort.

2 Theoretical foundation using monotone operators

Consider solving the inclusion

$$0 \in T(z) + N_B(z), \tag{1}$$

where T is a possibly set-valued maximal monotone operator on $\mathbb{R}^n$ and B is a box set of the form

$$B = \{z \in \mathbb{R}^n \mid a \leq x \leq b\},$$

for $a \in [-\infty, +\infty)$ and $b \in (\infty, +\infty]$, $a \leq b$. Often, we simply choose $a = 0$ and $b = +\infty$, obtaining B equal to the nonnegative orthant $\mathbb{R}^n_+ = \{z \in \mathbb{R}^n \mid z \geq 0\}$. By a particular choice of T, we will apply (1) to general convex programming problems.

We propose an inexact proximal point method with extragradient step, inspired by the work of Solodov and Svaiter [30] and Burachik and Svaiter [8]. The method may been seen as an extension of the work of Burachik and Svaiter from the context of second-order homogeneous kernels to general double regularizations, as defined in [26]. The general double regularization class includes not only regularizations derived from φ-divergences, but

also those obtained from Bregman distances. We use a similar notation to [26]: a *double regularization* for the box B is a separable function $\widetilde{D} : B \times \operatorname{int} B \to \mathbb{R}$ of the form

$$\widetilde{D}(x,y) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \widetilde{d}_i(x_i,y_i) = \sum_{i=1}^{n} d_i(x_i,y_i) + \frac{\mu}{2}(x_i-y_i)^2, \tag{2}$$

where the individual $\widetilde{d}_i$, $i = 1,\ldots,n$, which we call *double regularization components*, conform to Assumptions 2.1.1–2.1.3 of [26]. In particular, μ is scalar and greater than 1. Moreover, we assume that the individual terms d_i conform to Assumption 2.3 in [26]. The distance-like measure $\widetilde{D}(x,y)$ is the sum of two terms: the first, $\sum_{i=1}^{n} d_i(x_i,y_i)$, is *coercive*, meaning that its derivative becomes infinite as x approaches the boundary of B; the second is a scalar multiple of the classic squared Euclidean distance. The following is a simplified version of Lemma 3.4 of [26]:

Lemma 2.1 *Suppose $\widetilde{D}$ be a double regularization for the box B with regularization factor $\mu \geq 1$. For any $z \in B$ and $x, y \in \operatorname{int} B$, we have*

$$\langle z-x, \nabla_1 \widetilde{D}(x,y)\rangle \leq \frac{\mu+1}{2}\left(\|z-y\|^2 - \|z-x\|^2\right) + \frac{1-\mu}{2}\|x-y\|^2.$$

Proof. Letting $D(x,y) \stackrel{\text{def}}{=} \sum_{i=1}^{n} d_i(x_i,y_i)$, we observe that

$$\begin{aligned}
\langle z-x, \nabla_1 \widetilde{D}(x,y)\rangle &= \langle z-x, \nabla_1 D(x,y)\rangle + \mu\langle z-x, x-y\rangle \\
&\leq \langle z-y, x-y\rangle + \mu\langle z-x, x-y\rangle \qquad \text{[Lemma 3.3 in [26]]} \\
&= \frac{1}{2}\big(\|x-y\|^2 - \|x-z\|^2 + \|y-z\|^2\big) \\
&\quad - \frac{\mu}{2}\big(\|y-x\|^2 - \|y-z\|^2 + \|x-z\|^2\big) \\
&= \frac{\mu+1}{2}\big(\|z-y\|^2 - \|z-x\|^2\big) + \frac{1-\mu}{2}\|x-y\|^2.
\end{aligned}$$

□

We propose the following algorithm:

Algorithm 2.2 Inexact Proximal Method based on a Double Regularization (IPMDR): *Let $\widetilde{D}$ be a double regularization of the box B with regularization factor $\mu > 1$.*

1. **Initialization:** *Let $k = 0$. Choose scalars $\sigma \in (0,1)$ and $\zeta, c > 0$, as well as an initial iterate $z^0 \in \operatorname{int} B$.*

2. **Iteration:**

   (a) *Choose $\alpha_k \in [c, +\infty)$.*

(b) With the definition

$$\hat{z}(z^k, \widetilde{v}^k) \stackrel{\text{def}}{=} \left(\nabla_1 \widetilde{D}(\cdot, z^k)\right)^{-1}(-\alpha_k \widetilde{v}^k), \tag{3}$$

find $(\widetilde{z}^k, \widetilde{v}^k) \in \operatorname{gph} T$ *such that*

$$\langle \hat{z}(z^k, \widetilde{v}^k) - \widetilde{z}^k, \alpha_k \widetilde{v}^k \rangle \geq \frac{\sigma(1-\mu)}{2} \left\| \hat{z}(z^k, \widetilde{v}^k) - z^k \right\|^2 \tag{4}$$

$$\left\| \hat{z}(z^k, \widetilde{v}^k) - \widetilde{z}^k \right\| \leq \zeta \left\| \hat{z}(z^k, \widetilde{v}^k) - z^k \right\|. \tag{5}$$

(c) Let $z^{k+1} = \hat{z}(z^k, \widetilde{v}^k)$, $k \leftarrow k+1$, *and repeat.* □

The error criteria (4) and (5) are carefully chosen to lead to the following result:

Lemma 2.3 *The sequence* $\{z^k\}$ *computed by the IPMDR is Fejér monotone with respect to the set of zeroes of* $T + N_B$, *that is,* $\{\|z^k - z^*\|\}$ *is a nonincreasing sequence for every* z^* *such that* $T(z^*) + N_B(z^*) \ni 0$. *Specifically, for every* $k \geq 0$ *and* z^* *with* $0 \in T(z^*) + N_B(z^*)$,

$$(\mu+1)\left\|z^{k+1} - z^*\right\|^2 \leq (\mu+1)\left\|z^k - z^*\right\|^2 - (\mu-1)(1-\sigma)\left\|z^{k+1} - z^k\right\|^2.$$

In particular, if the set of zeroes $(T + N_B)^{-1}(0)$ *is nonempty,* $\left\|z^{k+1} - z^k\right\| \to 0$.

Proof. Using Lemma 2.1 with $z = z^*$, $x = z^{k+1}$ and $y = z^k$, we have that

$$\langle z^* - z^{k+1}, \nabla_1 \widetilde{D}(z^{k+1}, z^k) \rangle \leq \frac{\mu+1}{2}\left(\|z^* - z^k\|^2 - \|z^* - z^{k+1}\|^2\right) + \frac{1-\mu}{2}\|z^{k+1} - z^k\|^2. \tag{6}$$

To bound the left-hand side of (6), we note that

$$\begin{aligned}
\langle z^* - z^{k+1}, \nabla_1 \widetilde{D}(z^{k+1}, z^k) \rangle &= \langle z^* - \widetilde{z}^k + \widetilde{z}^k - z^{k+1}, \nabla_1 \widetilde{D}(z^{k+1}, z^k) \rangle \\
&= \langle z^* - \widetilde{z}^k + \widetilde{z}^k - z^{k+1}, -\alpha_k \widetilde{v}^k \rangle \qquad (7) \\
&= \langle z^* - \widetilde{z}^k, -\alpha_k \widetilde{v}^k \rangle + \langle z^{k+1} - \widetilde{z}^k, \alpha_k \widetilde{v}^k \rangle \\
&\geq \langle z^{k+1} - \widetilde{z}^k, \alpha_k \widetilde{v}^k \rangle. \qquad (8)
\end{aligned}$$

Above, (7) follows from (3) and $z^{k+1} = \hat{z}(z^k, \widetilde{v}^k)$. We note that $\widetilde{v}^k \in T(\widetilde{z}^k)$ by construction, and $0 \in T(z^*) + N_B(z^*)$ by assumption. Since $\widetilde{z}^k \in B$, we have $0 \in N_B(\widetilde{z}^k)$, and $\widetilde{v}^k \in T(\widetilde{z}^k) \subseteq T(\widetilde{z}^k) + N_B(\widetilde{z}^k)$. So, the monotonicity of the operator $T + N_B$ implies $\langle z^* - \widetilde{z}^k, -\alpha_k \widetilde{v}^k \rangle \geq 0$, and we obtain (8). Combining (8) with the error criterion (4) and $z^{k+1} = \hat{z}(z^k, \widetilde{v}^k)$ yields

$$\langle z^* - z^{k+1}, \nabla_1 \widetilde{D}(z^{k+1}, z^k) \rangle \geq \frac{\sigma(1-\mu)}{2}\|z^{k+1} - z^k\|^2.$$

Combining this inequality with (6) completes the proof. □

Lemma 2.4 *Suppose that the set of zeroes of $T + N_B$ is non-empty. Let $\bar{z} \in \mathbb{R}^n$ be a limit point of the IPMDR $\{z^k\}$,* i.e., *$z^k \to_{\mathcal{K}} \bar{z}$ for some infinite set $\mathcal{K} \subseteq \mathbb{N}$. Then for $i = 1, \ldots, n$,*

$$\begin{array}{rcll} \lim\limits_{k \to_{\mathcal{K}} \infty} \widetilde{v}_i^k & = & 0 & \text{if } \bar{z}_i \in (a_i, b_i) \\ \liminf\limits_{k \to_{\mathcal{K}} \infty} \widetilde{v}_i^k & \geq & 0 & \text{if } \bar{z}_i = a_i \\ \limsup\limits_{k \to_{\mathcal{K}} \infty} \widetilde{v}_i^k & \leq & 0 & \text{if } \bar{z}_i = b_i. \end{array} \tag{9}$$

Proof. First, note that the conclusions of this lemma are identical to [27, Lemma 2.4]; however, since the error criterion is different, a new proof is required.

Recalling the definition of z^{k+1}, we know that $\alpha_k \widetilde{v}^k = -\nabla_1 \widetilde{D}(z^{k+1}, z^k)$. Moreover, Lemma 2.3 implies that $z^{k+1} \to \bar{z}$ too.

For each coordinate i, we consider the three possible cases:

1. $\bar{z}_i \in (a_i, b_i)$. Here, we use the fact that $\widetilde{d}_i$ is continuously differentiable in the interval (a_i, b_i) to see that $-\alpha_k \widetilde{v}_i^k \to_{\mathcal{K}} \widetilde{d}_i(\bar{z}_i, \bar{z}_i) = 0$. As α_k is positive and bounded away from zero, we have $\widetilde{v}_i^k \to_{\mathcal{K}} 0$.

2. $\bar{z}_i = a_i$. Suppose, for the sake of a contradiction, that there is an $\epsilon > 0$ and an infinite index set $\mathcal{K}' \subset \mathcal{K}$ such that for $k \in \mathcal{K}'$, $\widetilde{v}_i^k \leq -\epsilon$. This says that $\widetilde{d}_i(z_i^{k+1}, z_i^k) \geq \alpha_k \epsilon \geq c\epsilon$, where c is the lower bound to α_k. As $\widetilde{d}_i(\cdot, z_i^k)$ attains its minimum at z_i^k and is convex, this implies that $z_i^{k+1} > z_i^k$. Hence, Lemma 2.4 in [26] implies that for all sufficiently large $k \in \mathcal{K}'$, one has $|\widetilde{d}_i'(z_i^{k+1}, z_i^k)| \leq (2 + \mu)|z_i^{k+1} - z_i^k| \to_{\mathcal{K}'} 0$, a contradiction.

3. $\bar{z}_i = b_i$. Analogous to the previous case. □

Now we can also mimic [27, Lemma 2.5].

Lemma 2.5 *Suppose that the set of zeroes of $T + N_B$ is non-empty. Then the sequence $\{\widetilde{v}^k\}$ is bounded.*

Proof. Let $\hat{z} \in \operatorname{dom} T \cap \operatorname{int} B$, which must be nonempty in order for the method to be well defined. Let $v \in T(\hat{z})$. Monotonicity implies that

$$\langle \widetilde{z}^k - \hat{z}, \widetilde{v}^k - \hat{v} \rangle \geq 0,$$

for all k. We will show that unboundedness of $\widetilde{v}^k$ would contradict this inequality.

Suppose, for the sake of contradiction that there is an infinite index set $\mathcal{K}$ such that $\{\widetilde{v}^k\}_{\mathcal{K}}$ converges in $[-\infty, +\infty]$, with at least one $\{\widetilde{v}_i^k\}_{\mathcal{K}}$ unbounded. Without loss of generality, as $\{z^k\}$ is bounded, we can assume it converges on this index set to a point $\bar{z}$. Using Lemma 2.3, we see that $z^{k+1} \to_{\mathcal{K}} \bar{z}$ and hence, due to (5), $\widetilde{z}^k \to_{\mathcal{K}} \bar{z}$ too.

Lemma 2.4 implies that for each unbounded coordinate i, either

$$\widetilde{v}_i^k \to_{\mathcal{K}} +\infty \text{ and } \bar{z}_i = a_i$$

or

$$\widetilde{v}_i^k \to_{\mathcal{K}} -\infty \text{ and } \bar{z}_i = b_i.$$

Hence, for each unbounded coordinate i we would have

$$(\widetilde{z}_i^k - \hat{z}_i)(\widetilde{v}_i^k - \hat{v}_i^k) \to_{\mathcal{K}} (a_i - \hat{z}_i)(+\infty) = -\infty$$
or
$$(\widetilde{z}_i^k - \hat{z}_i)(\widetilde{v}_i^k - \hat{v}_i^k) \to_{\mathcal{K}} (b_i - \hat{z}_i)(-\infty) = -\infty.$$

We conclude that

$$\langle \widetilde{z}^k - \hat{z}, \widetilde{v}^k - \hat{v} \rangle \to_{\mathcal{K}} -\infty,$$

a contradiction. □

Theorem 2.6 *Suppose that the set of zeroes of $T + N_B$ is nonempty. Then, the sequence $\{z^k\}$ computed by the IPMDR converges to some $\bar{z} \in (T + N_B)^{-1}(0)$.*

Proof. Lemma 2.3 already shows that $\{z^k\}$ is Fejér monotone with respect to the set of zeroes. All we need to show is that it has a zero as limit point. Let $z^k \to_{\mathcal{K}} \bar{z}$ for some infinite index set $\mathcal{K}$. The same Lemma shows that $z^{k+1} \to_{\mathcal{K}} \bar{z}$ and hence (5) implies that $\widetilde{z}^k \to_{\mathcal{K}} \bar{z}$. Lemma 2.5 shows then that $\widetilde{v}^k \in T(\widetilde{z}^k)$ is also bounded and Lemma 2.4 shows that the limits of $\{\widetilde{v}^k\}$ have the correct sign structure to imply that $\bar{z}$ is a zero of $T + N_B$. □

When $B = \mathbb{R}^n$, the assumptions concerning double regularizations permit one to choose $D(x, y) = (1/2)\|x - y\|^2$, in which case $\widetilde{D}(x, y)$ is a scalar multiple of $\|x - y\|^2$ In this case, it may appear that the IPMDR does not directly generalize the classical proximal point algorithm using the squared Euclidean norm, due to the presence of the constant μ in the approximation criterion (4). However, with a little effort, one can see that this is not the case. Consider the following variation of the IPMDR using the classical regularization:

Algorithm 2.7 Inexact Classic Proximal Method (ICPM)

1. **Initialization:** *Let $k = 0$. Choose scalars $\sigma \in (0, 1)$ and $\zeta, c > 0$, as well as an initial iterate $x^0 \in \operatorname{int} B$.*

2. **Iteration:**

   (a) *Choose $\alpha_k \in [c, +\infty)$.*

   (b) *With the definition*

   $$\hat{z}(z^k, \widetilde{v}^k) \stackrel{\text{def}}{=} z^k - \alpha_k \widetilde{v}^k, \tag{10}$$

   find $(\widetilde{z}^k, \widetilde{v}^k) \in \operatorname{gph} T$ such that

   $$\langle \hat{z}(z^k, \widetilde{v}^k) - \widetilde{z}^k, \alpha_k \widetilde{v}^k \rangle \geq -\frac{\sigma}{2} \left\| \hat{z}(z^k, \widetilde{v}^k) - z^k \right\|^2 \tag{11}$$

   $$\left\| \hat{z}(z^k, \widetilde{v}^k) - \widetilde{z}^k \right\| \leq \zeta \left\| \hat{z}(z^k, \widetilde{v}^k) - z^k \right\|. \tag{12}$$

   (c) *Let $z^{k+1} = \hat{z}(z^k, \widetilde{v}^k)$, $k \leftarrow k + 1$, and repeat.*

□

Note that this algorithm contains no explicit reference to μ. Now, find $\bar{\mu}$ big enough such that $(\bar{\mu}-1)/(\bar{\mu}+1) > \sigma$ and choose $\bar{\sigma} \in (0,1)$ such that $\bar{\sigma}(\bar{\mu}-1)/(\bar{\mu}+1) = \sigma$. Define the sequence $\bar{\alpha}_k = (\bar{\mu}+1)\alpha_k$ for all k. Some simple algebra can then show that the ICPM is actually the IPMDR using $\mu = \bar{\mu}$, with $\bar{\sigma}$ and $\{\bar{\alpha}_k\}$ in place of the original parameters σ and $\{\alpha_k\}$ (and the same ζ).

2.1 Alternate notation

Define $\widetilde{P}(\cdot, z')$ to be the convex conjugate of $\widetilde{D}(\cdot, z')$, implying that the functions $\nabla_1 \widetilde{P}(\cdot, z')$ and $\nabla_1 \widetilde{D}(\cdot, z')$ are inverses. Note that by simple algebraic manipulation and application of this inverse relationship, the following equations are all equivalent:

$$\alpha_k v + \nabla_1 \widetilde{D}(z, z^k) = 0, \qquad v \in T(z) \tag{13}$$

$$\nabla_1 \widetilde{D}(z, z^k) = -\alpha_k v, \qquad v \in T(x) \tag{14}$$

$$z = \nabla_1 \widetilde{P}(-\alpha_k v, z^k), \quad v \in T(z) \tag{15}$$

$$z - \nabla_1 \widetilde{P}(-\alpha_k v, z^k) = 0, \qquad v \in T(z) \tag{16}$$

Solving any one of these systems is equivalent to z being the next iterate in the exact double-regularized proximal algorithm of [26]. We may in fact view (16) as an equivalent "dual" version of the original relation (13). Given (z, v) such that $v \in T(z)$, define

$$E_k(z, v) \stackrel{\text{def}}{=} \nabla_1 \widetilde{P}(-\alpha_k v, z^k), \tag{17}$$

that is, $E_k(z, v)$ is the right-hand side of (15); here the "E" stands for "extragradient", since the final extragradient step taken by the algorithm turns out to be of the form $z^{k+1} = E_k(z, v)$. Also define

$$R_k(z, v) \stackrel{\text{def}}{=} z - E_k(z, v). \tag{18}$$

Here, the "R" stands for "residual", since $R_k(z)$ is the residual of the equation (16). Finally, let

$$S_k(z, v) \stackrel{\text{def}}{=} z^k - E_k(z, v). \tag{19}$$

Here, the "S" stands for "step", since it represents the step taken by the algorithm.

The IPMDR can now be written in the following alternative form:

1. Let $(\tilde{z}^k, \tilde{v}^k)$ be a solution in (z, v) to the conditions

$$v \in T(z) \tag{20}$$

$$\langle R_k(z, v), -\alpha_k v \rangle \geq -\frac{\sigma(\mu - 1)}{2} \left\| S_k(z, v) \right\|^2 \tag{21}$$

$$\left\| R_k(z, v) \right\| \leq \zeta \left\| S_k(z, v) \right\|. \tag{22}$$

   Typically, one would find such a pair (z, v) by applying some iterative method to the system $R_k(z, v) = 0$, $v \in T(z)$, or equivalently (16).

2. Set $z^{k+1} = E_k(\tilde{z}^k)$, $k \leftarrow k+1$, and repeat.

Note that a sufficient condition guaranteeing both (21) and (22) is

$$\|R_k(z,v)\| \leq \min\left(\frac{\sigma(\mu-1)\|S_k(z,v)\|}{2\alpha_k\|v\|}, \zeta\right)\|S_k(z,v)\|, \tag{23}$$

note that if we take each $d_i(x_i, y_i) = (1/2)(x_i - y_i)^2$, that is, the classical proximal regularization, we then have $\widetilde{D}(x,y) = \frac{\mu+1}{2}\|x-y\|^2$ and thus $\nabla_1\widetilde{D}(z,z^k) = (\mu+1)(z-z^k)$. It then follows that $\nabla_1\widetilde{P}(u,z^k) = u/(\mu+1) + z^k$, and thus (23) simplifies to

$$\begin{aligned}\|R_k(z,v)\| &\leq \min\left(\frac{\sigma(\mu-1)\alpha_k\|v\|}{2\alpha_k(\mu+1)\|v\|}, \zeta\right)\|S_k(z,v)\| \\ &= \min\left(\frac{\sigma(\mu-1)}{2\alpha_k(\mu+1)}, \zeta\right)\|S_k(z,v)\|,\end{aligned}$$

which shows that the new criterion is in the spirit of the fixed relative error criterion suggested by Solodov and Svaiter [30].

3 Application to convex programming

Now consider applying the above algorithm to the primal-dual operator of a standard convex program

$$\begin{array}{ll}\min & f(x) \\ \text{ST} & g(x) \leq 0, \\ & x \in X,\end{array} \tag{24}$$

where $g(x) = \big(g_1(x), \ldots, g_m(x)\big)$, $f, g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ are convex function, and X is a closed convex set in $\mathbb{R}^n$ that represents "simple" constraints which may be enforced directly in the augmented Lagrangian subproblems. We assume that we know an explicit representation of the normal cone $N_X(x)$ to X at a point x.

We let T be the multivalued map on $\mathbb{R}^n \times \mathbb{R}^m$ given by

$$T(x,y) = \begin{bmatrix} \nabla f(x) + N_X(x) + \nabla g(x)^\top y \\ -g(x) \end{bmatrix} = \begin{bmatrix} \nabla_1 L(x,y) + N_X(x) \\ -g(x) \end{bmatrix}, \tag{25}$$

where $L(x,y) = f(x) + y^\top g(x)$ is the classical Lagrangian function for (24). Solving (24) is equivalent to solving the inclusion

$$T(x,y) + N_{\mathbb{R}^n \times \mathbb{R}^m_+}(x,y) \ni 0.$$

Assumptions 2.1.1–2.1.3 of [26] for $B = \mathbb{R}^n \times \mathbb{R}^m_+$ require that $\widetilde{D}$ take the form

$$\widetilde{D}\big((x,y),(x',y')\big) = \tfrac{1}{2}\|x-x'\|^2 + \overline{D}(y,y'),$$

where $\overline{D}$ is some double-regularized distance for $\mathbb{R}^m_+$. Thus, we have

$$\nabla_1 \widetilde{D}\big((x,y),(x',y')\big) = \begin{bmatrix} x - x' \\ \nabla_1 \overline{D}(y,y') \end{bmatrix}. \tag{26}$$

Define $\overline{P}(\cdot, y')$ to be the convex conjugate of $\overline{D}(\cdot, y')$ for all y'. Inverting (26) then yields

$$\nabla_1 \widetilde{P}\big((u,w),(x',y')\big) = \begin{bmatrix} x' + u \\ \nabla_1 \overline{P}(w,y') \end{bmatrix}. \tag{27}$$

We now restate the algorithm with (x,y) in place of z, and T as defined in (25). Using from (25) that $T(x,y) = \big\{\big(\nabla_1 L(x,y) + t, -g(x)\big) \mid t \in N_X(x)\big\}$, we define from (27), for $t \in N_X(x)$, the following quantities analogous to (17)-(19):

$$\begin{aligned} E_k(x,y,t) &\stackrel{\text{def}}{=} \begin{bmatrix} x^k - \alpha_k(\nabla_1 L(x,y) + t) \\ \nabla_1 \overline{P}(\alpha_k g(x), y^k) \end{bmatrix} \\ R_k(x,y,t) &\stackrel{\text{def}}{=} \begin{bmatrix} x - x^k + \alpha_k(\nabla_1 L(x,y) + t) \\ y - \nabla_1 \overline{P}(\alpha_k g(x), y^k) \end{bmatrix} \\ S_k(x,y,t) &\stackrel{\text{def}}{=} \begin{bmatrix} \alpha_k(\nabla_1 L(x,y) + t) \\ y^k - \nabla_1 \overline{P}(\alpha_k g(x), y^k) \end{bmatrix}. \end{aligned}$$

To simplify the notation, define

$$Y_k(x) = \nabla_1 \overline{P}(\alpha_k g(x), y^k).$$

In step one of the algorithm, we must compute an approximate solution of $R_k(x,y,t) = 0$. The process may be greatly simplified if we assume that we always choose $y = Y_k(x)$, which yields

$$R_k(x,t) \stackrel{\text{def}}{=} R_k(x, Y_k(x), t) = \begin{bmatrix} x - x^k + \alpha_k(\nabla_1 L(x, Y_k(x)) + t) \\ 0 \end{bmatrix}.$$

We can similarly define $E_k(x,t) \stackrel{\text{def}}{=} E_k(x, Y_k(x), t)$, and $S_k(x,t) \stackrel{\text{def}}{=} S_k(x, Y_k(x), t)$, and thus

$$E_k(x,t) = \begin{bmatrix} x^k - \alpha_k(\nabla_1 L(x, Y_k(x)) + t) \\ Y_k(x) \end{bmatrix} \qquad S_k(x,t) = \begin{bmatrix} \alpha_k(\nabla_1 L(x, Y_k(x)) + t) \\ y^k - Y_k(x) \end{bmatrix}.$$

Now consider the expression $\nabla_1 L(x, Y_k(x))$ that occurs in the formulas for $E_k(x,t)$, $R_k(x,t)$, and $S_k(x,t)$. We have

$$\begin{aligned} \nabla_1 L(x, Y_k(x)) &= \nabla f(x) + \nabla g(x)^\top Y_k(x) \\ &= \nabla f(x) + \nabla g(x)^\top \nabla_1 \overline{P}(\alpha_k g(x), y^k) \\ &= \nabla L_k^+(x), \end{aligned}$$

where $L_k^+(x)$ denotes the *augmented Lagrangian*

$$L_k^+(x) = f(x) + \tfrac{1}{\alpha_k} \overline{P}(\alpha_k g(x), y^k).$$

We also define the *proximal augmented Lagrangian*

$$\begin{aligned} L_k^{++}(x) &= L_k^{+}(x) + \tfrac{1}{2\alpha_k}\left\|x - x^k\right\|^2, \\ &= f(x) + \tfrac{1}{\alpha_k}\overline{P}(\alpha_k g(x), y^k) + \tfrac{1}{2\alpha_k}\left\|x - x^k\right\|^2, \end{aligned}$$

and note that

$$E_k(x,t) = \begin{bmatrix} x^k - \alpha_k(\nabla L_k^{+}(x) + t) \\ Y_k(x) \end{bmatrix} \tag{28}$$

$$R_k(x,t) = \begin{bmatrix} \alpha_k(\nabla L_k^{++}(x) + t) \\ 0 \end{bmatrix} \tag{29}$$

$$S_k(x,t) = \begin{bmatrix} \alpha_k(\nabla L_k^{+}(x) + t) \\ y^k - Y_k(x) \end{bmatrix}. \tag{30}$$

Now consider the first approximation criterion (21). Using that $v \in T(x,y)$ if and only if $v = \big(\nabla_1 L(x,y) + t, -g(x)\big)$ for some $t \in N_X(x)$, we obtain from (28)-(29) that the left-hand side of (21) now takes the form

$$\begin{aligned} \langle R_k(x,t), -\alpha_k T_k(x)\rangle &= \langle \alpha_k(\nabla L_k^{++}(x) + t), -\alpha_k(\nabla L_k^{+}(x) + t)\rangle + \langle 0, -g(x)\rangle \\ &= -\alpha_k^2 \langle \nabla L_k^{++}(x) + t, \nabla L_k^{+}(x) + t\rangle. \end{aligned} \tag{31}$$

Similarly, using (30), the right-hand side of (21) becomes

$$-\frac{\sigma(\mu-1)}{2}\left\|S_k(z,t)\right\|^2 = -\frac{\sigma(\mu-1)}{2}\left(\alpha_k^2\left\|\nabla L_k^{+}(x) + t\right\|^2 + \left\|y^k - Y_k(x)\right\|^2\right) \tag{32}$$

Combining (31) and (32), we obtain that

$$-\alpha_k^2\langle \nabla L_k^{++}(x) + t, \nabla L_k^{+}(x) + t\rangle \geq -\frac{\sigma(\mu-1)}{2}\left(\alpha_k^2\left\|\nabla L_k^{+}(x) + t\right\|^2 + \left\|y^k - Y_k(x)\right\|^2\right)$$

is equivalent to (21), and after some minor algebraic manipulations, we obtain the equivalent form

$$\langle \nabla L_k^{++}(x) + t, \nabla L_k^{+}(x) + t\rangle \leq \frac{\sigma(\mu-1)}{2}\left(\left\|\nabla L_k^{+}(x) + t\right\|^2 + \tfrac{1}{\alpha_k^2}\left\|y^k - Y_k(x)\right\|^2\right). \tag{33}$$

Similarly, we can square both sides of (22) and then substitute (29)-(30) to obtain the equivalent conditions

$$\begin{aligned} & \alpha_k^2\left\|\nabla L_k^{++}(x) + t\right\|^2 + \|0\|^2 \leq \zeta^2\left(\alpha_k^2\left\|\nabla L_k^{+}(x) + t\right\|^2 + \left\|y^k - Y_k(x)\right\|^2\right) \\ \Leftrightarrow \quad & \left\|\nabla L_k^{++}(x) + t\right\|^2 \leq \zeta^2\left(\left\|\nabla L_k^{+}(x) + t\right\|^2 + \tfrac{1}{\alpha_k^2}\left\|y^k - Y_k(x)\right\|^2\right). \end{aligned} \tag{34}$$

If we were to choose $\zeta = \sqrt{\sigma(1-\mu)/2}$, a sufficient condition guaranteeing both (33) and (34) would be

$$\max\left\{\langle \nabla L_k^{++}(x)+t, \nabla L_k^{+}(x)+t\rangle, \left\|\nabla L_k^{++}(x)+t\right\|^2\right\} \leq \frac{\sigma(\mu-1)}{2}\left(\left\|\nabla L_k^{+}(x)+t\right\|^2 + \tfrac{1}{\alpha_k^2}\left\|y^k - Y_k(x)\right\|^2\right).$$

We can now state a compact form of the IPMDR for this problem:

1. Apply an iterative method to the either of the equivalent problems

$$\min_{x\in X} L_k^{++}(x) \qquad \text{or} \qquad \nabla L_k^{++}(x) + N_X(x) \ni 0 \tag{35}$$

until obtaining an approximate solution x that satisfies the conditions

$$t \in N_X(x) \tag{36}$$

$$\langle \nabla L_k^{++}(x)+t, \nabla L_k^{+}(x)+t\rangle \leq \frac{\sigma(\mu-1)}{2}\left(\left\|\nabla L_k^{+}(x)+t\right\|^2 + \tfrac{1}{\alpha_k^2}\left\|y^k - Y_k(x)\right\|^2\right) \tag{37}$$

$$\left\|\nabla L_k^{++}(x)+t\right\|^2 \leq \zeta^2\left(\left\|\nabla L_k^{+}(x)+t\right\|^2 + \tfrac{1}{\alpha_k^2}\left\|y^k - Y_k(x)\right\|^2\right). \tag{38}$$

Call this solution $\tilde{x}^k$.

2. Perform the updates

$$y^{k+1} = Y_k(\tilde{x}^k) \tag{39}$$

$$x^{k+1} = x^k - \alpha_k(\nabla_1 L_k^{+}(\tilde{x}^k) + t). \tag{40}$$

Note that for each trial point x in the inner loop of step 1, we may calculate in sequence

$$\begin{aligned} Y_k(x) &= \nabla_1 \overline{P}(\alpha_k g(x), y^k) \\ \nabla L_k^{+}(x) &= \nabla_1 L(x, Y_k(x)) = \nabla f(x) + \nabla g(x)^\top Y_k(x) \\ \nabla L_k^{++}(x) &= \nabla L_k^{+}(x) + \tfrac{1}{\alpha_k}(x - x^k), \end{aligned}$$

which makes the tests (37)-(38) and possible subsequent updates (39)-(40) straightforward to perform. Note also that $\nabla L_k^{++}(x)$ is a quantity that algorithms for (35) are likely to need anyway. Note that if one exactly minimizes $L_k^{++}(x)$ over $x \in X$, there exists a $t \in N_X(x)$ such that the left-hand sides of (37) and (38) are both zero; in trying to satisfy (36)-(38) before finding an exact minimum, there may be some freedom in choosing $t \in N_X(x)$. Natural choices include the $t \in N_X(x)$ minimizing $\|\nabla L_k^{++}(x)+t\|$ or maximizing the number of zero elements in the vector $\nabla L_k^{++}(x)+t$. For the former, we define

$$t_X(w,x) \stackrel{\text{def}}{=} \arg\min\{\|w+t\| \mid t \in N_X(x)\} \qquad r_X(w,x) \stackrel{\text{def}}{=} w + t_X(w,x) \tag{41}$$

We would then choose $t = t_X(\nabla L_k^{++}(x), x)$ in (36). If $X = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\}$, that is, X is a box set, we obtain

$$r_X(w,x) = \begin{bmatrix} r_1(w_1,x_1) \\ \vdots \\ r_n(x_n,x_n) \end{bmatrix}, \quad \text{where} \quad r_i(w_i,x_i) \stackrel{\text{def}}{=} \begin{cases} \max\{w_i,0\}, & x_i = \ell_i \\ w_i, & \ell_i < x_i < u_i \\ \min\{w_i,0\}, & x_i = u_i. \end{cases} \tag{42}$$

Note that in this case, $t_X(w,x)$ is also the choice of $t \in N_X(x)$ that maximizes the number of zero components in $w + t$.

4 Numerical Experiments

We have created a prototype of the proximal augmented Lagrangian method described in Section 3 using Python [32] and SciPy [19], an open source environment including capabilities similar to MATLAB. We call this working edition "PyAugLag", for "Python augmented Lagrangian". Currently, PyAugLag uses uses only dense linear algebra, and is thus suitable only for small- to medium-scale problems. We believe that this limitation is acceptable for an initial, experimental implementation. Our source code can be downloaded from `http://www.ime.usp.br/~pjssilva/pyauglag`.

To test PyAugLag, we selected problems from the AMPL version of the CUTE library kindly made available by Hande Benson at `http://orfe.princeton.edu/~rvdb/ampl/nlmodels/cute/`. Our test set was composed all the problems from this repository meeting the following criteria:

- There are a most 500 variables, due to the prototype nature of our code and its use of dense linear algebra.
- There are at most 500 constraints, for the same reasons.
- All function values are defined throughout the whole space: PyAugLag currently has no provision for implicit constraints defined through function domains.
- There are at least two inequality constraints: without inequality constraints, the general nonquadratic penalties analyzed in Section 3 do not apply.
- There are no two-sided "range" constraints: PyAugLag does not yet have the ability to recognize and process such constraints.

The resulting test set consisted of 127 problems. The majority of the problems in the CUTE set are nonconvex, but in the spirit of [26], we still apply our method to them, seeking local optima. The results of [21, 18] provide some theoretical justification for such applications of proximal methods.

Instead of developing our own solver to minimize the subproblems of the proximal augmented Lagrangian algorithm, we used the ASA code due to Hager and Zhang [14]. ASA

can deal directly with simple constraints in the form of a box. Thus, in the terminology of Section 3, we take the set X of "simple" constraints to be $X = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\}$. ASA implements an efficient active set method [16] that selects various faces of the box, and uses an advanced nonlinear conjugate gradient algorithm [15] within each selected face. As a first-order method, it has good potential to take advantage of the premature subproblem termination conditions by error tolerance criteria like those given in (36)-(38). We made some minor changes to the ASA code, inserting "callbacks" to PyAugLag, so that it could use these termination criteria in place of its usual convergence test. In testing the criteria (37)-(38), we chose $t = t_X(\nabla L_k^{++}(x), x)$, where $t_X(\cdot, \cdot)$ is defined as in (41) for $X = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\}$.

Many test problems have both equality and inequality constraints, and hence PyAugLag processes problems in the form

$$\begin{array}{ll} \min & f(x) \\ \text{ST} & h(x) = 0 \\ & g(x) \leq 0 \\ & \ell \leq x \leq u \end{array} \tag{43}$$

where $f, h_1, \ldots, h_p, g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable. For each equality constraint $h_i(x) = 0$, we use the simple classical augmented Lagrangian penalty

$$P_i(x, y_i) = \langle h_i(x), y_i \rangle + \frac{1}{2\alpha_k} h_i(x)^2.$$

Because the Lagrange multiplier for each such constraint may range over all of $\mathbb{R}$, the double-regularization theory in [26] and Section 2 allows only this classical penalty. Furthermore, because this penalty is already smooth if h is, there is less motivation for substituting more sophisticated penalties than in the case of inequalities.

4.1 Details of the implementation

To test for termination of the IPMDR outer loop, we define the following quantities:

$$\gamma(x, y) = \|r_X(\nabla_1 L(x, y), x)\|_\infty \tag{44}$$

$$\phi(x) = \max\{\|h(x)\|_\infty, \|\max\{0, g(x)\}\|_\infty\} \tag{45}$$

$$\kappa_\epsilon(x, y) = \max\{|y_i| \mid i : g_i(x) \leq -\epsilon\}, \tag{46}$$

where $r_X(\cdot, \cdot)$ in (44) is defined as in (41) and (42). Here, $\gamma(x, y)$ measures how close x comes to minimizing the classical Lagrangian $L(x, y)$ with respect to x, $\phi(x)$ measures how close x is to being feasible, $\kappa_\epsilon(x, y)$ measures the violation of complementary slackness for the inequality constraints $g(x) \leq 0$. If $\gamma(x, y) = 0$, $\phi(x) = 0$, and $\kappa_0(x, y) = 0$, then (x, y) satisfies the KKT conditions for (43).

Given a parameter $\epsilon > 0$, PyAugLag terminates, declaring success, whenever it finds a primal-dual pair (x, y) satisfying the approximate KKT conditions

$$\gamma(x, y) < \epsilon \qquad \phi(x) < \epsilon \qquad \kappa_\epsilon(x, y) < \epsilon. \tag{47}$$

In our tests, we used $\epsilon = 10^{-4}$ (in future, we may wish to use tighter tolerances, but scale them in some relation to the problem data). On the other hand, we considered our method to have failed if any of the following occur:

- We cannot satisfy the approximate KKT conditions (47) within 200 box-constrained optimizations ($k \leq 200$)
- The ASA suproblems solver declares failure 5 or more times
- There are more than 1 million function evaluations
- The total CPU time exceeds one hour.

An important factor in the performance of any augmented Lagrangian code is its approach to updating the penalty parameter α_k. Here, we have borrowed the strategy used in the Algencan code [1, 2]. At the end of iteration k, PyAugLag checks whether

$$\phi(x^k) < \epsilon \quad \text{or} \quad \phi(x^k) \leq 0.5\,\phi(x^{k-1}) \tag{48}$$

$$\kappa_\epsilon(x^k, y^k) < \epsilon \quad \text{or} \quad \kappa_\epsilon(x^k, y^k) \leq 0.5\,\kappa_\epsilon(x^{k-1}, y^{k-1}). \tag{49}$$

If both (48) and (49) hold, then the method is considered to be making "good progress" towards a solution and the penalty parameter α_k is kept unchanged. Otherwise, the penalty parameter is increased by a factor of 5. Note that in the Algencan code, which uses a truncated Newton subproblem solver, the default increase factor is 10. However, as our method is based on a first-order box-constrained inner solver, such an aggressive increase in α_k proved too fast, decreasing the robustness of the overall method.

Finally, PyAugLag currently implements a choice of three different kinds of penalties for inequality constraints: the classical quadratic penalty, the log-quadratic penalty first proposed in [5], and the neural penalty with $\mu = 2$, as described in Appendix A. The neural penalty is similar to the penalty found to perform the best out of a collection of double-regularized and Bregman-distance kernels for complementarity problems in [26], but with a different μ for compatibility with our approximation criteria (however, [26] could not test a purely quadratic kernel approach, since it leads in that context to a nondifferentiable subproblem).

The log-quadratic and neural penalties are only defined for strictly positive multipliers y^k, and the multiplier update formula (39) in theory always return a strictly positive number. In practice, however, the computed multipliers can quickly become very small and numerically indistinguishable from zero. Such zero multipliers can arise in the first few iterations of the method, especially for multipliers associated with inactive constraints. As soon as a multiplier becomes zero, the log-quadratic and neural penalties become undefined. To overcome this problem, we used that, as a multiplier converges to 0, any penalty based on a double regularization will converge epigraphically, up to a constant, to the classical penalty with penalty parameter μ. This fact was proved for the log-quadratic penalty in [3], and the same proof can easily be adapted to the neural penalty. Hence, if a multiplier is very small

or zero, we can simply substitute the classical penalty for the coercive one. This approach is very different from the approach taken in [6, 7], in which one artificially limits the speed that a multiplier can approach zero. This technique can interfere with the convergence of the method, and we believe it should be avoided.

4.2 Computational experiments

In the experiments described below, we measure PyAugLag's performance via the total number of gradient evaluations performed until successful termination. Essentially identical results are obtained if one measures function evaluations instead of gradient evaluations. Here, such measures are more appropriate than CPU time for two reasons: first, PyAugLag is not yet performance-optimized and contains a mix of compiled code (in ASA) and interpreted Python code; second, some of the problems nevertheless solve so quickly that CPU time measurements are highly unreliable.

Exact versus inexact solution of subproblems

Our first experiment compares the performance of the "inexact" and the "exact" variants of the proximal augmented Lagrangian methods based on the three penalties. The "inexact" variant solves each subproblem up to the tolerance given in (36)-(38), whereas the "exact" version solves all subproblems to the same fixed, relatively high accuracy.

We selected the parameter σ controlling the tolerance of the inexact method, through a preliminary test using a subset of 65 of the more quickly solvable test problems. We tried the values 0.01, 0.1, 0.3, 0.5, 0.7, 0.9, and 0.99. For all three kinds of penalties, the best performance was obtained for $\sigma = 0.9$. For the exact method, we terminated the subproblem once $\|\nabla_1 L_k^{++}(x^k)\|_\infty < 10^{-5}$.

Figure 1 shows the three performance profiles [11], one for each penalty type, comparing the performance of the exact and inexact methods. We used $\mu = 2$ for both the log-quadratic and neural penalties. In terms of overall performance, all the penalty classes benefit from inexact solution of subproblems. With inexact computation, the log-quadratic penalty did fail in a few more cases than for the exact version, but the classical and the neural penalties did not suffer any clear decrease on robustness.

Penalty type comparison

Our next test compares the three kinds of penalties currently available in PyAugLag. There is a very extensive literature of proximal point methods devoted to theoretical convergence with different distance kernels, but it is almost entirely lacking numerical experiments. The only reference we know of that tries to compare different penalties numerically is [7]. In that paper, the authors conclude that the classical penalty outperforms many others, in particular the log-quadratic, in the context of nonlinear optimization. However, there are several possible reasons why these conclusions might not have been definitive:

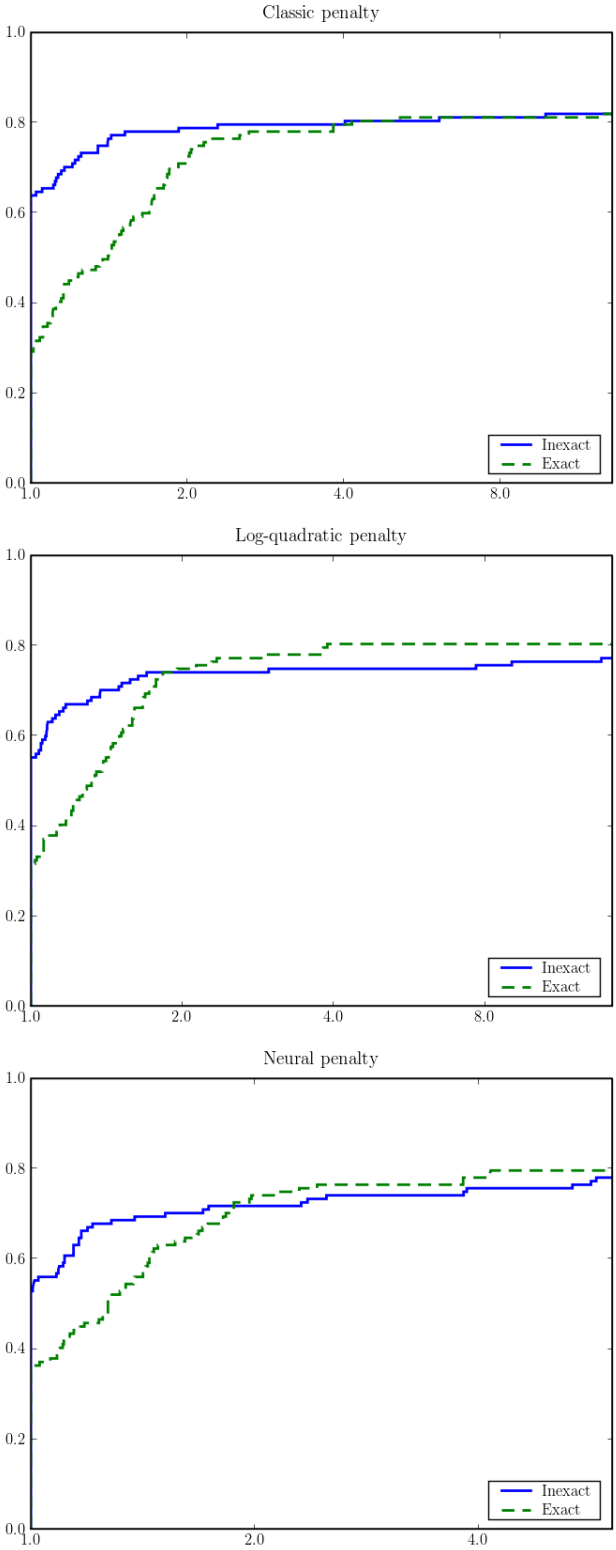

Figure 1: Exact versus Inexact veriants of the proximal augmented Lagrangian

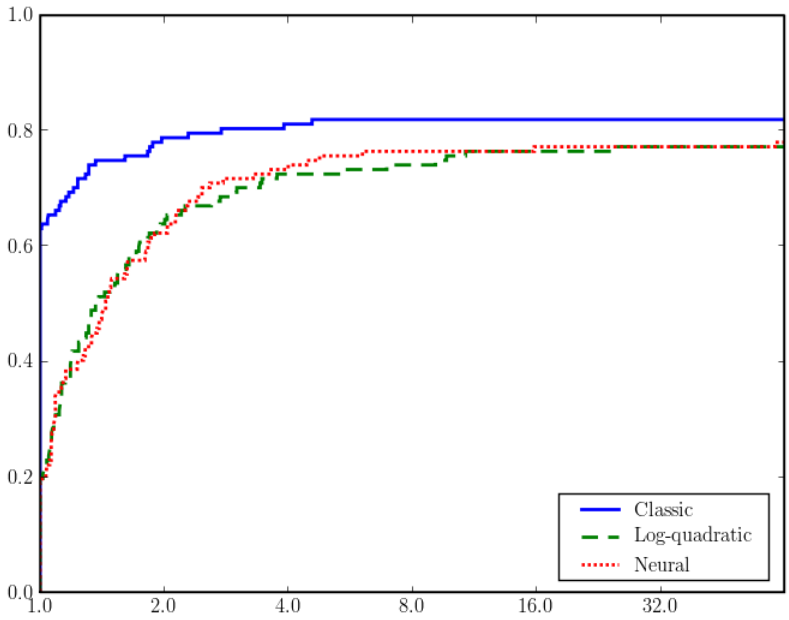

Figure 2: Comparing the three penalties in the inexact proximal augmented lagragian: classic, log-quadradic, and neural.

- They did not consider the neural penalty, which had the best performance for complementarity problems in [26].
- They artificially limit the speed of convergence of multipliers to zero, as mentioned at the end of Section 4.1. This technique could have placed the nonquadratic methods at a disadvantage.
- They considered only exact minimization of subproblems.
- They use only the augmented Lagrangian L_k^+, rather than the proximal augmented Lagrangian L_K^{++}.

Figure 2 compares the performance of the three penalty classes, using inexact computation with $\sigma = 0.9$. In these results, the performance of the loq-quadratic and neural penalties appears very similar. As in [7], but under different conditions, the classic penalty clearly performs better than the others. This result is somewhat surprising given that the classic penalty is not twice differentiable, often considered a significant practical disadvantage. However, its gradient is semi-smooth, and such functions are well known to behave well even with algorithms that use explicit second-order information, such as the Newton method [20, 22, 23]. This property is probably also related to the good performance we observe using a conjugate gradient method.

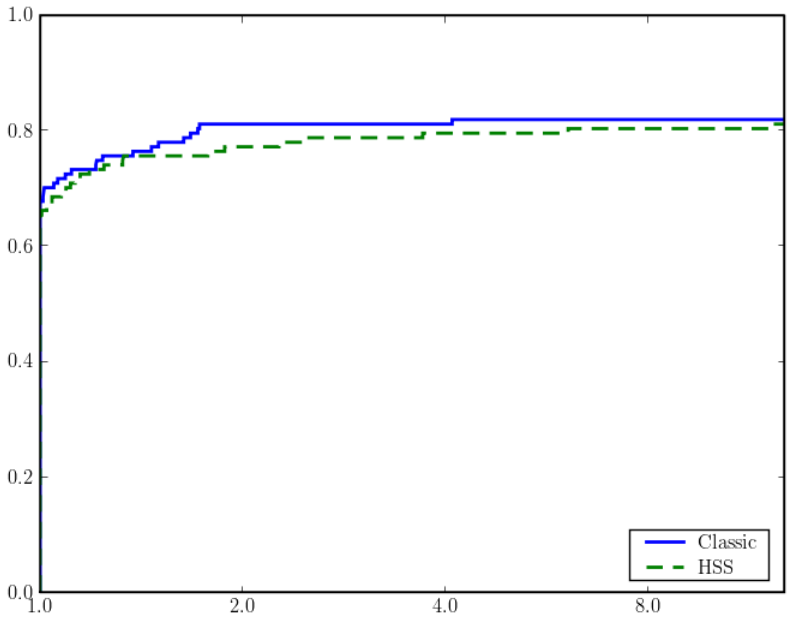

Figure 3: PyAugLag with the classic quadratic penalty versus the HSS inexact algorithm of [17].

Comparison to previous methods

In [17], Humes, Silva, and Svaiter have already proposed variations of the proximal augmented Lagrangian method including a relative error criterion. The methods proposed are similar to the algorithm described here, but limited to the classic penalty. In our present notation, the subproblem termination criterion in [17] is

$$\left\|\nabla L^{++}(x)\right\|^2 \leq \frac{\sigma^2}{\alpha_k^2}\left(\left\|x - x^k\right\|^2 + \left\|Y_k(x) - y^k\right\|^2\right). \tag{50}$$

Taking into account (40), the right-hand side of this criterion is essentially the same as that of (37); however, the left-hand sides of the two criteria are different.

Figure 3 shows a performance profile comparing PyAugLag, using the classic quadratic penalty, to the method proposed in [17], which we call HSS. It shows that the new method constitutes a small improvement over HSS, solving a few of the hard problems faster. Note that HSS also has a parameter σ controlling the acceptance criterion: once again, we selected σ by experimentation on subset of the easier problems. Once again, $\sigma = 0.9$ appeared to be the best choice. Note also that our current experiments test the HSS algorithm much more thoroughly than in [17] itself.

The HSS method was shown in [17] to be slower than the usual, "pure dual" augmented Lagrangian method without the primal regularization term $(1/2\alpha_k)\|x - x^k\|^2$, that is, the method that at each iteration minimizes the augmented Lagrangian function L_k^+, rather

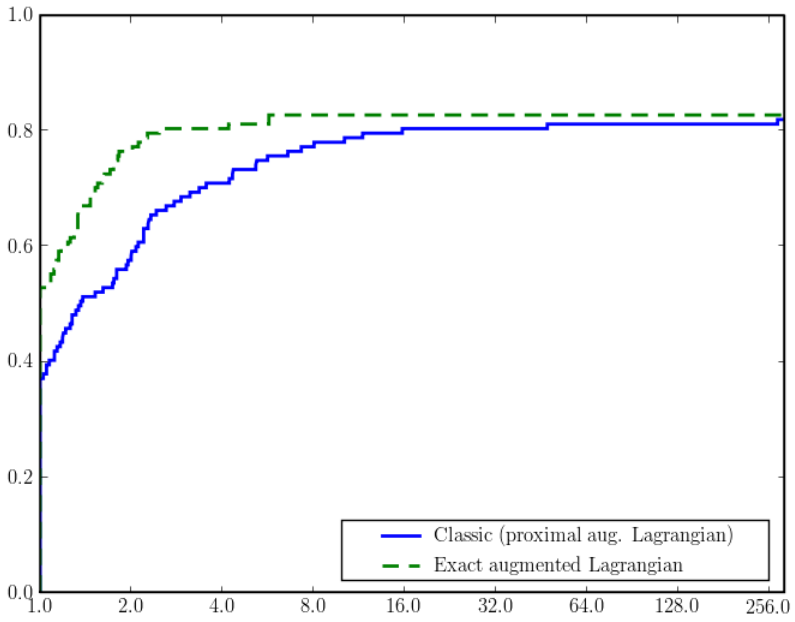

Figure 4: Exact proximal augmented Lagrangian versus an exact (pure dual) augmented Lagrangian.

than L_k^{++}, followed by the multiplier update (39). This method serves as the basis of many successful implementations like Lancelot [10] and Algencan [1, 2].

Figure 4 compares the performance of PyAugLag with and without the primal regularizing term $(1/2\alpha_k)\|x - x^k\|^2$, in both cases using the "exact" subproblem minimization criterion $\left\|\nabla L_k^{++}(x)\right\| \leq 10^{-5}$.

Clearly the pure dual method omitting the primal regularizing term performs better, being faster without any sacrifice in reliability. This result is at odds with the results concerning complementarity problems in [26], where the primal-dual approach was seen to significantly improve reliability at the cost of a minor speed penalty. However, the primal-dual, proximal augmented Lagrangian method has theoretical advantages over the pure dual approach: it is much easier to develop error criteria such as (36)-(38) for primal-dual methods, and they have a more satisfactory convergence theory for the primal sequence $\{x^k\}$.

Future directions: inexact computation in pure dual methods

It is now natural to ask if one can use what we have learned from the primal-dual approach to derive an inexact subproblem termination criterion for the pure dual method. Let us first consider the right-hand side of the acceptance criteria given in (37)-(38) and (50). In both cases, we can find a term that measures the change in the primal iterate x^k. It is a subgradient of the augmented Lagrangian in the first case, and the primal step divided by α_k in the second. There is also a term that measures the dual change, which is the dual step

divided by α_k in both cases. This result is natural, as the proximal augmented Lagrangian can be viewed as a proximal point method applied to the primal-dual operator T defined in (25).

Since the augmented Lagrangian algorithm is a proximal method acting only in the dual variables, it seems natural to use only the dual term in the right hand side of an acceptance criterion. Moreover the natural left-hand side is the gradient of the augmented Lagrangian L_k^+, the function that is minimized at each iteration. That is, we propose to use the following inequality as an acceptance criterion for the augmented Lagrangian method:

$$\left\|\nabla L_k^+(x)\right\| \leq \frac{\sigma}{\alpha_k} \left\|Y_k(x) - y^k\right\|. \tag{51}$$

For equality constraints, $1/\alpha_k \left\|(Y_k(x))_i - y_i^k\right\|$ can be rewritten as $h_i(x)$; for inequality constraints, it can be expressed as $\max\{g_i(x), -y_i^k/\alpha_k\}$. This quantity has already appeared in the literature: in [1], the authors show, under suitable assumptions, that if each subproblem is solved to the precision

$$\left\|\nabla L_k^+(x)\right\| \leq \frac{\eta_k}{\alpha_k} \left\|Y_k(x) - y^k\right\|,$$

where η_k is a positive sequence converging to zero, then the sequence penalties parameters $\{\alpha_k\}$ computed by their algorithm is bounded. However, in this work, the condition that ensures convergence is $\left\|\nabla L_k^+(x)\right\| \leq \delta_k$, where $\delta_k \to 0$. Hence, in order to guarantee a convergent method with bounded parameters, the approximate solution to each subproblem should conform to

$$\left\|\nabla L_k^+(x)\right\| \leq \min\left\{\delta_k, \frac{\eta_k}{\alpha_k} \left\|Y_k(x) - y^k\right\|\right\}.$$

However, the Algencan code described in [1] does not use the criterion above directly. By default, the current stable version simply solves each subproblem to a fixed high precision. A possible reason for this choice is that, as unlike our implementation, Algencan uses a truncated-Newtron-based subproblem solver that can achieve high accuracy easily once in the neighborhood of the solution.

However, if the inner solver is a first-order method, obtaining high precision for all problems may not be beneficial, as our tests with the proximal augmented Lagrangian suggest. Figure 5 compares the performance of two variants of PyAugLag, both with the primal regularizing term omitted, and both using the ASA method to solve the subproblems. One version uses the "exact" subproblem termination criterion $\|\nabla L_k^+(x)\|_\infty \leq 10^{-5}$, and the second uses (51), augmented by safeguards ensuring theoretical convergence for convex problems, following the criterion suggested in [13]. Once again, we used a subset of the problems to select a good value for σ, which in this case turned out to be 0.5. Note that the inexact variant is clearly superior to the exact one.

Finally, we note that the current beta version of Algencan incorporates a new subproblem termination strategy. Let ϵ denote the precision required in the approximate KKT condition (47). Now, instead of using a fixed subproblem precision $\tilde{\epsilon} < \epsilon$, Algencan first solves the subproblems with precision $\sqrt{\epsilon}$ until it obtains a pair (x, y) satisfying $\phi(x) < \sqrt{\epsilon}$ and

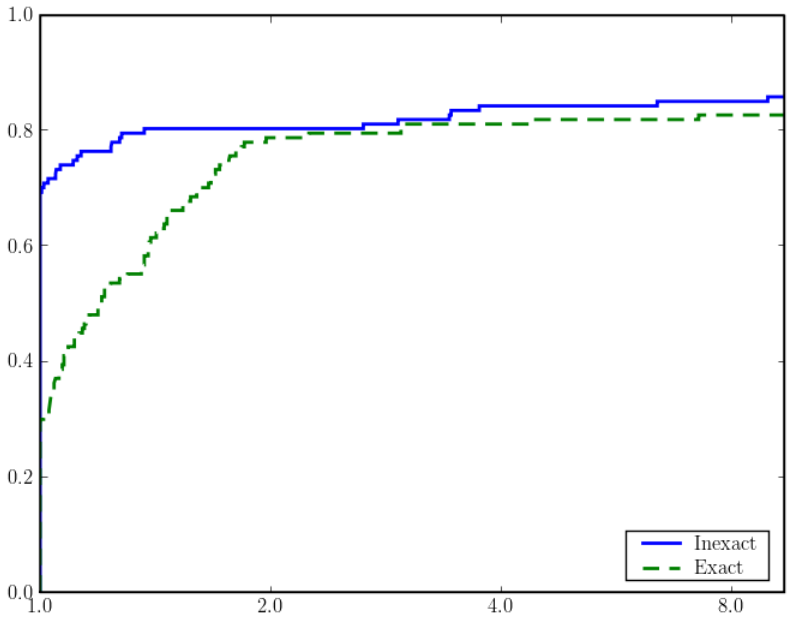

Figure 5: Inexact versus exact variant of the (pure dual) augmented Lagrangian.

$\kappa_{\sqrt{\epsilon}}(x, y) < \sqrt{\epsilon}$. After the first iteration in which these conditions are met, all subproblems are solved to the full precision $\tilde{\epsilon}$.

Implementing this same strategy is also beneficial in PyAugLag, but it does not achieve the performance of the variant based on equation (51); see Figure 6. It appears that further research should be carried out to analyze the acceptance criterion (51), in particular to see if the safeguards adapted from [13] are really necessary, and whether a more attractive theoretical acceptance condition can be obtained.

4.3 Conclusions from numerical experiments

In conclusion, it appears that one of our motivations for developing PyAugLag, to take advantage of recently developed nonquadratic distance kernels for proximal methods, was somewhat misconceived. For nonlinear optimization applications, we found that the classical quadratic penalty works remarkably well, confirming in a different context the results of [7]. On the other hand, the idea of using a first-order subproblem solver coupled with a relative-error subproblem termination criterion appears very promising. However, our empirical experiments suggest that the idea will work even better in a pure dual context than a primal-dual one. It would be desirable however, to have a firm theoretical foundation for such a use of a relative error criterion, at least for convex problems. Verifiable error criteria for pure dual augmented Lagrangian methods are tricky to develop, but in view of our results, it appears worthwhile to try to further develop the summable-error approximation analysis

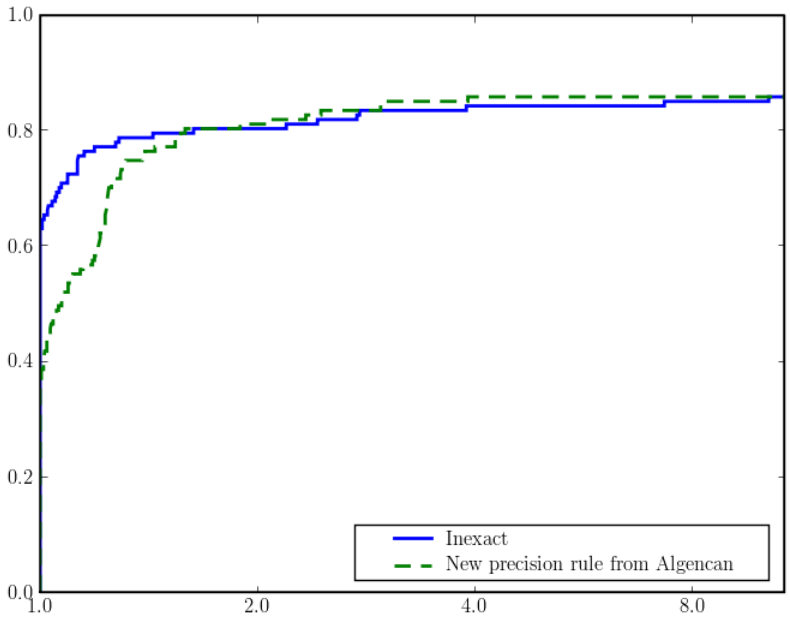

Figure 6: Inexact variants based on (51) and the beta Algencan acceptance rule.

in [13] into some kind of relative-error form.

A The neural penalty

In [26], the best-performing penalty was the neural penalty, defined by

$$\nabla_1 \overline{P}(u, y) = y \log_2(2^{\frac{u}{y}} + 1), \tag{52}$$

interpreted componentwise, that is, $(\partial/\partial x_i)\overline{P}(u, y) = y_i \log_2(2^{\frac{u_i}{y_i}} + 1)$. This penalty was shown to be associated with a double regularization with $\mu = 1$. However, the convergence results given in Section 2 depend on the choice of $\mu > 1$. Furthermore, setting $\mu = 1$ makes the error criterion (4) so strict it is tantamount to exact minimization of the augmented Lagrangian. To experiment with the neural penalty in the $\mu > 1$ context, we develop a modified version equivalent to a double regularization for which $\mu = 2$. From the results of [26], the gradient of the regularization corresponding to (52) is (again interpreted componentwise)

$$\nabla_1 \overline{D}(x, y) = y \log_2(2^{\frac{x}{y}} - 1).$$

It was shown in [26] that $\overline{D}$ is a regularization of the form (2) with $\mu = 1$. For general μ, we would thus have the regularization gradient

$$\nabla_1 \overline{D}_\mu(x, y) = y \log_2(2^{\frac{x}{y}} - 1) + (\mu - 1)(x - y).$$

To find the penalty $\overline{P}_\mu$ corresponding to this regularization, we must find the inverse of this expression with respect to x. Using that the expression is separable, we set, for each i,

$$\left(\frac{\partial}{\partial x_i}\right)\overline{D}_\mu(x,y) = y\log_2(2^{\frac{u_i}{y_i}}-1) + (\mu-1)(x_i-y_i) = u_i.$$

After some simple algebra, we arrive at

$$2^{\frac{x_i}{y_i}} - 1 = 2^{\frac{u_i+(\mu-1)y_i}{y_i}}(2^{\frac{-x_i}{y_i}})^{\mu-1}.$$

Defining $\alpha \stackrel{\text{def}}{=} 2^{\frac{x_i}{y_i}}$ and $\beta \stackrel{\text{def}}{=} 2^{\frac{u_i+(\mu-1)y_i}{y_i}}$, this equation is equivalent to

$$\alpha^\mu - \alpha^{\mu-1} - \beta = 0.$$

This equation can be easily solved for $\mu = 1$, but that would simply recover the original neural penalty (52). However, it can also be soleved for $\mu = 2$, leading to the modified penalty

$$\nabla_1\overline{P}_2(u,y) = y\log_2\left(1+\sqrt{1+2^{\frac{u+3y}{y}}}\right) - y,$$

again interpreted componentwise. Since this penalty is built upon the original neural penalty, conforming to the bounds given in [26, Equation (25)] for $\mu = 1$, it conforms to the same bounds for $\mu = 2$.

References

[1] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. On augmented Lagrangian methods with general lower-level constraints. *SIAM J. Optim.*, 18(4):1286–1309, 2007.

[2] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. Augmented Lagrangian methods under the constant positive linear dependence constraint qualification. *Math. Program.*, 111(1–2):5–32, 2008.

[3] A. Auslender, P. J. S. Silva, and M. Teboulle. Nonmonotone projected gradient methods based on barrier and Euclidean distances. *Comput. Optim. Appl.*, 38(3):305–327, 2007.

[4] A. Auslender, M. Teboulle, and S. Ben-Tiba. Interior proximal and multiplier methods based on second order homogeneous kernels. *Math. Oper. Res.*, 24(3):645–668, 1999.

[5] A. Auslender, M. Teboulle, and S. Ben-Tiba. A logarithmic-quadratic proximal method for variational inequalities. *Comput. Optim. Appl.*, 12(1–3):31–40, 1999.

[6] A. Ben-Tal and M. Zibulevsky. Penalty/barrier multiplier methods for convex programming problems. *SIAM J. Optim.*, 7(2):347–366, 1997.

[7] E. G. Birgin, R. A. Castillo, and J. M. Martínez. Numerical comparison of augmented Lagrangian algorithms for nonconvex problems. *Comput. Optim. Appl.*, 31(1):31–55, 2005.

[8] R. Burachick and B. F. Svaiter. A relative error tolerance for a family of generalized proximal point methods. *Math. Oper. Res.*, 26(4):816–831, 2001.

[9] Y. Censor and S. A. Zenios. Proximal minimization algorithm with D-functions. *J. Optim. Theory Appl.*, 73(3):451–464, 1992.

[10] A. R. Conn, N. I. M. Gould, and P. L. Toint. *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*. Springer-Verlag, 1992.

[11] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2):201–213, 2002.

[12] J. Eckstein. Nonlinear proximal point algorithms using Bregman functions, with applications to convex programming. *Math. Oper. Res.*, 18(1):202–226, 1993.

[13] J. Eckstein. A practical general approximation criterion for methods of multipliers based on Bregman distances. *Math. Program.*, 96(1):61–86, 2003.

[14] W. W. Hager and H. Zhang. ASA-CG source code. `http://www.math.ufl.edu/~hager/papers/CG/`.

[15] W. W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM J. Optim.*, 16(1):170–192, 2005.

[16] W. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIAM J. Optim.*, 17(2):526–557, 2006.

[17] C. Humes Jr., P. J. S. Silva, and B. F. Svaiter. Some inexact hybrid proximal augmented Lagrangian algorithms. *Numer. Alg.*, 35(2–4):175–184, 2004.

[18] A. N. Iusem, T. Pennanen, and B. F. Svaiter. Inexact variants of the proximal point algorithm without monotonicity. *SIAM J. Optim.*, 13(4):1080–1097, 2003.

[19] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001. `http://www.scipy.org/`.

[20] J.-S. Pang and L. Qi. Nonsmooth equations: motivation and algorithms. *SIAM J. Optim.*, 3:443–465, 1993.

[21] T. Pennanen. Local convergence of the proximal point algorithm and multiplier methods without monotonicity. *Math. Oper. Res.*, 27(1):170–191, 2002.

[22] L. Q. Qi. Convergence analysis of some algorithms for solving nonsmooth equations. *Math. Oper. Res.*, 18(1):227–244, 1993.

[23] L. Q. Qi and J. Sun. A nonsmooth version of Newton's method. *Math. Program.*, 58(3):353–367, 1993.

[24] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Math. Oper. Res.*, 1(2):97–116, 1976.

[25] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5):877–898, 1976.

[26] P. J. S. Silva and J. Eckstein. Double-regularization proximal methods, with complementarity applications. *Comput. Optim. Applic.*, 33(2–3):115–156, 2006.

[27] P. J. S. Silva, J. Eckstein, and C. Humes Jr. Rescaling and stepsize selection in proximal methods using generalized distances. *SIAM J. on Optim.*, 12(1):238–261, 2001.

[28] M. V. Solodov and B. F. Svaiter. A hybrid approximate extragradient-proximal point algorithm using the enlargement of a maximal monotone operator. *Set-Valued Anal.*, 7(4):323–345, 1999.

[29] M. V. Solodov and B. F. Svaiter. A hybrid projection-proximal point algorithm. *J. Convex Anal.*, 6(1):59–70, 1999.

[30] M. V. Solodov and B. F. Svaiter. An inexact hybrid generalized proximal point algorithm and some new results on the theory of Bregman functions. *Math. Oper. Res.*, 25(2):214–230, 2000.

[31] M. Teboulle. Entropic proximal mappings with applications to nonlinear programming. *Math. Oper. Res.*, 17(3):670–690, 1992.

[32] G. van Rossum et al. Python language website. `http://www.python.org/`.