CONSTRAINED BUNDLE METHODS FOR UPPER INEXACT ORACLES WITH APPLICATION TO JOINT CHANCE CONSTRAINED ENERGY PROBLEMS

WIM VAN ACKOOIJ[†‡] AND CLAUDIA SAGASTIZÁBAL[§]

Abstract. Joint chance constrained problems give rise to many algorithmic challenges. Even in the convex case, i.e., when an appropriate transformation of the probabilistic constraint is a convex function, its cutting-plane linearization is just an approximation, produced by an oracle providing subgradient and function values that can only be evaluated inexactly. As a result, the cutting-plane model may lie above the true constraint. For dealing with such *upper* inexact oracles, and still solving the problem up to certain precision, a special numerical algorithm must be put in place. We introduce a family of constrained bundle methods, based on the so-called improvement functions, that is shown to be convergent and encompasses many previous approaches as well as new algorithms. Depending on the oracle accuracy, we analyze to which extent the considered methods solve the joint chance constrained program. The approach is assessed on real-life energy problems, arising when dealing with stochastic hydro-reservoir management.

Key words. Joint Chance Constraints, Bundle Methods, Inexact Oracles, Hydro-Reservoir Management

AMS subject classifications. 49M37, 65K05, 90B36, 90C15, 90C25

1. Introduction and Motivation. Real-life problems are often modeled in an uncertain setting, to better reflect unknown phenomena specific to the application. In particular, such is the case in the energy sector; see [33]. For the numerical experience of this paper we focus on a specific energy problem arising in stochastic unit-commitment (e.g., [44] and references therein).This is the problem of optimally managing reservoirs of a hydro valley in the short term.

A hydro valley is a set of power plants cascaded along the same hydrological bassin. For the considered system, part of Electricité de France mix, uncertainty is mostly related to the amount of melted snow arriving as a streamflow to the most uphill reservoirs. The volume of these reservoirs changes with the inflows and determines the amount of water that can be converted into energy. After turbining water to produce power, the uphill reservoirs release a certain volume that fills the reservoirs downhill, and the process continues until the power plant at the bottom of the bassin. In this interconnected context, it is important to jointly manage the generation of the cascaded plants in a manner that not only is economical but also reliable. More precisely, it is crucial to keep the volume of each reservoir in the valley between prescribed minimum and maximum levels (to prevent floods, to ensure touristic activities, etc). Since it is not realistic to ensure such conditions for every possible streamflow, satisfaction of lower and upper bounds for the volumes can be required in a probabilistic manner.

Probabilistic (or chance) constraints, introduced by [3], have the form

$$\mathbb{P}[g(x,\xi) \geq 0] \geq p, \tag{1.1}$$

where $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{\tilde{m}}$ is a vectorial mapping, $x \in \mathbb{R}^n$ the decision vector, $\xi \in \mathbb{R}^m$ a random variable and $p \in (0,1)$ the requested probability level. Such constraints naturally arise whenever a decision has to be taken prior to observing uncertainty and the decision has to satisfy an inequality system $g(x,\xi) \geq 0$. One can then give a simple interpretation to safety of the decision x by requesting that (1.1) holds. For hydro valley management, chance constraints have been employed in [8, 9, 24, 25, 27, 41, 45, 46]. Most of these works consider the probability separately, over each scalar component in the mapping g in (1.1). As explained in [41], a stochastic model with individual chance constraints may sometimes result in unreliable optimal decisions, because there is no guarantee that the whole stochastic inequality system $g(x,\xi) \geq 0$ will be satisfied with a given probability. In this work we build upon the model in [41] with *joint* chance constraints, that takes $\tilde{m} > 1$ in (1.1), and derive a sound numerical solution procedure, based on bundle methods, [2, 19].

In the classical textbook [28], convexity of chance constraints is ensured for a variety of distributions. Differentiability properties including gradient formulæ for Gaussian distributions are analyzed in [41, 16] and [40], whereas gradient formulæ for other distributions can be found in [31, 28, 37, 15].

[†]EDF R&D. OSIRIS, 1, avenue du Général de Gaulle, F-92141 Clamart Cedex France (wim.van-ackooij@edf.fr).

[‡]Ecole Centrale Paris, Grande Voie des Vignes, F-92295, Châtenay-Malabry.

[§]Visiting researcher at IMPA, Brazil. On leave from INRIA, France (sagastiz@impa.br).

Our hydro valley management problem is a convex joint chance constrained problem with the abstract form:

$$\min_{x\in X\subseteq\mathbb{R}^n}\{f(x) : c(x)\leq 0\} \tag{1.2}$$

for X a compact convex polyhedron and f and c finite-valued convex continuous functions, possibly smooth. Nevertheless, even when the joint chance constraint

$$c(x) := \log(p) - \log(\mathbb{P}[g(x,\xi)\geq 0]) \tag{1.3}$$

is smooth, it can define a gully-shaped feasible region that makes (1.2) hard to tackle with standard nonlinear programming solvers [26]. Bundle methods, by contrast, handle well this type of problems. Indeed, as explained in [2, Chapter 12.1], a (nonsmooth) bundle method can outperform a quasi-Newton algorithm when the smooth problem is stiff. The situation with problem (1.2) is similar, worsened by the fact that in (1.3) the probability cannot be computed exactly, but just *estimated* by combining numerical integration techniques with sampling (a detailed discussion is given in Section 5.1). We have a problem for which not only the feasible set is ill conditioned, but also the available information is inaccurate. The context is therefore favourable for solving the problem with a nonsmooth method that is reliable and robust regarding both ill conditioning and inaccuracy, such as the recent bundle variants capable of dealing with inexact information, e.g. [6]. Incidentally, it is worth mentioning that in [17] it is shown that the constraint can indeed be nonsmooth when the covariance matrix of ξ in (1.1) is positive semidefinite (such is the case in network optimization problems with random nodal injections). Using the general gradient formulæ in [16] it is possible to compute approximate subgradients for c. In this setting, a smooth method has chances of failing because it can get stuck at a non-optimal kink.

Since neither the chance-constraint nor its gradient can be evaluated exactly, a linearization of the form $c_x + \langle g_{c_x}, \cdot - x\rangle$ with $c_x \approx c(x)$ and g_{c_x} an approximate subgradient, may lie *above* the function $c(\cdot)$. To circumvent this difficulty, in this paper we present a bundle method specially taylored to solve problems of the form (1.2) when computing f and/or c (as well as respective gradients) exactly is not possible or it is computationally heavy. The algorithm is special because it solves a constrained nonsmooth problem based on the information provided by an inexact oracle, possibly of *upper* type: linearizations for f and c in (1.2) are inexact and may lie *above* the corresponding function. The simpler case of *lower* oracles, yielding linearizations that always remain below the convex function, is also considered as a corollary. The convergence analysis of bundle methods with lower oracles is simpler, because it is closer to the usual exact framework, in which the oracle linearizations define cutting planes for the involved functions (f and c in our case).

For unconstrained problems, bundle methods dealing with inexact oracles can be found in [10, 11, 18, 20, 7, 36]. Most of these works consider only *lower* oracles; we refer to [7] for a discussion on how such a setting considerably simplifies the convergence analysis. For constrained problems like (1.2), inexact bundle methods are more rare; see [21, 23, 5]. These works consider oracles that are either lower ones, or asymptotically exact. In this paper, we give a method suitable for the more general upper setting, and hence, adapted to the hydro application of interest.

Our paper is organized as follows. Section 2 is devoted to bundle methods for upper inexact oracles. After giving the initial bundle setting in Section 2.1, the attenuation step required for the method to converge in the presence of oracle noise is explained in Section 2.2. The new bundle algorithm is given in full details in Section 2.3. Based on Section 3 asymptotic results, Section 4 proves convergence of the method to a solution of (1.2), up to the accuracy provided by the oracle. In particular, we show that for lower oracles that are asymptotically exact, the method finds an exact minimizer whenever (1.2) has a Slater point. Since our setting is more general than previous work (cf. the discussion in Section 4.2), our approach significantly generalizes and extends results in the literature. The specific application is considered in the last two sections. Section 5 formulates the joint chance constrained problem to be solved, and discusses the upper inexact oracle employed. Regarding this last issue, since our method is *deterministic* and not randomized, we make the simplifying assumption that the joint chance constraint and its gradient can be effectively computed with any required precision (at a 100% confidence level, up to the machine precision); we refer to Section 5.1 for more details. The different solvers used for comparison and a thorough set of numerical tests showing the interest of the approach are given in Section 6. The paper ends with some concluding remarks and an appendix with details of the hydro valley problem considered in the numerical tests.

2. Designing Bundle methods for Constrained Optimization. Before proceeding with our discussion, we introduce the *improvement function* $H_\tau : \mathbb{R}^n \to \mathbb{R}$ associated with problem (1.2):

$$H_\tau(y) = \max\Big(f(y) - \tau_1, c(y) - \tau_2 \Big), \text{ for suitably chosen scalar targets } \tau_1, \tau_2 . \tag{2.1}$$

Among other properties, assuming a Slater constraint qualification and letting $\bar\tau := (f(\bar x), 0)$, Theorem 2.1 in [34] shows that $\bar x \in X$ is a solution to (1.2) if and only if $\bar x$ solves the simpler problem of minimizing $H_{\bar\tau}(y)$ over X.

Like in [34], our algorithmic procedure defines targets that vary at each iteration and eventually converge to the final target, $\bar\tau$. However, unlike the exact setting considered in [34], the oracles which provide function and subgradient values for f and c make calculations with some error. Our method minimizes approximations of H_τ, built using oracle information computed with some inaccuracy. For clarity, at any point $x \in \mathbb{R}^n$ the symbols $f(x)$ and $c(x)$ refer to *exact* function values. Following [20], to denote *inexact* values the argument is put as a subscript, like in f_x, c_x (for functions) and $g_{\mathtt{f}x}$ and $g_{\mathtt{c}x}$ (for subgradients). Given an iterate $x^j \in X$, the oracle provides f_{x^j} and $g_{\mathtt{f}x^j}$ shortened for convenience to $f^j = f_{x^j}$ and $g^j_{\mathtt{f}} = g_{\mathtt{f}x^j}$, and similarly for the c-values.

2.1. Initial setting. We assume that at any $x^j \in X$, the oracle provides

$$\begin{array}{lll} f^j \quad \text{and} \quad c^j, & \text{estimates for the functional values, as well as} \\ g^j_{\mathtt{f}} \quad \text{and} \quad g^j_{\mathtt{c}}, & \text{estimates for the respective subgradients.} \end{array} \tag{2.2}$$

Since in this oracle the signs of the errors, e.g., $f(x^j) - f^j$, are not specified, the true function values can be either over or underestimated, and similarly for the subgradients. In particular, nothing is known on the linearizations, e.g., $f^j + \left\langle g^j_{\mathtt{f}}, \cdot - x^j \right\rangle$, that may lie below or above the corresponding function, e.g., f. Further conditions on the (possibly upper) inexact oracle will be required in what follows as needed (cf. (2.20) and (4.1) below).

Along iterations, the method keeps an algorithmic center, denoted by $\hat x^k$ at iteration k, with function values denoted by $\hat f^k$ and $\hat c^k$. The center is some past iterate that was singled out because it produced significant progress towards the goal of solving (1.2). As explained in what follows, progress is measured with respect to the current approximation of the improvement function. Specifically, the k-th *inexact improvement function* is

$$h^k_y = \max\Big(f_y - \tau^k_1, c_y - \tau^k_2 \Big) \quad \text{where} \quad \left\{ \begin{array}{ll} \tau^k_1 = \hat f^k + \rho_k \max(\hat c^k, 0) & \text{for } \rho_k \geq 0 \\ \tau^k_2 = \sigma_k \max(\hat c^k, 0) & \text{for } \sigma_k \geq 0 . \end{array} \right. \tag{2.3}$$

In the expression above, the targets τ^k are more general than those considered in [34], that takes null penalties ρ_k and σ_k. Relations with other improvement functions in the literature are discussed in Section 4.2.

The oracle output is collected along iterations to form the *Bundle of information*

$$\mathscr{B}^k = \{\hat x^k, \hat f^k, \hat c^k\} \cup \Big\{ (x^j, f^j, c^j, g^j_{\mathtt{f}}, g^j_{\mathtt{c}}) : j \in J^k \Big\} \quad \text{for } J^k \subset \{1, \ldots, k\} .$$

Having this information, the k-th inexact improvement function is modelled by a convex function $\mathscr{M}_k : \mathbb{R}^n \to \mathbb{R}$ which uses approximate cutting-plane functions $\check f_k$ and $\check c_k$:

$$\mathscr{M}_k(y) = \max\Big(\check f_k(y) - \tau^k_1, \check c_k(y) - \tau^k_2 \Big) \text{ where } \left\{ \begin{array}{l} \check f_k(y) = \max\Big\{ f^j + \Big\langle g^j_{\mathtt{f}}, y - x^j \Big\rangle : j \in J^k \Big\} \\ \check c_k(y) = \max\Big\{ c^j + \Big\langle g^j_{\mathtt{c}}, y - x^j \Big\rangle : j \in J^k \Big\} . \end{array} \right. \tag{2.4}$$

To generate iterates, the algorithm chooses a prox-parameter $\mu_k > 0$ and solves the quadratic program (QP)

$$x^{k+1} = \arg\min_{y \in X} \{ \mathscr{M}_k(y) + \frac{1}{2}\mu_k \|y - \hat x^k\|^2 \} . \tag{2.5}$$

As a result, $x^{k+1} \in X$ and

$$x^{k+1} = \hat x^k - \frac{1}{\mu^k}(G^k + v^k) \quad \text{where} \quad G^k \in \partial \mathscr{M}_k(x^{k+1}) \text{ and } v^k \in N_X(x^{k+1}) , \tag{2.6}$$

where $N_X(x^{k+1})$ denotes the normal cone (of convex analysis) of X at the new iterate and $\partial \mathscr{M}_k(x^{k+1})$ the subdifferential of $\mathscr{M}_k$ at x^{k+1}. After solving the problem, the *aggregate linearization*

$$\mathtt{M}^k(y) = \mathscr{M}_k(x^{k+1}) + \left\langle G^k, y - x^{k+1} \right\rangle , \tag{2.7}$$

which is an affine function, can be defined. Clearly, because $G^k \in \partial \mathscr{M}_k(x^{k+1})$,

$$\mathtt{M}^k(y) \leq \mathscr{M}_k(y) \quad \text{for all } y \in \mathbb{R}^n . \tag{2.8}$$

The last ingredient in the bundle method is given by the *aggregate error*, defined by

$$\mathtt{E}^k = h^k_{\hat{x}^k} - \mathtt{M}^k(\hat{x}^k) - \left\langle v^k, \hat{x}^k - x^{k+1} \right\rangle . \tag{2.9}$$

In view of (2.7), for any y it holds that $G^k = \nabla \mathtt{M}^k(y)$. Therefore, because $\mathtt{M}^k(\hat{x}^k) = \mathtt{M}^k(y) + \langle G^k, \hat{x}^k - y \rangle$ with $\mathtt{M}^k \leq \mathscr{M}_k$ by (2.8), we derive the relation

$$h^k_{\hat{x}^k} - \mathtt{E}^k \leq \mathscr{M}_k(y) + \left\langle G^k + v^k, \hat{x}^k - y \right\rangle \quad \text{for all } y \in X , \tag{2.10}$$

where we used the fact that the term $\langle v^k, x^{k+1} - y \rangle$ is nonnegative for all $y \in X$, because $v^k \in N_X(x^{k+1})$.

2.2. Handling inexact oracle information. Usually, the noise introduced by the inexact evaluations is deemed "too large" when the function value at the algorithmic center is *below* the minimum model value (a situation that is impossible with an exact oracle, by convexity). For our setting, this amounts to checking if the noise measurement quantity defined below is negative:

$$h^k_{\hat{x}^k} - \left(\mathscr{M}_k(x^{k+1}) + \frac{1}{2}\mu_k \|x^{k+1} - \hat{x}^k\|^2 \right) < 0 .$$

When the relation above holds, the algorithm maintains the model and the center, and reduces the prox-parameter. The new iterate yields a smaller noise measurement quantity, thus *attenuating* the noise induced by the inexact bundle information. For the sake of numerical versatility, we consider here an alternative mechanism that is more general, and checks asymptotic satisfaction of the inequality above, based on a *relative* criterion. More precisely, noise is considered too large if, for a parameter $\beta_k \in (-1, 1)$,

$$\frac{h^k_{\hat{x}^k} - \left(\mathscr{M}_k(x^{k+1}) + \frac{1}{2}\mu_k \|x^{k+1} - \hat{x}^k\|^2 \right)}{\frac{1}{2}\mu_k \|x^{k+1} - \hat{x}^k\|^2} < -\beta_k . \tag{2.11}$$

To measure progress towards the goal of solving (1.2), a certain *predicted decrease* δ^k is employed. Usual definitions for the decrease are $\delta^k = h^k_{\hat{x}^k} - \mathscr{M}_k(x^{k+1})$, or $\delta^k = h^k_{\hat{x}^k} - \mathscr{M}_k(x^{k+1}) - \frac{1}{2}\mu_k \|x^{k+1} - \hat{x}^k\|^2$. We consider a slightly more general variant, and let

$$\delta^k = h^k_{\hat{x}^k} - \mathscr{M}_k(x^{k+1}) - \frac{1}{2}\alpha_k \mu_k \|x^{k+1} - \hat{x}^k\|^2 \quad \text{for a parameter } \alpha_k \in [0, 2]. \tag{2.12}$$

Since the numerator in (2.11) equals $\delta^k - \frac{1}{2}(1 - \alpha_k)\mu_k \|x^{k+1} - \hat{x}^k\|^2$, we see that detecting the need of a noise attenuation step amounts to checking satisfaction of the inequality

$$\delta^k < \frac{1}{2}\Big(1 - (\alpha_k + \beta_k)\Big)\mu_k \|x^{k+1} - \hat{x}^k\|^2 . \tag{2.13}$$

The choice of parameters α_k, β_k cannot be arbitrary, since we need to ensure that the nominal decrease in (2.12) is nonnegative when noise is not too large. Accordingly, we suppose that

$$\exists b > -1 \text{ and } B > 0 \text{ such that } \beta_k \in [b, 1 - \alpha_k - B] \text{ for } \alpha_k \in [0, 2] . \tag{2.14}$$

Only when noise is declared acceptable, that is when (2.13) does not hold, the algorithm examines if the new iterate is good enough to become the next center by checking

$$\begin{cases} \text{either if} & f^{k+1} \le \hat{f}^k - m\delta^k \quad \text{and} \quad c^{k+1} \le 0 & \text{when } \hat{c}^k \le 0\,, \\ \text{or if} & c^{k+1} \le \hat{c}^k - m\delta^k & \text{when } \hat{c}^k > 0\,. \end{cases} \tag{2.15}$$

When the relation holds, the iteration is declared *serious*, because it provides a new algorithmic center: $\hat{x}^{k+1} = x^{k+1}$. Otherwise, the center is maintained and the iteration is declared *null*.

The rationale behind (2.15) is to measure progress towards minimization of (1.2) by focusing either in reducing the objective value without losing feasibility if the center is approximately feasible. Otherwise, when $\hat{c}^k > 0$, the emphasis is put in reducing infeasibility by checking the second condition in (2.15).

The parameters $\alpha_k\,, \beta_k$ in the criterion (2.13) make it possible to control the relation between noise attenuation and descent, a flexibility that can help the numerical performance of the algorithm. More specifically, to progress towards a solution, it is preferable for the algorithm to:

- make more serious iterations, because serious iterates converge to a solution, and
- have few noise attenuation steps. Noise attenuation steps are undesirable to occur often, because they prevent the algorithm from having "true" iterates, for which to check the descent condition.
- However, checking (2.13) does not involve any f/c-oracle calculation at x^{k+1}, and can therefore be considered an inexpensive test.

The flexibility introduced by the additional parameters α_k, β_k allows the user to seek for a trade-off between the time spent in oracle calculations and the CPU time required for the algorithm to find a solution to (1.2). By (2.15), more serious iterations are achieved by taking larger α_k's (yielding smaller δ^k's), while a larger β_k reduces the left handside term in (2.13), making less likely noise attenuation. Our numerical experience in Section 6 shows how different choices of these parameters impact the numerical performance, both in terms of CPU time and accuracy.

Both the prox-parameter μ_k and penalty parameters ρ_k, σ_k require specific updating in order to assure convergence, though there is still much freedom. For the prox-parameter, we suggest the following updating rule that uses positive constants $\mu_{\max}$ and Δ:

$$\mu_{k+1} \le \mu_{\max} < +\infty \qquad \text{if iteration } k \text{ was declared serious} \tag{2.16a}$$

$$\mu_{k+1} \le \mu_k - \Delta < \mu_k \qquad \text{if iteration } k \text{ resulted in noise attenuation} \tag{2.16b}$$

$$\mu_{k+1} \in [\mu_k, \mu_{\max}] \qquad \text{if iteration } k \text{ was declared null.} \tag{2.16c}$$

The penalty parameters update depends on a constant $\underline{s} > 0$ and satisfies

$$0 \le \sigma_{k+1} \le 1 \text{ and } \rho_{k+1} \ge 0 \text{ satisfy } 1 - \sigma_{k+1} + \rho_{k+1} \ge \underline{s} > 0\,. \tag{2.17}$$

We now list some consequences from the various definitions above obtained with simple algebraic manipulations. First, by (2.7) written with $y = \hat{x}^k$ and (2.12),

$$h^k_{\hat{x}^k} - \mathtt{M}^k(\hat{x}^k) + \left\langle G^k, \hat{x}^k - x^{k+1} \right\rangle = \delta^k + \frac{1}{2}\alpha_k\mu_k\|x^{k+1} - \hat{x}^k\|^2\,.$$

Together with (2.9) and (2.6) we see that

$$\mathtt{E}^k = \delta^k - \frac{2-\alpha_k}{2}\mu_k\|x^{k+1} - \hat{x}^k\|^2 \quad \text{and, therefore,} \quad \mathtt{E}^k \le \delta^k \text{ at all iterations} \tag{2.18}$$

because $\alpha_k \le 2$, by (2.14).

Second, by adding δ^k to both members of the identity (2.18), we see that

$$\text{inequality (2.13) holds} \iff \delta^k + \mathtt{E}^k < -\frac{(\alpha_k + 2\beta_k)}{2}\mu_k\|x^{k+1} - \hat{x}^k\|^2\,. \tag{2.19}$$

2.3. A Nonsmooth Optimization Solver for Inexact Oracles. We now give our bundle algorithm for solving problem (1.2).

Algorithm 2.1. *We assume that an oracle as in* (2.2)*, possibly of upper type, is available.*

Step 0 (Input and Initialization) *Select an initial starting point x^0, a stopping tolerance* TOL≥ 0 *and an Armijo-like parameter $m \in (0,1)$. Call* (2.2) *to compute f^0, c^0 as well as ${g_{\mathtt{f}}}^0$ and ${g_{\mathtt{c}}}^0$. Initialize the iteration counter $k=0$, the bundle index set $J^0 := \{0\}$, and the first stability center $\hat{x}^0 := x^0$. Choose the starting prox-parameter $\mu_0 > 0$, parameters α_0, β_0 satifying* (2.14)*, and penalties $\rho_0, \sigma_0 \geq 0$ satisfying* (2.17) *below.*

Step 1 (Model Generation and QP Subproblem) *Having the current algorithmic center $\hat{x}^k$, the bundle $\mathscr{B}^k$, the prox-parameter μ_k and the penalties ρ_k, σ_k, define the convex piecewise linear model function $\mathscr{M}_k$ and compute $x^{k+1} = \arg\min\{\mathscr{M}_k(y) + \frac{1}{2}\mu_k\|y-\hat{x}^k\|^2 : y \in X\}$. Define the predicted decrease δ^{k+1} as in* (2.12)*.*

Step 2 (Noise attenuation test)
If condition (2.13) *is true, noise is too large: decrease the prox-parameter as in* (2.16b)*; maintain the center, the bundle, and the penalties:*

$$\left(\hat{x}^{k+1}, \mathscr{B}^{k+1}, \rho_{k+1}, \sigma_{k+1}\right) = \left(\hat{x}^k, \mathscr{B}^k, \rho_k, \sigma_k\right);$$

choose parameters $\alpha_{k+1}, \beta_{k+1}$ satisfying (2.14)*, and loop to Step1.*
Otherwise, if (2.13) *does not hold, proceed to Step 3.*

Step 3 (Stopping Test and New Oracle Information) *If $\delta^{k+1} \leq$*TOL *then stop. Otherwise call the oracle to obtain $f^{k+1}, c^{k+1}, {g_{\mathtt{f}}}^{k+1}$ and ${g_{\mathtt{c}}}^{k+1}$.*

Step 4 (Serious step test) *Check the descent condition* (2.15)*. If this condition is true, declare a serious iteration and set $\hat{x}^{k+1} = x^{k+1}$. Otherwise, declare a null step and maintain the center: $\hat{x}^{k+1} = \hat{x}^k$.*

Step 5 (Bundle Management and updates) *Choose a new prox-parameter μ_{k+1} satisfying* (2.16a) *or* (2.16c) *if the iteration was declared serious or null, respectively. In all cases choose parameters $\alpha_{k+1}, \beta_{k+1}$ satisfying* (2.14) *and penalties ρ_{k+1}, σ_{k+1} satisfying* (2.17)*. Define the new bundle $\mathscr{B}^{k+1}$, for example by appending to the index set the last iterate information: $J^{k+1} = J^k \cup \{k+1\}$. Increase k by 1 and loop to Step 1.*

Both in Step 2 and Step 5 leave some freedom for choosing the new bundle $\mathscr{B}^{k+1}$. In Step 2, the conservative choice of keeping the same cutting-plane models for both f and c seems reasonable. Choices for Step 5 are discussed after Lemma 3.1, 3.2, and 3.3, for the respective cases of serious, noisy, and null iterations.

The main purpose of conditions (2.17) is to ensure satisfaction of the relations stated in the proposition below.

Proposition 2.2 (Consequences of penalization updates). *When Algorithm 2.1 uses the rule* (2.17) *for the penalties, the following holds.*

– At every iteration k,

$$h^k_{\hat{x}^k} = 0 \quad \text{if } \hat{c}^k \leq 0 \quad \text{and} \quad h^k_{\hat{x}^k} = \hat{c}^k(1-\sigma_k) \geq 0 \quad \text{otherwise}.$$

– At every iteration k declared null, the inequality $h^k_{x^{k+1}} > h^k_{\hat{x}^k} - m\delta^k$ is satisfied.
– Suppose that, in addition, the oracle (2.2) *is bounded, in the sense that*

$$\text{the inexact values } \{f^k, c^k\} \text{ and } \{\|g^k_{\mathtt{f}}\|, \|g^k_{\mathtt{c}}\|\} \text{ are bounded for every sequence } \{x^k\} \subset X. \tag{2.20}$$

Then for any iteration k declared null or needing noise attenuation there exist constants $M, M' > 0$ such that

$$\mathscr{M}_k(\hat{x}^k) \leq \max(M - \hat{f}^k, M) \quad \text{and} \quad h^k_{\hat{x}^k} \leq M'. \tag{2.21}$$

Proof. From (2.3), it readily follows that $\hat{c}^k \leq 0$ implies $\tau^k_1 = \hat{f}^k$, whereas $\hat{c}^k > 0$ implies $\tau^k_1 = \hat{f}^k + \rho_k\hat{c}^k$ and $\tau^k_2 = \sigma_k\hat{c}^k$. As a consequence $\hat{c}^k \leq 0$ implies $h^k_{\hat{x}^k} = 0$. Inversely when $\hat{c}^k > 0$, we have $h^k_{\hat{x}^k} = \hat{c}^k \max(-\rho_k, 1-\sigma_k) = \hat{c}^k(1-\sigma_k) \geq 0$ because $\sigma_k \in [0,1]$ and $\rho_k \geq 0$ by (2.17).

To prove the second item, let us first assume $\hat{c}^k > 0$. Then negating (2.15) and using the above derived identity for $h^k_{\hat{x}^k}$ gives

$$h^k_{x^{k+1}} = \max(f^{k+1} - \hat{f}^k - \rho_k\hat{c}^k, c^{k+1} - \sigma_k\hat{c}^k) \geq c^{k+1} - \sigma_k\hat{c}^k > \hat{c}^k(1-\sigma_k) - m\delta^k = h^k_{\hat{x}^k} - m\delta^k.$$

When $\hat{c}^k \leq 0$, negating (2.15) implies that either $f^{k+1} > \hat{f}^k - m\delta^k$ or $c^{k+1} > 0$. In the first case we establish:

$$h^k_{x^{k+1}} = \max(f^{k+1} - \hat{f}^k, c^{k+1}) \geq f^{k+1} - \hat{f}^k > -m\delta^k,$$

as was to be shown, since $h^k_{\hat{x}^k} = 0$. In the second case, since k is a null step, (2.13) does not hold and $\delta^k > 0$ as a consequence. We thus establish $h^k_{x^{k+1}} = \max(f^{k+1} - \hat{f}^k, c^{k+1}) \geq c^{k+1} > 0 \geq h^k_{\hat{x}^k} - m\delta^k$ as was to be shown.

Finally, to see (2.21), notice that $x^k \in X$ with X bounded, so (2.20) ensures that each linearization $f^k + \langle g^k_{\mathtt{f}}, \cdot - x^k \rangle$ (or its c-counterpart) is bounded over X. In particular, both $\check{f}_k(\hat{x}^k)$ and $\check{c}_k(\hat{x}^k)$ are bounded by some constant M. In view of the model definition (2.4) and (2.3), $\mathscr{M}_k(\hat{x}^k) \leq \max(M - \tau^k_1, M - \tau^k_2) = \max(M - \hat{f}^k - \rho_k \max(\hat{c}^k, 0), M - \sigma_k \max(\hat{c}^k, 0))$. The bound for $\mathscr{M}_k(\hat{x}^k)$ follows, because the penalty terms are nonnegative by (2.17). An analogous reasoning gives the bound for $h^k_{\hat{x}^k}$. □

3. Asymptotic Analysis. We now analyse the different cases that can arise when algorithm 2.1 loops forever, i.e., $k \to \infty$. Then only one of the following mutually exclusive cases can occur:

- either there are infinitely many serious iterates,
- or the stability center remains unchanged after a finite number of iterations. In this case,
  - either there is an infinite number of noise attenuation steps,
  - or there is a finite number of noise attenuation steps and eventually only null steps are done.

The first case is considered in Lemma 3.1, the second case in Lemma 3.2 and the last case in Lemma 3.3.

LEMMA 3.1 (Infinitely many serious iterations). *Consider solving* (1.2) *with Algorithm 2.1 using an oracle satisfying* (2.2) *and* (2.20), *with parameters* α_k, β_k *satisfying* (2.14).

Let K_s *denote the set gathering indices of serious iterations. If there are infinitely many of such indices, and the prox-parameter sequence satisfies* (2.16a), *then*

$$h^k_{\hat{x}^k} \leq \mathscr{M}_k(y) + o(1/k) \quad \textit{for all } y \in X \textit{ and } k \in K_s \textit{ sufficiently large.}$$

Proof. Consider $k \in K_s$. We first show that $\delta^k \to 0$. If for all serious steps $c^{k+1} > 0$ then (2.15) implies that

$$\hat{c}^{k+1} = c^{k+1} \leq c^k - m\delta^k = \hat{c}^k - m\delta^k, \tag{3.1}$$

noting that $\delta^k \geq 0$, because serious steps can only take place when no noise attenuation occurs. Since the nonincreasing sequence $\{\hat{c}^k\}_{K_s}$ is bounded below by 0, it converges. From (3.1) we deduce that $0 \leq \delta^k \leq \frac{c^k - c^{k+1}}{m}$. Since $\{\hat{c}^k\}_{K_s}$ converges this gives gives that $\delta^k \to 0$ as $K_s \ni k \to \infty$.

Otherwise, there is some $\bar{k} \in K_s$ such that $\hat{c}_{\bar{k}} \leq 0$. In view of (2.15), all subsequent serious iterates are feasible: for each serious step $\bar{k} \leq k \in K_s$ we have $\hat{c}^{k+1} = c^{k+1} \leq 0$ and

$$\hat{f}^{k+1} \leq \hat{f}^k - m\delta^k \quad \text{with} \quad \delta^k \geq 0. \tag{3.2}$$

Thus, the nonincreasing sequence $\{\hat{f}^k\}_{\bar{k} \leq k \in K_s}$ is either unbounded below or converges. The first case is ruled out by (2.20). As for the second case, it implies that $\delta^k \to 0$ since $0 \leq \delta^k \leq \frac{f^k - f^{k+1}}{m}$.

We now show that both $G^k + v^k \to 0$ and $\mathtt{E}^k \to 0$. Since $(1 - (\alpha_k + \beta_k)) \geq B > 0$ by (2.14), the negation of (2.13) and (2.6) imply that

$$\begin{aligned} 0 \leftarrow \delta^k &\geq \frac{1}{2}\Big(1 - (\alpha_k + \beta_k)\Big)\mu_k \|x^{k+1} - \hat{x}^k\|^2 = \frac{1}{2\mu_k}\Big(1 - (\alpha_k + \beta_k)\Big)\|G^k + v^k\|^2 \\ &\geq \frac{B}{2\mu_k}\|G^k + v^k\|^2 \geq \frac{B}{2\mu_{max}}\|G^k + v^k\|^2 \geq 0 \end{aligned}$$

where we used (2.16a) for the last inequality. As a result, both $\|G^k + v^k\|^2/\mu_k \to 0$ and $G^k + v^k \to 0$ as $K_s \ni k \to \infty$.

Since $\alpha_k \geq 0$ and $\alpha_k + \beta_k \leq 1 - B$ by (2.14), we deduce that $-(\alpha_k + 2\beta_k)/2 = -(\alpha_k + \beta_k) + \alpha_k/2 \geq B - 1$. To show that the aggregate error goes to 0, pass to the limit in the negation of (2.19) and use the above estimate to see that

$$\lim_{k \in K_s} \mathtt{E}^k \geq (B-1) \lim_{k \in K_s} \mu_k \|x^{k+1} - \hat{x}^k\|^2.$$

Since by (2.6), $\mu_k\|x^{k+1}-\hat{x}^k\|^2 = \|G^k+v^k\|^2/\mu_k$ and we have just shown this term vanishes asymptotically, the error limit is nonnegative. Then $\mathrm{E}^k \to 0$, because $\mathrm{E}^k \le \delta^k$ by (2.18) and $\delta^k \to 0$.

The stated inequality holds follows from (2.10) by boundedness of X and the fact that both G^k+v^k and $\mathrm{E}^k \to 0$. □

The analysis above shows that Step 5 of Algorithm 2.1 can freely manage the bundle at serious iterations. Of course, a richer bundle yields better cutting-plane models for f and c, so having larger index-sets J^k should improve the speed of the method (keeping in mind that large sets J^k make it harder to solve the QP subproblem).

Notice also that in the inequality in Lemma 3.1 the remainder $o(1/k)$ corresponds in fact to both G^k+v^k and E^k going to zero. Similar relations will hold when there is a finite number of serious iterations, as shown below.

In the next two cases, eventually no more serious steps occur and the algorithmic center remains unchanged. As a result, there exists $\hat{k}$ and $\hat{x}$ such that $\hat{x}^k=\hat{x}$ for all $k>\hat{k}$. Unlike Lemma 3.1, the results below rely on conditions (2.17), required for the penalty parameters defining the inexact improvement function (2.3).

LEMMA 3.2 (Infinitely many noisy iterations). *Consider solving* (1.2) *with Algorithm 2.1 using an oracle satisfying* (2.2) *and* (2.20)*, with parameters* α_k, β_k *satisfying* (2.14) *and penalties as in* (2.17)*.*

Suppose at iteration $\hat{k}$ *there is a last serious iterate* $\hat{x}$ *and let* $K_a := \{k>\hat{k}:\ (2.13)\ \textit{holds}\}$*. If there are infinitely many of such indices and* (2.16b) *holds then*

$$h^k_{\hat{x}^k} = h^k_{\hat{x}} \le \mathscr{M}_k(y)+o(1/k) \quad \textit{for all } y\in X \textit{ and } k\in K_a \textit{ sufficiently large.}$$

Proof. Consider $k\in K_a$. By (2.18) and (2.13) we obtain the following estimate

$$2\mathrm{E}^k = 2\delta^k - (2-\alpha_k)\mu_k\|x^{k+1}-\hat{x}^k\|^2 < -(1+\beta_k)\mu_k\|x^{k+1}-\hat{x}^k\|^2 \le 0, \tag{3.3}$$

since $\beta_k \ge b > -1$ by (2.14). Since $\mathrm{E}^k\le 0$, $h^k_{\hat{x}^k}-\mathrm{E}^k \ge h^k_{\hat{x}^k}$, and with (2.10) we see that

$$h^k_{\hat{x}^k} \le \mathscr{M}_k(y) + \left\langle G^k+v^k, \hat{x}-y\right\rangle \quad \text{for all } y\in X.$$

Since the set X is bounded, the stated inequality would hold if $G^k+v^k\to 0$. To show this result, first note that (2.9) and (2.7) imply that

$$-\mathrm{E}^k = \mathscr{M}_k(x^{k+1}) + \left\langle G^k+v^k, \hat{x}-x^{k+1}\right\rangle - h^k_{\hat{x}^k} \le \mathscr{M}_k(\hat{x}) - h^k_{\hat{x}^k},$$

where the inequality comes from (2.6). Let $\hat{f}$ denote the f-value computed by the oracle at $\hat{x}$. By the first item in Proposition 2.2, $h^k_{\hat{x}^k}\ge 0$ because $\sigma_k\le 1$ by (2.17), so $-\mathrm{E}^k \le \mathscr{M}_k(\hat{x})$. Since $\mathscr{M}_k(\hat{x}) \le \max(\hat{f}-M,\hat{f})$ by the third item in the proposition, for some constant $\hat{M}$

$$-\mathrm{E}^k \le \hat{M} \text{ for all } k\in K_a.$$

From (3.3) and the condition $\beta_k\ge b$ from (2.14) we obtain the inequality $\mathrm{E}^k < -\frac{1}{2}(1+b)\mu_k\|x^{k+1}-\hat{x}^k\|^2$. Moreover, using (2.6) and the fact that $1+b>0$ by (2.14), this means that

$$\frac{2}{1+b}\mu_k\mathrm{E}^k < -\|G^k+v^k\|^2$$

or, equivalently, that

$$0\le\|G^k+v^k\| \le \sqrt{-\frac{2}{1+b}\mu_k\mathrm{E}^k} \le \sqrt{\frac{2}{1+b}\mu_k\hat{M}}.$$

Since the update in (2.16b) ensures that the sequence $\{\mu_k\}_{k\in K_a}$ is strictly decreasing, when $K_a\ni k\to\infty$ we obtain that $\mu_k\to 0$ and $G^k+v^k\to 0$, as desired. □

For the result above to hold, Step 2 in Algorithm 2.1 only needs the sequence $\{\mu_k\}_{K_a}$ to be strictly decreasing, and the left bound in (2.21) to hold. So, as for the case of infinitely many serious iterations, there is freedom in

how the bundle is managed when noise is deemed too large. Since in this case there is no new oracle information, it seems reasonable to maintain the current bundle. Note also that the remainder term in Lemma 3.2 corresponds to having $G^k + v^k \to 0$ and $\mathtt{E}^k \leq 0$ (instead of $\mathtt{E}^k \to 0$ like in Lemma 3.1).

The final case refers to finitely many serious and noise attenuation steps, which implies the algorithm makes infinitely many consecutive null steps. For this case, the model is required to satisfy the following conditions whenever the iteration k is declared null:

$$\mathscr{M}_{k+1}(y) \geq \mathtt{M}^k(y) \text{ and} \tag{3.4a}$$

$$\mathscr{M}_{k+1}(y) \geq \max\Big(f^{k+1} + \Big\langle g_{\mathtt{f}}^{k+1}, y - x^{k+1} \Big\rangle - \tau_1^k, c^{k+1} + \Big\langle g_{\mathtt{c}}^{k+1}, y - x^{k+1} \Big\rangle - \tau_2^k \Big) \tag{3.4b}$$

The result below uses standard arguments in bundle methods [4], taking advantage of the boundedness relations in (2.20) and (2.21) to adapt those arguments to the inexact improvement function setting.

LEMMA 3.3 (Infinitely many consecutive null iterations). *Consider solving* (1.2) *with Algorithm 2.1 using an oracle satisfying* (2.2) *and* (2.20), *with parameters* α_k, β_k *satisfying* (2.14) *and penalties as in* (2.17).

Suppose at iteration $\hat{k}$ *there is a last serious step, denoted by* $\hat{x}$ *and after an iteration* $\bar{k} > \hat{k}$ *there are no more noise attenuation steps: eventually only null steps occur. Let* K_n *denote the set gathering indices of iterations larger than* $\bar{k}$. *If there are infinitely many of such indices and both* (2.16c) *and* (3.4) *hold, then*

$$h_{\hat{x}^k}^k = h_{\hat{x}}^k \leq \mathscr{M}_k(y) + o(1/k) \text{ for all } y \in X \text{ and } k \in K_n \text{ sufficiently large.}$$

Furthermore, $x^k \to \hat{x}$ *as* $K_n \ni k \to \infty$.

Proof. Consider $k \in K_n$. Once again, the stated inequality will follow from (2.10) by boundedness of X, if we show that both $G^k + v^k \to 0$ and $\mathtt{E}^k \to 0$. In turn, these results follow from showing that $\delta^k \to 0$.

To see that $\delta^k \to 0$, we start by expanding squares and using (2.6) to write the identity

$$2\langle G^k + v^k, y - x^{k+1} \rangle = 2\mu_k \langle \hat{x} - x^{k+1}, y - x^{k+1} \rangle = \mu_k \|x^{k+1} - \hat{x}\|^2 + \mu_k \|y - x^{k+1}\|^2 - \mu_k \|y - \hat{x}\|^2 \tag{3.5}$$

for all $y \in \mathbb{R}^n$. Since the function $\mathtt{M}^k$ is affine with gradient G^k, for all $y \in \mathbb{R}^n$

$$\begin{aligned} \mathtt{M}^k(y) &= \mathtt{M}^k(x^{k+1}) + \langle G^k, y - x^{k+1} \rangle \\ &= \mathtt{M}^k(x^{k+1}) + \langle G^k, y - x^{k+1} \rangle + \langle v^k, y - x^{k+1} \rangle - \langle v^k, y - x^{k+1} \rangle \\ &= \mathtt{M}^k(x^{k+1}) + \frac{1}{2}\mu_k \|x^{k+1} - \hat{x}\|^2 + \frac{1}{2}\mu_k \|y - x^{k+1}\|^2 - \frac{1}{2}\mu_k \|y - \hat{x}\|^2 - \langle v^k, y - x^{k+1} \rangle \\ &= \mathtt{V}^k + \frac{1}{2}\mu_k \|y - x^{k+1}\|^2 - \frac{1}{2}\mu_k \|y - \hat{x}\|^2 - \langle v^k, y - x^{k+1} \rangle, \end{aligned} \tag{3.6}$$

where in the last equality we define $\mathtt{V}^k := \mathscr{M}_k(x^{k+1}) + \frac{1}{2}\mu_k \|x^{k+1} - \hat{x}\|^2$ and use the relation $\mathtt{M}^k(x^{k+1}) = \mathscr{M}_k(x^{k+1})$ from (2.7) (the value $\mathtt{V}^k$ is the optimal value of the QP subproblem (2.5) defining x^{k+1}). Since $v^k \in N_X(x^{k+1})$ and $y \in X$, the term $-\langle v^k, y - x^{k+1} \rangle \geq 0$ and, hence, we derive from (3.6) that

$$\forall y \in X \quad \mathtt{M}^k(y) + \frac{1}{2}\mu_k \|y - \hat{x}\|^2 \geq \mathtt{V}^k + \frac{1}{2}\mu_k \|y - x^{k+1}\|^2 . \tag{3.7}$$

By evaluating at $y = \hat{x} \in X$, using (2.8) and the third item in Proposition 2.2, there exists $\hat{M} > 0$ such that

$$\mathtt{V}^k + \frac{1}{2}\mu_k \|\hat{x} - x^{k+1}\|^2 \leq \mathtt{M}^k(\hat{x}) \leq \mathscr{M}_k(\hat{x}) \leq \hat{M} .$$

In particular, the sequence $\{\mathtt{V}^k\}$ is bounded from above.

Assumption (3.4a) in (3.7) yields that

$$\mathscr{M}_{k+1}(y) + \frac{1}{2}\mu_k \|y - \hat{x}\|^2 \geq \mathtt{V}^k + \frac{1}{2}\mu_k \|y - x^{k+1}\|^2$$

for all $y \in X$. Since $\mu_{k+1} \geq \mu_k$ by (2.16c), by evaluating the inequality above at $y = x^{k+2}$ we see that

$$\mathtt{V}^{k+1} \geq \mathtt{V}^k + \frac{1}{2}\mu_k \|x^{k+2} - x^{k+1}\|^2 ,$$

and, being bounded above, the non-decreasing sequence $\{\mathtt{V}^k\}$ converges. Since the righthand side is larger than $\mathtt{V}^k$, we conclude that

$$\mathtt{V}^{k+1} - \left(\mathtt{V}^k + \frac{1}{2}\mu_k \|x^{k+2} - x^{k+1}\|^2\right) \to 0 \quad \text{and, furthermore,} \quad \|x^{k+2} - x^{k+1}\|^2 \to 0 \tag{3.8}$$

because $\mu_k \geq \mu_{k+1}^-$ by (2.16c).

Recall that condition (2.14) implies that $\delta^k \geq 0$ when (2.13) fails, i.e., no noise is detected. The descent test also fails, so by the second item in Proposition 2.2,

$$h^k_{x^{k+1}} > h^k_{\hat{x}^k} - m\delta^k = h^k_{\hat{x}} - m\delta^k ,$$

because the algorithmic center is fixed. Adding δ^k to both terms and using the definition in (2.12) we obtain that

$$\begin{aligned} 0 \leq (1-m)\delta^k < \delta^k + h^k_{x^{k+1}} - h^k_{\hat{x}^k} \\ &= h^k_{x^{k+1}} - \mathscr{M}_k(x^{k+1}) - \frac{1}{2}\alpha_k\mu_k\|x^{k+1} - \hat{x}\|^2 \\ &\leq h^k_{x^{k+1}} - \mathscr{M}_k(x^{k+1}) \\ &= h^k_{x^{k+1}} - \mathscr{M}_{k+1}(x^{k+2}) + \mathscr{M}_{k+1}(x^{k+2}) - \mathscr{M}_k(x^{k+1}) . \end{aligned}$$

Writing (3.4b) for $y = x^{k+2}$ and using the Cauchy-Schwarz inequality yields that

$$\mathscr{M}_{k+1}(x^{k+2}) \geq \max(f^{k+1} - \tau_1^k - \|g_{\mathtt{f}}^{k+1}\|\|x^{k+2} - x^{k+1}\|, c^{k+1} - \tau_2^k - \|g_{\mathtt{c}}^{k+1}\|\|x^{k+2} - x^{k+1}\|) .$$

By (2.20), there exists a constant Γ such that $\|g_{\mathtt{f}}^{k+1}\|, \|g_{\mathtt{c}}^{k+1}\| \leq \Gamma$ and, hence,

$$\mathscr{M}_{k+1}(x^{k+2}) \geq \max(f^{k+1} - \tau_1^k, c^{k+1} - \tau_2^k) - \Gamma\|x^{k+2} - x^{k+1}\| .$$

The first righthand side term above equals $h^k_{x^{k+1}}$, by (2.3). Continuing and using the definition of $\mathtt{V}^k$ and (2.16c),

$$\begin{aligned} 0 \leq (1-m)\delta^k &\leq \Gamma\|x^{k+2} - x^{k+1}\| + \mathscr{M}_{k+1}(x^{k+2}) - \mathscr{M}_k(x^{k+1}) \\ &= \Gamma\|x^{k+2} - x^{k+1}\| + \mathtt{V}^{k+1} - \frac{1}{2}\mu_{k+1}\|x^{k+2} - \hat{x}\|^2 - \mathtt{V}^k + \frac{1}{2}\mu_k\|x^{k+1} - \hat{x}\|^2 \\ &= \Gamma\|x^{k+2} - x^{k+1}\| + \mathtt{V}^{k+1} - \mathtt{V}^k - \frac{1}{2}\mu_k\|x^{k+2} - \hat{x}\|^2 + \frac{1}{2}\mu_k\|x^{k+1} - \hat{x}\|^2 \\ &= \Gamma\|x^{k+2} - x^{k+1}\| + \mathtt{V}^{k+1} - (\mathtt{V}^k + \frac{1}{2}\mu_k\|x^{k+2} - x^{k+1}\|^2) \\ &\qquad + \frac{1}{2}\mu_k\left(\|x^{k+2} - x^{k+1}\|^2 - \|x^{k+2} - \hat{x}\|^2 + \|x^{k+1} - \hat{x}\|^2\right) . \end{aligned}$$

Following (3.5), we can observe that the last three terms equal $\mu_k\langle x^{k+2} - x^{k+1}, \hat{x} - x^{k+1}\rangle$. By (2.6), $\mu_k(\hat{x} - x^{k+1}) = G^k + v^k$ and, since $v^k \in N_X(x^{k+1})$ and $x^{k+2} \in X$,

$$\mu_k\left\langle x^{k+2} - x^{k+1}, \hat{x} - x^{k+1}\right\rangle = \left\langle x^{k+2} - x^{k+1}, G^k + v^k\right\rangle \leq \left\langle x^{k+2} - x^{k+1}, G^k\right\rangle \leq \|G^k\|\|x^{k+2} - x^{k+1}\| .$$

Since $G^k \in \partial\mathscr{M}_k(x^{k+1}) \subset conv\{g_{\mathtt{f}}^j, g_{\mathtt{c}}^j : j \in J_k\}$, assumption (2.20) implies that $\|G^k\| \leq \Gamma$ and, hence,

$$0 \leq (1-m)\delta^k \leq 2\Gamma\|x^{k+2} - x^{k+1}\| + \mathtt{V}^{k+1} - (\mathtt{V}^k + \frac{1}{2}\mu_k\|x^{k+2} - x^{k+1}\|^2) .$$

In view of (3.8), the right handside above tends to zero. Since $m \in (0,1)$, $\delta^k \to 0$, as desired.

By the negation of (2.13), together with $1-(\alpha_k+\beta_k)\geq B>0$ from (2.14) and $\mu_k\geq\mu_{k+1}^-$ from (2.16c) imply,

$$0\leftarrow\delta^k\geq\frac{1}{2}\mu_{k+1}^-B\|x^{k+1}-\hat{x}\|^2\geq 0\,,$$

so $x^{k+1}\to\hat{x}$, as stated. Since by (2.6) $G^k+v^k=\mu_k(\hat{x}-x^{k+1})$ and by (2.16c) $\mu_k\leq\mu_{\max}$, we obtain that

$$0\leq\frac{1}{\mu_{\max}}\|G^k+v^k\|\leq\frac{1}{\mu_k}\|G^k+v^k\|=\|x^{k+1}-\hat{x}\|\to 0 \text{ and, hence, } G^k+v^k\to 0\,.$$

Finally, using in (2.18) that $\alpha_k\geq 0$ by (2.14) gives

$$0\leftarrow\delta^k\geq\mathrm{E}^k=\delta^k-\frac{2-\alpha_k}{2}\mu_k\|x^{k+1}-\hat{x}\|^2\geq\delta^k-\mu_k\|x^{k+1}-\hat{x}\|^2\geq\delta^k-\mu_{\max}\|x^{k+1}-\hat{x}\|^2.$$

Since both righthand side terms go to zero, so does E^k and the proof is finished. □

For the result above to hold, at null steps the bundle needs to be managed in a manner ensuring the relations (3.4). Condition (3.4b) holds if the last generated information enters the bundle, that is,

$$k+1\in J^{k+1}\quad\text{ which is equivalent to having }\left(x^{k+1},f^{k+1},c^{k+1},g_{\mathtt{f}}{}^{k+1},g_{\mathtt{c}}{}^{k+1}\right)\in\mathscr{B}^{k+1}\,.$$

As for (3.4a), the inequality is typically ensured by introducing aggregate information in the bundle. In our improvement function setting, this can be done by splitting the aggregate information into their f and c parts. Specifically, in view of the model definition (2.4), there is a multiplier $\lambda^k\in[0,1]$ such that

$$\mathscr{M}_k(x^{k+1})=\lambda^k(\check{f}_k(x^{k+1})-\tau_1^k)+(1-\lambda^k)(\check{c}_k(x^{k+1})-\tau_2^k)\,,$$

and

$$G^k=\lambda^kG_f^k+(1-\lambda^k)G_c^k\quad\text{ with }G_f^k\in\partial\check{f}_k(x^{k+1})\,,G_c^k\in\partial\check{c}_k(x^{k+1})\,.$$

For (3.4a) to hold it is then enough to take

$$\left(x^{k+1},\check{f}_k(x^{k+1}),\check{c}_k(x^{k+1}),G_f^k,G_c^k\right)\in\mathscr{B}^{k+1}\,.$$

Similar calculations can be derived for *economic* bundles which, in the $(x^j,f^j,c^j,g_{\mathtt{f}}^j,g_{\mathtt{c}}^j)$, replace the knowledge of the vector x^j by two scalars, the linearization errors for f and c at $\hat{x}^k$. We refer to [34] for more details.

Finally, like in the case of infinitely many serious steps, the remainder term corresponds to having both $G^k+v^k\to 0$ and $\mathrm{E}^k\to 0$. For this reason, instead of the stopping criterion in Step 3 of Algorithm 2.1, one could stop when G^k+v^k is sufficiently small, as long as E^k is also small (serious and null cases) or nonpositive (noisy case).

4. Link with the Original Problem.

In Section 3 we established the limiting behaviour of Algorithm 2.1. We still need to analyze in which sense the method converges to an approximate solution to problem (1.2). It is at this stage that the oracle errors play a major role. Our results below show what can be expected in terms of solving (1.2) for oracles that are of upper or lower type. Partly assymptotically exact and exact oracles are also considered.

4.1. Convergence Results. We start with a general result, suitable for oracles yielding inexact linearizations that may lie *above* the function. Specifically, we suppose that for any $x^k\in X$ the oracle output (2.2) is of *upper* type, in the sense that errors ε^k at iteration k satisfy:

$$\forall y\in X\quad\left\{\begin{array}{l}f^k+\langle g_{\mathtt{f}}^k,y-x^k\rangle\leq f(y)+\varepsilon^k\\ c^k+\langle g_{\mathtt{c}}^k,y-x^k\rangle\leq c(y)+\varepsilon^k\end{array}\right.\tag{4.1}$$

We define the asymptotic error ε as

$$\varepsilon:=\limsup\varepsilon^k.\tag{4.2}$$

Theorem 4.1 (Asymptotic Bounds for Upper Oracles). *Consider solving* (1.2) *with Algorithm 2.1 using an oracle* (2.2) *such that* (2.20) *and* (4.1) *hold, with parameters α_k, β_k satisfying* (2.14) *and penalties as in* (2.17). *Suppose the prox-parameter is updated according to* (2.16), *and the model satisfies* (3.4).

When the algorithm loops forever, for any accumulation point $\bar{x}$ of the (bounded) sequence $\{\hat{x}^k\}$ and its limiting inexact f/c-values and parameters $(\bar{f}, \bar{c}, \bar{\rho}, \bar{\sigma})$ it holds that

$$\forall y \in X \quad \max(\bar{c},0)(1-\bar{\sigma}) \le \max\Big(f(y)-\bar{f}-\bar{\rho}\max(\bar{c},0), c(y)-\bar{\sigma}\max(\bar{c},0)\Big)+\varepsilon. \tag{4.3}$$

Moreover, taking ε as defined in (4.2), *it holds that:*

(i) If $\bar{c}>0$ then

$$\exists R : \bar{\rho} \ge R \Longrightarrow \bar{c} \le c(y)+\varepsilon \text{ for all } y \in X.$$

(ii) If $\bar{c} \le 0$ and the set $X_\varepsilon = \{y \in X : c(y) \le -2\varepsilon\}$ is not empty, then

$$\bar{f} \le f(y)+\varepsilon \text{ for all } y \in X_\varepsilon.$$

Proof. When the algorithm loops forever, one of the index sets K_s, K_a, K_n, defined respectively in Lemma 3.1, 3.2, 3.3, is infinite. Since $\hat{x}^k \in X$ and X is compact, for any of such sets there exists a subset K' such that $\{\hat{x}^k\}_{k\in K'} \to \bar{x}$, recalling that when $K'=K_a$ and $K'=K_n$ eventually $\hat{x}^k = \hat{x} = \bar{x}$ remains fixed. By the same three Lemmas, and the first item in Proposition 2.2,

$$\max(\hat{c}^k,0)(1-\sigma_k) = h^k_{\hat{x}^k} \le \mathscr{M}_k(y)+o(1/k) \text{ for all } y \in X \text{ and } k \in K' \text{ sufficiently large.}$$

By (4.1), $\mathscr{M}_k(y) \le H_{\tau^k}(y)+\varepsilon^k$ for the exact improvement function (2.1) written with target $\tau=\tau^k$, so

$$\forall y \in X \quad \max(\hat{c}^k,0)(1-\sigma_k) \le \max(f(y)-\tau^k_1, c(y)-\tau^k_2)+o(1/k)+\varepsilon^k. \tag{4.4}$$

By (2.3), the target is $\tau^k = (\hat{f}^k+\rho_k\max(\hat{c}^k,0), \sigma_k\max(\hat{c}^k,0))$ and by (2.20) the sequences $\{\hat{f}^k\}$ and $\{\hat{c}^k\}$ are bounded. Extracting from K' a further subsequence K if needed, we let

$$\bar{f}=\lim_{k\in K}\hat{f}^k, \quad \bar{c}=\lim_{k\in K}\hat{c}^k, \quad \bar{\rho}=\lim_{k\in K}\rho_k, \quad \bar{\sigma}=\lim_{k\in K}\sigma_k,$$

noting that $\bar{\rho}$ is not necessarily finite. Passing to the limit as $K \ni k \to \infty$ in (4.4) yields (4.3).

Consider first the case $\bar{c}>0$. Since X is bounded and both f and c are real-valued, the constant

$$R = \frac{1}{\bar{c}}\Big(\max_{y\in X}(f(y)-c(y))-\bar{f}\Big)+\bar{\sigma}$$

is well defined and any $\bar{\rho}>R$ satisfies $(\bar{\rho}-\bar{\sigma})\bar{c} > f(y)-c(y)-\bar{f}$ for all $y \in X$. Since the inequality is equivalent to having $f(y)-\bar{f}-\bar{\rho}\max(\bar{c},0) < c(y)-\bar{\sigma}\max(\bar{c},0)$, the stated results follows from (4.3).

When $\bar{c} \le 0$, (4.3) becomes $0 \le \max\Big(f(y)-\bar{f}, c(y)\Big)+\varepsilon$ and evaluating at any $y \in X_\varepsilon$ gives the final result. □

Theorem 4.1 states that, as long as ρ_k is managed as a penalty parameter (for instance $\rho_{k+1}=2\rho_k$ if $c_{\hat{x}^{k+1}}>0$, and $\rho_{k+1}=\rho_k$ otherwise), Algorithm 2.1 will eventually detect problem (1.2) as infeasible up to the accuracy ε, or it will find an approximate minimizer in the sense stated by the second item in the theorem. For the latter to happen, the accuracy should be small enough to ensure nonemptiness of the set X_ε (noting that in this set the factor -2 could be replaced by any value strictly smaller than -1).

We now state a refinement of this result, for the case when inaccurate linearizations eventually stay *below* the functions f and c. We refer to this situation as having "lower" oracles.

COROLLARY 4.1 (Convergence for lower oracles). *In the setting of Theorem 4.1, suppose $\varepsilon=0$ in* (4.2). *If problem* (1.2) *has a Slater point and $\bar{\rho}$ is sufficiently large, then $\bar{c} \le 0$ and $\bar{f} \le f(y)$ for all y feasible in* (1.2).

Proof. The case $\bar{c} \le 0$ is Theorem 4.1(ii) with $\varepsilon=0$. As for the case $\bar{c}>0$, it is excluded by observing that for $\bar{\rho}$ sufficiently large (4.3) becomes

$$\bar{c}(1-\bar{\sigma}) \le \max\Big(f(y)-\bar{f}-\bar{\rho}\bar{c}, c(y)-\bar{\sigma}\bar{c}\Big) = c(y)-\bar{\sigma}\bar{c},$$

so $0 < \bar{c} \le c(y)$ for all $y \in X$, an inequality that cannot hold when y is the Slater point. □

A further refinement is possible for lower oracles that are *partly asymptotically exact*, see [12], [22], [5]. Specifically, these are oracles that become progressively more and more accurate, but only at serious steps. We now show that, unlike the previous cases, in this situation the penalty ρ_k does not need to increase to infinity when the center is deemed infeasible (that is, when $c_{\hat{x}^k} > 0$).

COROLLARY 4.2 (Convergence for partly asymptotically exact lower oracles). *In the setting of Theorem 4.1, suppose* $\varepsilon = 0$ *in* (4.2) *and calculations are eventually exact for serious steps:* $f(\bar{x}) = \bar{f}$ *and* $c(\bar{x}) = \bar{c}$. *The following holds:*

(i) Either (1.2) *is feasible and* $c(\bar{x}) \le 0$ *with* $\bar{x}$ *solving* (1.2).
(ii) Or $c(\bar{x}) > 0$ *and* (1.2) *is unfeasible with* $\bar{x}$ *minimizing infeasibility.*

As a result, when (1.2) *has a Slater point, only item (i) is possible.*

Proof. When $\bar{c} \le 0$, the first result is Theorem 4.1(ii), recalling that $\bar{f} = f(\bar{x})$ and $\varepsilon = 0$.

Similarly for $\bar{c} > 0$ and $\bar{\rho} = +\infty$, applying the first item in Theorem 4.1 with $\bar{c} = c(\bar{x})$ and $\varepsilon = 0$. When $\bar{c} > 0$ and $\bar{\rho}$ is finite (not necessarily larger than R in the Theorem), adding $\bar{\sigma}\bar{c}$ to both sides of (4.3) and using that $\varepsilon = 0$ together with the assumption that $c(\bar{x}) = \bar{c}$ and $f(\bar{x}) = \bar{f}$ gives that

$$c(\bar{x}) \le H(y) := \max\Big(f(y) - f(\bar{x}) - (\bar{\rho} - \bar{\sigma})c(\bar{x}), c(y) \Big) \text{ for all } y \in X. \tag{4.5}$$

In the right handside above we use the notation $H(\cdot)$ for the (convex) exact improvement function (2.1), written with target $\tau = (f(\bar{x}) + (\bar{\rho} - \bar{\sigma})c(\bar{x}), 0)$. In particular, when $y = \bar{x}$ the inequality (4.5) becomes

$$0 < c(\bar{x}) \le H(\bar{x}) = \max\Big(-(\bar{\rho} - \bar{\sigma})c(\bar{x}), c(\bar{x}) \Big).$$

Since by (2.17) the penalties satisfy $\rho_k \ge \underline{s} + \sigma_k - 1 > \sigma_k - 1$, this means that $-(\bar{\rho} - \bar{\sigma}) < 1$ and the maximum above is attained at $c(\bar{x})$ and, hence, $H(\bar{x}) = c(\bar{x})$. As a result, (4.5) states that $0 \in \partial(H + i_X)(\bar{x})$, where i_X denotes the indicator function of the set X. Being a max-function, the subgradients of H at $\bar{x}$ are of the form

$$\begin{gathered} \lambda g_{\mathtt{f}} + (1-\lambda) g_{\mathtt{c}} \text{ for } \lambda \in [0,1] \text{ with } g_{\mathtt{f}} \in \partial f(\bar{x}) \text{ and } g_{\mathtt{c}} \in \partial c(\bar{x}) \\ \text{with } \lambda \in (0,1] \iff -(\bar{\rho} - \bar{\sigma})c(\bar{x}) \ge c(\bar{x}). \end{gathered}$$

Since $c(\bar{x}) > 0$ and $-(\bar{\rho} - \bar{\sigma}) \le 1 - \underline{s} < 1$ by (2.17), the only possibility is $\lambda = 0$, i.e., $0 \in \partial c(\bar{x})$ and, therefore, $0 < c(\bar{x}) \le c(y)$ for all $y \in X$, as stated. □

For completeness, and to relate our method with previous algorithms in the literature, we finish with a result for oracles that make exact calculations. In this case, there is no noise attenuation step and the nominal decrease (2.12) is defined with $\alpha_k \in [0,1]$.

COROLLARY 4.3 (Convergence for exact oracles). *Consider an exact oracle: for any* $x^j \in X$ (2.2) *returns* $f^j = f(x^j)$, $c^j = c(x^j)$ *and subgradients* $g_{\mathtt{f}}^j \in \partial f(x^j)$ *and* $g_{\mathtt{c}}^j \in \partial c(x^j)$. *Then Algorithm 2.1 with* $\beta_k \equiv 0$ *never executes Step 2 (noise attenuation).*

Furthermore, suppose $\alpha_k \in [0,1]$, *the penalties satisfy* (2.17), *the prox-parameter is updated according to* (2.16a) *and* (2.16c), *and the model satisfies* (3.4). *Then the statements in Corollary 4.2 apply.*

Proof. By convexity, an exact oracle is of the lower type, so (4.1) holds with $\varepsilon = 0$. In particular, writing (2.3) and (2.4) with $y = \hat{x}^k$ we see that $h_{\hat{x}^k}^k \ge \mathscr{M}_k(\hat{x}^k)$. Since, by (2.5), $\mathscr{M}_k(\hat{x}^k) \ge \mathscr{M}_k(x^{k+1}) + \frac{1}{2}\mu_k \|x^{k+1} - \hat{x}^k\|^2$, the left handside in (2.11) is always nonnegative and, as long as $\beta_k \ge 0$, the algorithm will never consider noise too large (naturally so, since for exact oracles there is no noise to be detected).

The set X is bounded and the subdifferential mapping of a convex function is locally bounded, so (2.20) holds. Since in addition, (2.14) holds by taking for example $B = \frac{1}{2} = -b$, Corollary 4.2 completes the proof. □

4.2. Relation with previous work. Corollary 4.3 covers methods based on improvement functions already considered in the literature for solving constrained nonsmooth problems using exact or lower oracles. More precisely, conditions (2.17) are satisfied by at least the following three choices:

$$\rho_k = \sigma_k \equiv 0, \quad \text{or } \rho_k \equiv \bar{\rho} < +\infty \text{ and } \sigma_k \equiv 1, \quad \text{or } \bar{\rho} = +\infty \text{ and } \sigma_k \equiv 0,$$

corresponding to the improvement functions in [34], [1], and [21], respectively. The first two methods were developed for exact oracles ([1] deals with nonconvex functions). The method of centers in [21] considers a setting corresponding to a lower oracle, and is addressed by Corollary 4.2.

To decide if the iterate gives a serious step, the three methods ([1, 21, 34]) use in Step 4 a criterion based on descent of the improvement function. Namely,

$$\text{if } h^k_{x^{k+1}} = \max(f^{k+1} - \tau_1^k, c^{k+1} - \tau_2^k) \leq h^k_{\hat{x}^k} - m\delta^k \quad \text{the iteration is declared serious} \tag{4.6}$$

(noting that the test (2.15) is also mentioned in [21] as a possibility). In view of the second item in Proposition 2.2, the criterion (2.15) is stronger for null steps. Depending on the problem, one criterion or the other might be preferable. By checking the proofs of Lemma 3.1 and 3.3, it is not difficult to see that the asymptotic results in Section 3 still hold with the alternative test.

It should also be mentioned that [21] uses an alternative stopping test, suitable for unbounded feasible sets (which is not the case in (1.2)). More precisely, instead of checking if $\delta^k \leq \text{TOL}$, the method of centers uses the conditions

$$\max\left\{\|G^k + v^k\|, \mathrm{E}^k + \left\langle G^k + v^k, \hat{x}^k \right\rangle\right\} \leq \text{TOL}_{[21]} \text{ and } c_{\hat{x}^k} \leq 0. \tag{4.7}$$

For bounded feasible sets in (1.2) conditions (4.7) are equivalent with the stopping test $\delta^k \leq \text{TOL}$. Indeed, following the definition of E^k in (2.9) and δ^k in (2.12) with $\alpha_k = 0$, we see that $\mathrm{E}^k + \langle G^k + v^k, \hat{x}^k \rangle = \delta^k + \langle G^k + v^k, \hat{x}^{k+1} \rangle$. Using the fact that X is compact and in particular bounded together with $\|G^k + v^k\| \leq \text{TOL}_{[21]}$, the stopping criteria (4.7) implies that $\delta^k \leq \text{TOL}$ for certain modified tolerance.

Another stopping criterion is the one used in [10]:

$$\max\left\{\mu_k \|x^{k+1} - \hat{x}^k\|^2, \mathrm{E}^k\right\} \leq \text{TOL}_{[10]}. \tag{4.8}$$

It was already mentioned that Lemma 3.1, 3.2, 3.3, ensure that $\|G^k + v^k\| = \mu_k \|x^{k+1} - \hat{x}^k\| \to 0$. Together with (2.18) the stopping test $\delta^k \leq \text{TOL}$ implies (4.8) for an appropriate tolerance $\text{TOL}_{[10]}$. On the other hand, keeping the prox-parameters uniformly bounded from above in (2.16), together with (2.18) and (2.6) gives

$$\delta^k = \mathrm{E}^k + \frac{2 - \alpha_k}{2} \mu_k \|x^{k+1} - \hat{x}^k\|^2 \leq 2 \max\left\{\mathrm{E}^k, \mu_k \|x^{k+1} - \hat{x}^k\|^2\right\}, \tag{4.9}$$

showing that (4.8) also implies $\delta^k \leq \text{TOL}$ for an appropriately chosen tolerance TOL.

Therefore, for convex problems as (1.2), our approach includes previous work, significantly extending the applicability of algorithms in the literature:

– not only exact and lower oracles can be used, but also upper ones, satisfying (4.1);
– the criterion for serious steps can be (2.15) or based on the improvement function;
– the descent and noise parameters α_k, β_k can be chosen in any manner satisfying (2.14). This versatility has a positive impact on the numerical results.
– any choice for the penalty parameters ρ_k and σ_k in (2.3) satisfying (2.17) is possible.

Regarding the last item, the role/utility of the penalty in the second target (in τ_2^k) is not clear. At least in the convex setting, taking $\sigma_k = 0$ and forgetting about this additional parameter seems sufficient. It is argued in [1], however, that a positive σ_k can be beneficial in the numerical performance of the algorithm, so the situation may be different for nonconvex problems (noting that our convergence theory, unlike [1] strongly relies on convexity).

5. Energy Application: Hydro Reservoir Management. In cascaded reservoir management, the decision vector $x \in \mathbb{R}^n$ corresponds to decisions on turbining and/or pumping. Generally, an individual hydro valley is part of a larger hydro-thermal system which acts on a price signal, coming either from some price decomposition scheme (e.g., [33] and references therein) or from the market. The pricing of generated power is a function of the turbining decisions. In turn, the water used to generate this power is valued according to "water-values", obtained when solving mid-term (yearly time-span) optimization problem by dynamic programming techniques.

An important constraint in cascaded reservoir management is the flow constraint, which balances the volume in a specific reservoir after turbining or pumping. The volume in each reservoir has to remain between a lower and upper bound. Whenever inflows are assumed to be uncertain so is the flow constraint. When decisions are taken prior to observing uncertainty on inflows, satisfying the upper/lower bound on the volume in each reservoir can no longer be guaranteed. A probabilistic constraint of the type (1.1) can then be used to define a safe decision along all the hydro valley. If inflows are modelled by a causal time series with Gaussian innovations, the optimization problem becomes:

$$\begin{aligned} &\min_{x\geq 0} \langle f,x\rangle \\ &s.t.\ \tilde{A}x \leq \tilde{b} \\ &\quad p \leq \mathbb{P}[a+Ax \leq \xi \leq b+Ax], \end{aligned} \tag{5.1}$$

where $\xi \in \mathbb{R}^m$ is a Gaussian random vector with covariance matrix Σ and zero mean (we have explicitly extracted the non-zero average in a^r, b^r). For details and a thorough discussion of this model we refer to [42]. For convenience of the reader, Appendix A describes briefly a very specific instance of problem (5.1).

By Theorem 4.2.4 in [28], the feasible set in (5.1) is convex and the function

$$c(x) := \log(p) - \log(\mathbb{P}[a+Ax \leq \xi \leq b+Ax]), \tag{5.2}$$

corresponding to (1.3), is convex. Taking in addition $f(x) := \langle f,x\rangle$ and $X := \{x \in \mathbb{R}^n : x \geq 0, \tilde{A}x \leq \tilde{b}\}$ (a bounded set because x represents turbining/pumping decisions that are physically bounded), problem (5.1) fits the general form in (1.2). The f-oracle is clearly exact and, as explained below, the c-oracle satisfies (4.1) with $\varepsilon > 0$.

5.1. Discussion of the core of the c-oracle. Consider the mapping $\varphi : \mathbb{R}^n \to [0,1]$ defined by

$$\varphi(x) = \mathbb{P}[a+Ax \leq \xi \leq b+Ax], \tag{5.3}$$

where $\xi \in \mathbb{R}^m$ is a Gaussian random variable with positive definite covariance matrix. It is shown in [41, Theorem 1], see also [42, Corollary 1], that φ is differentiable and as a consequence so is $\nabla c(x) = -\frac{1}{\varphi(x)}\nabla\varphi(x)$.

The gradient $\nabla\varphi(x)$ is computed recursively by means of the transformations explained below. For $\hat{a} := Ax+a$, $\hat{b} := Ax+b$ the mapping $F_\xi : \mathbb{R}^m \times \mathbb{R}^m \to [0,1]$ defined by $F_\xi(\hat{a},\hat{b}) := \mathbb{P}[\hat{a} \leq \xi \leq \hat{b}]$ gives the relation:

$$\nabla\varphi(x) = \nabla_{\hat{a}} F_\xi(\hat{a},\hat{b})^\mathsf{T} A + \nabla_{\hat{b}} F_\xi(\hat{a},\hat{b})^\mathsf{T} A. \tag{5.4}$$

If the covariance matrix of ξ is only positive semidefinite, [16] shows that (5.4) holds for almost all x (for the remaining x, the righthand side of (5.4) might still provide a sub-gradient, but this assertion is not shown).

By (5.4), computing the gradient $\nabla\varphi(x)$ amounts to computing the two right handside gradients therein, and these two terms can be written using a recursive-like relation involving alike probability terms. Indeed, the i-th component of $\nabla_{\hat{a}} F_\xi(\hat{a},\hat{b})$ ($-\nabla_{\hat{b}} F_\xi(\hat{a},\hat{b})$ respectively) is equal to

$$-\psi(z)\mathbb{P}[\tilde{a}+\tilde{A}x \leq \tilde{\xi} \leq \tilde{b}+\tilde{A}x]. \tag{5.5}$$

In this expression, ψ is the density of a standard Gaussian random variable in 1 dimension; z a specific point depending on $\hat{a}$ ($\hat{b}$), and $\tilde{a},\tilde{b},\tilde{A},\tilde{\xi}$ of a,b,A are appropriate modifications of the respective objects. These modifications depend on whether the derivative is taken with respect to $\hat{a}$ or $\hat{b}$.

As a result, to compute the gradient, the c-oracle needs to evaluate in a fast and reliable manner mappings of the form $F_\zeta(\hat{a},\hat{b})$ at $\hat{a} \leq \hat{b} \in \mathbb{R}^{m-1}$ for a multivariate Gaussian random variable $\zeta \in \mathbb{R}^{m-1}$. And to compute the approximate c-function value, a similar calculation is needed, but on $\mathbb{R}^m$. Estimates $\hat{F}_\zeta(\hat{a},\hat{b})$ can be obtained by Monte-Carlo sampling, in a manner such that the relation

$$\left|\hat{F}_\zeta(\hat{a},\hat{b}) - F_\zeta(\hat{a},\hat{b})\right| \leq \varepsilon^g, \tag{5.6}$$

holds with probability at least q. The confidence level ε^g and probability q are related by the standard deviation σ_{MC} of the Monte-Carlo sampling procedure. For instance

$$\varepsilon^g = \begin{cases} 1.96\sigma_{MC} \\ 3.5\sigma_{MC} \\ 7\sigma_{MC} \end{cases} \Rightarrow q = \begin{cases} 0.95 \\ 0.995 \\ 1-2.6\cdot 10^{-12}. \end{cases}$$

Since σ_{MC} can be reduced by increasing the sample size (or by using variance reduction techniques), for practical purposes we can assume that the c-oracle returns an inexact value satisfying (5.6) with 100% confidence (i.e., $q = 1$). Indeed, one can readily design the oracle to continue drawing Monte-Carlo samples until a pre-given confidence level ε^g has been reached with pre-given confidence level q.

To compute the estimates $\hat{F}_\zeta(\hat{a},\hat{b})$ in (5.6) we use the code developed by A. Genz ([13, 14]) that sets $\varepsilon^g \geq 3.5\sigma_{MC}$, so (5.6) holds with $q = 0.9995 \approx 1$. Since the integral approximation estimate can be larger or smaller than the exact value, the c-linearizations may lie below or above the exact function c, i.e., in (4.2) we have $\varepsilon = C\varepsilon^g > 0$ for some constant $C > 0$ depending on the data. Therefore, asymptotically exact calculations are possible up to the machine precision, provided σ_{MC} is small enough (this can considerably increase the time spent in the oracle). We will now discuss these issues in more details.

5.2. An inexact upper oracle for the constraint. The oracle having as a core the procedure described in Section 5.1 is of the upper type. We now will derive a more explicit expression for $\varepsilon > 0$ in (4.2).

Consider a constant $\Phi > 0$, for which $\varphi(x^k) > \Phi > 0$ for all bundle iterations k. For example, the initial point $\hat{x}^0$ can be a convex combination with the Slater point $x^s \in \mathbb{R}^n$ ensuring that $\varphi(\hat{x}^0) > \Phi$ for any given $p > \Phi > 0$. Then (2.15) ensures that $\varphi(\hat{x}^k) > \Phi$ at serious iterates. Since ε in (4.1) depends on the limiting behaviour, clearly one can choose μ_k in (2.5) according to rule (2.16c) in such a way that $\varphi(x^k) > \Phi$ also at null steps.

By uniform continuity of the logarithm on $[\Phi, 1]$, the compactness of X implies that for any $\varepsilon' > 0$ a precision ε^g can be chosen such that $\left|c(x^k) - c_{x^k}\right| \leq \varepsilon'$ for any iteration k. By combining (5.4), (5.5) and (5.6),

$$\|\nabla\varphi(x) - \nabla\hat{\varphi}(x)\|_\infty \leq \frac{2}{\sqrt{2\pi}}\|A\|_\infty\varepsilon^g, \tag{5.7}$$

for all $x \in X$, since the standard Gaussian density ψ satisfies $|\psi(z)| \leq \frac{1}{\sqrt{2\pi}}$ for all $z \in \mathbb{R}$. Here $\nabla\hat{\varphi}(x)$ refers to the approximate gradient discussed in Section 5.1. By the Cauchy-Schwarz inequality and compactness of X,

$$\left|\left\langle \nabla c(x^k), y - x^k\right\rangle - \left\langle \nabla\hat{c}(x^k), y - x^k\right\rangle\right| \leq M\varepsilon^g$$

for an appropriate constant $M > 0$, any $y \in X$ and any arbitrary iterate x^k (again $\nabla\hat{c}(x^k)$ refers to the approximate c-gradient). So (4.1) holds with $\varepsilon^k \geq \varepsilon' + M\varepsilon^g$, where ε' can also be made arbitrarily small whenever ε^g is taken small. By (4.2), the accuracy ε can also be made arbitrarily small and satisfies $\varepsilon \geq \varepsilon' + M\varepsilon^g$.

5.3. Convergence results for the application. In (5.1) the objective function f of problem (1.2) is linear and the oracle is exact. Inexactness arises only from the probabilistic constraint c which, as explained in Sections 5.1 and 5.2, has an upper oracle. Section 5.2 also shows that the conditions of Theorem 4.1 are satisfied.

We now argue that if a Slater point x^s for problem (1.2) exists (a reasonable assumption in our application), then a proper choice of ε^g in (5.6) ensures convergence of Algorithm 2.1 to a feasible approximate solution $\bar{x}$ of problem (1.2).

Given x^s, we choose ε^g, the user-defined precision in (5.6), in such a way that $c(x^s) \leq -2\varepsilon$. This ensures that the set X_ε in Theorem 4.1 is not empty. In the setting of the theorem, let $\bar{x}$ be the limiting point generated by Algorithm 2.1. Assume moreover that ρ^k is sent to infinity when no feasible point is produced by the Algorithm. If $\bar{x}$ is infeasible according to the oracle information, $\bar{c} > 0$ and item (i) of Theorem 4.1 holds. According to this item there exists an $R > 0$ such that $\bar{\rho} \geq R$ implies $0 < \bar{c} \leq c(y) + \varepsilon$ for all $y \in X$. The specifically chosen updating rule of ρ^k entails $\infty = \bar{\rho} \geq R$ and hence leads to the contradiction $\bar{c} < c(x^s) + \varepsilon \leq -\varepsilon < 0$. Therefore, $\bar{x}$ is approximately feasible ($\bar{c} < 0$) and satisfies the relation $\bar{f} \leq f(y) + \varepsilon$ for all $y \in X_\varepsilon$ by item (ii) of the same theorem. Consequently, $\bar{x}$ solves approximately (1.2).

Returning to the discussion in Section 5.1, arguing that ε^g in (5.6) could, in a way, be considered as an absolute accuracy, we now make precise a statement that reintegrates the fact that (5.6) can only be obtained with high probability.

LEMMA 5.1. *Assume that the oracle providing estimates of the form (5.6) does so with independent errors on successive calls. Let ε^g be an upper bound on 3.5 times the Monte-Carlo standard deviation in (5.6) and assume that ε as defined in (4.2) is defined with respect to $2\varepsilon^g$ (according to the derivation of Section 5.2) instead of just*

ε^g. If a Slater point x^s exists for problem (1.2) with $c(x^s) \leq -2\varepsilon$ and Algorithm 2.1 converges in less than a 1000 iterations to a point $\bar{x}$, then this is an approximately optimal solution of (1.2) with probability at least $1-2.6e^{-6}$, whenever $m \leq 500$.

The probability that (5.6) holds with $2\varepsilon^g$ replacing ε^g is equal to $2\Phi(7)-1$, where Φ is the distribution function of a standard normal random variable. The errors are independent. At each iteration of Algorithm 2.1 at most $2m+1$ estimates are made. Therefore, the probability that (5.6) holds with $2\varepsilon^g$ for all $2m+1$ estimates simultaneously is $(2\Phi(7)-1)^{2m+1}$. By assumption the errors are also independent in successive iterations, so the probability that (5.6) holds simultaneously for all estimates over the course of a 1000 iterations is $q := (2\Phi(7)-1)^{1000(2m+1)}$. Hence upon defining ε with respect to $2\varepsilon^g$ (as derived in Section 5.2) we may consider $2\varepsilon^g$ as an absolute bound with probability at least q. The analysis conducted in Section 5.2 shows that the oracle behaves as an upper oracle with probability at least q. The analysis conducted above then shows convergence to an approximate solution with probability at least q.

6. Numerical experience. To define the starting point we use x^d, a solution to the linear program

$$\begin{aligned} \min_{x\geq 0, x\in\mathbb{R}^n} \ & \langle f, x\rangle \\ s.t. \quad & \tilde{A}x \leq \tilde{b} \\ & a + Ax \leq 0, b + Ax \geq 0, \end{aligned} \tag{6.1}$$

the "deterministic" counterpart of (5.1) wherein the random vector ξ is replaced by its expectation. In general x^d will not be feasible for (5.1) (unless it solves the problem).

Since improvement functions are scale-dependent, [34], it is useful to scale the constraint. Accordingly, we consider the constraint $Kp \leq K\mathbb{P}[a+Ax \leq \xi \leq b+Ax]$ for some value of $K > 0$. A natural choice for K would be $\left|\left\langle f, x^d\right\rangle\right|$. Finally rules (2.16a), (2.16c) and (2.16b) are dealt with in the following way. In serious steps we take $\mu_{k+1} = \mu_k$ without making any changes. When noise is detected, we take $\mu_{k+1} = \kappa\mu_k$ for a parameter $\kappa \in (0,1)$. Finally when null steps are made we choose $\mu_{k+1} = \min\{\mu_s\mu_k, \mu_{max}\}$ for a parameter $\mu_s > 1$.

6.1. Algorithms in the Benchmark. We first compare the performance of several methods, including variants of Algorithm 2.1 using different choices for the parameters. More precisely, we consider:

– A configuration of Algorithm 2.1 with a strong noise test:

$$Alg.2.1\texttt{SEV}: \quad \sigma_k \equiv 0, \quad \rho_k \equiv 0, \quad \alpha_k \equiv 1, \quad \beta_k \equiv -1+\varepsilon^m, \quad \text{StopTest (4.8)}, \text{ DescTest (4.6)},$$

where ε^m is the machine precision.

– A configuration of Algorithm 2.1 with null parameters:

$$Alg.2.1\texttt{NUL}: \quad \sigma_k \equiv 0, \quad \rho_k \equiv 0, \quad \alpha_k \equiv 0, \quad \beta_k \equiv 0, \quad \text{StopTest (4.7)}, \text{ DescTest (4.6)},$$

– The method of centers from [21], bearing some similarities with Algorithm 2.1 if parameters are properly set:

$$Alg.[21]: \quad \sigma_k \equiv 0, \rho_k \to \infty \text{ if } \{\hat{x}^{k+1}\} \text{ infeasible}, \alpha_k \equiv 0, \beta_k = \text{ad-hoc}, \text{StopTest (4.7)}, \text{ DescTest (4.6)}.$$

– The conic bundle method from [23]:

$$Alg.[23]: \quad \text{No } \sigma_k, \rho_k, \quad \alpha_k \equiv 0, \quad \text{ad-hoc QP in Step 1, } \beta_k, \text{ Stop and Descent Test}.$$

– The supporting hyperplanes method from [29] (see also [43, 30, 38]):

$$Alg.[29]: \quad \text{No } \sigma_k, \rho_k, \alpha_k, \beta_k, \qquad \text{Linear Program in Step 1, ad-hoc Stop and Descent Test}.$$

In both configurations of Algorithm 2.1 above taking $\underline{s} = 1$ ensures satisfaction of (2.17) for any choice of ρ_{k+1}; as for (2.14), it suffices to take any positive B and $b = \beta_k$ (constant in k). Since the conditions in Section 4 are satisfied, eventual convergence to an approximate solution is assured with these methods.

We now describe the specific ad-hoc updates of the last three methods. To help the comparison, the algorithms steps are numbered as in Algorithm 2.1.

Alg.[21] Specifics. The penalty ρ_{k+1} is halved at serious steps and doubled if $c_{\hat{x}^{k+1}} > 0$. Once a feasible stability center is found, the numerical value of this penalty becomes irrelevant. For an additional Armijo-like parameter $m' \in (0,1]$, the method of centers modifies the noise management Step 2 in Alg. 2.1 as follows.

Step 2'a (Infeasible serious point) If $\hat{c}^k > 0$ and $\delta^k < m'\hat{c}^k$, noise is too large. Decrease the prox-parameter as in (2.16b), take $\rho_{k+1} = 2\rho_k$, maintain the bundle, and loop to Step 1 (solve a new QP subproblem).

Step 2'b (Noise attenuation). In the other cases, that is, if $\hat{c}^k \leq 0$ or $\delta^k \geq m'\hat{c}^k$, check if condition (2.13) holds. If this condition holds, noise is too large. Decrease the prox-parameter as in (2.16b), take $\rho_{k+1} > \rho_k$ if $\hat{c}^k > 0$, maintain the bundle, and loop to Step 1 (solve a new QP subproblem).

The difference with our noise management step is in the first item: when the current serious point is infeasible, instead of (2.13) the condition $\delta^k < m'\hat{c}^k$ is checked. Such condition amounts to verifying if (2.13) holds with $\beta_k = 1 - \frac{2m'\hat{c}^k}{\mu_k\|x^{k+1}-\hat{x}^k\|^2}$. Suppose such value of β_k also satisfies (2.14) (for example suppose m' is sufficiently small). Then this alternative can also be considered a special case of Algorithm 2.1, with a specific updating of β_k for infeasible serious points.

Alg.[23] Specifics. This method is applicable when, like in our application, the objective function in (1.2) is linear or quadratic: the QP subproblem approximates (1.2) with a cutting-plane model for the constraint. Furthermore, the method needs the knowledge of a Slater point x^s for (1.2), which is also its starting point: $\hat{x}^0 = x^s$. The algorithm performs the following steps.

Step 1' (Alternative subproblem, δ^k, E^k) Let (x^{k+1}, η^{k+1}) be a primal-dual solution to the QP

$$\begin{aligned} \min_{y\in X\subseteq\mathbb{R}^n} & \langle f, y\rangle + \frac{1}{2}\mu_k\|y-\hat{x}^k\|^2 \\ s.t. \quad & \check{c}_k(y) \leq 0\,. \end{aligned}$$

This amounts to taking in (2.6), instead of subgradient $G^k \in \partial\mathscr{M}^k(x^{k+1})$, a vector $G^k \in f + \eta^{k+1}\partial\check{c}_k(x^{k+1})$ given by the QP optimality conditions. Taking $\delta^k := \langle f, \hat{x}^k - x^{k+1}\rangle$ and $\mathrm{E}^k = \delta^k - \mu_k\|x^{k+1}-\hat{x}^k\|^2$ ensures satisfaction of the left handside identity in (2.18) with $\alpha_k = 0$.

Step 3'a (Stopping test) Stop if $\|G^k + v^k\| \leq tol$ and $\mathrm{E}^k + \langle G^k + v^k, \hat{x}^k\rangle \leq 0$,

Step 3'b (Interpolation Step) If $c_{x^{k+1}} \leq 0$, set $\gamma^k := 1$, otherwise $\gamma^k := \frac{-c_{x^s}}{c_{x^{k+1}} - c_{x^s}}$. Define $\check{x}^k := \gamma^k x^{k+1} + (1-\gamma^k)x^s$.

Step 4' (Serious step Test) If $\langle f, \check{x}^k\rangle \langle f, \hat{x}^k\rangle - m\delta^k$ then declare a serious step, taking as next center $\hat{x}^{k+1} = \check{x}^k$. Otherwise, declare a null step.

When the *c*-oracle is exact or lower*, each stability center will be feasible, by convexity. When the *c*-oracle is of the upper type, like in our case, by the discussion in Section 5.2 it is possible to take a slightly larger γ^k so that $\varphi(\check{x}^k) \geq p + \varepsilon^g$. This is a potentially costly fix that may require several evaluations (our implementation uses the original definition for γ^k.) The analysis in [23] does not cover upper oracles, so convergence of this method is unclear, although its numerical behaviour was reasonable for our tests.

Alg.[29] Specifics. Like Alg.[23], the cutting-planes model for the constraint is built using for the *c*-evaluation points, points obtained from interpolating with the Slater point. The difference is that now the constraint is required to be active at the interpolation point ($x_c^0 = (1-\lambda^0)x^0 + \lambda^0 x^s$ satisfies $c_{x_c^0} = 0$ for some $\gamma^0 \in [0,1]$.) For a linear objective function the algorithm performs the following steps.

Step 1' (Linear Programming subproblem) Let x^{k+1} be a solution to

$$\begin{aligned} \min_{y\in X\subseteq\mathbb{R}^n} & \langle f, y\rangle \\ s.t. \quad & \check{c}_k(y) \leq 0\,. \end{aligned}$$

Step 3'a (Interpolation and Oracle) Determine $\gamma^k \in [0,1]$ for which $x_c^{k+1} = (1-\gamma^k)x^{k+1} + \gamma^k x^s$ satisfies $c_{x_c^{k+1}} = 0$. Call the *c*-oracle and add the linearization to the cutting-plane model.

Step 3'b (Stopping Test and Loop) If $\langle f, x_c^{k+1} - x^{k+1}\rangle < \langle f, x^{k+1}\rangle_{\mathrm{TOL}}$ then stop. Otherwise, set $k = k+1$ and loop to Step 1'.

*In this case one requires the *c*-oracle to be exactly evaluated at the Slater point

For exact c-oracles, the method is a specialization of the cutting-plane algorithm in nonsmooth optimization, hence it converges. For inexact c-oracles, convergence results are not known.

6.2. Computational results. We comment here the most relevant observations from our numerical experience; the table with full results is in Appendix B. We note that the reported CPU times are to be taken as a way of comparing different algorithms, rather than as a measure of performance (e.g., our implementation does not use parallelization, etc...). Another important issue is that computing a Slater point can be time consuming, because it requires to solve the "maximize-p" problem, i.e., (5.1) with objective function replaced by the maximum probability level. In our experience, solving the max-p problem to optimality is as difficult as solving the original problem (5.1).

Results reported in Tables 6.1,B.1,6.2,6.3 were obtained on a HP xw6200 workstation with 8 Gb memory, whereas results of Table 6.4 were obtained on a HP z600 workstation.

TABLE 6.1
Comparison of Algorithms (nne^x stands for nn10^x), assuming a Slater Point available. Precision $\varepsilon^g = 1e^{-4}$ in (5.6).

method	Obj. Value	Nb. Iter.	CPU time (mins)	parameters
Alg.[29]	-103197	17	247.7	$tol = 1e^{-2}$
Alg.[29]	-104070	45	940.3	$tol = 1e^{-3}$
Alg.[29]	-104154	94	2079.17	$tol = 1e^{-4}$
Alg.2.1SEV	-104162	294	1028.43	$K = 1e^5, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.7$
Alg.2.1SEV	-104160	215	723.55	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.25$
Alg.2.1SEV	-104162	239	827.26	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 4, \kappa = 0.5$
Alg.2.1SEV	-104162	265	932.19	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.7$
Alg.2.1NUL	-104089	277	1106.43	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.25$
Alg.2.1NUL	-104076	234	887.38	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.25$
Alg.[23]	-104026	273	1030.21	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.05, \kappa = 0.05$
Alg.[23]	-104024	241	761.37	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.001, \kappa = 0.01$

6.3. Feasible Start using a Slater Point and Infeasible Start. In order to put all algorithms on equal foot, a first comparison supposes a Slater point is available to all of the methods in the benchmark. In this case, for a suitable convex combination of x^d and x^s, the starting point $\hat{x}^0$ is always feasible (Alg.[23] starts at x^s). Computing this convex multiplier requires executing step 3'a of Alg.[29]. This involves calling the c-oracle several times and might be costly. Table B.1 in the Appendix contains the output of all the runs, with various parameter settings for each algorithm (to decide on the best choice of parameters for the benchmark). We report here a shorter Table 6.1, with some illustrative results.

The best objective function value in this test was -104162, found by Alg.2.1SEV for different settings. To compare solutions, we observe if they are feasible (up to accuracy ε^g) and analyse the respective objective values. When $K = 1e^4$ Alg.2.1SEV found in 239 minutes a similar solution as Alg.[29], which took approximately the same CPU time. Taking $K = 5e^4$ the same point is reached in only 193 minutes and with $K = 1e^4$ that point is reached in 178 minutes. Table 6.1 shows the importance of the scaling parameter in the bundle method. It also shows that the supporting hyperplane method reaches good points early, but takes a very long time to converge. Basically each iteration takes approximately 15 to 20 minutes. Each bundle iteration takes less time (3 to 4 minutes). This can be explained by the fact that the supporting hyperplane method computes the exact interpolation. Now typically the cutting-plane method would yield a new iterate x^k far from the probability boundary $\mathbb{P}[a^r + A^r x \leq \xi \leq b^r + A^r x] = p$ and it would seem that oracle takes more time on such points than on strictly feasible points. It is also interesting to note that each stability center in the bundle methods is feasible. Thereby empirically providing support for the initial discussion in Section 5.2. Since in this setting Alg.[21] and Alg.2.1NUL would have similar results, we did not include them in this comparison.

The results in Table 6.2 were obtained with an infeasible starting point, $\hat{x}^0 = x^d$ from (6.1), using for each algorithm the best parameter settings in Table B.1.

For these runs, both Alg.2.1NUL and Alg.[21] modify the serious step test in Step 4, as follows. When the current stability center is infeasible, in addition to the usual test for acceptance, any improvement in feasibility by at least the c-oracle precision ε^g declares the current iterate as a new stability center. The modification was done

TABLE 6.2
Comparison of Algorithms (nne^x stands for nn10^x), infeasible starting point. Precision $\varepsilon^g = 1e^{-4}$ in (5.6).

method	Obj. Value	Nb. Iter.	CPU time (mins)	parameters
Alg.2.1SEV	-104162	133	379.45	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.25$
Alg.2.1NUL	-104154	73	167.25	$K = 1e^4, \mu_0 = 1e^{-5}, \mu_s = 2.0, \kappa = 0.1$
Alg.[21]	-104153	55	114.52	$K = 1e^4, \mu_0 = 1e^{-5}, \mu_s = 2.0, \kappa = 0.1$

to prevent the methods from stalling in the early stages of the algorithmic process and can only be active a finite number of iterations. Alg.2.1SEV does not stall and needs no modification. Once more, this variant is the best one in the benchmark, since it finds a feasible point with objective function value -104153 in only 56 iterations (similar to the other two algorithms in the comparison.) The best objective value of -104162 was obtained at the stake of many iterations. This table shows that important speed ups can be gained when starting with infeasible points. Alg.[29] requiring a Slater point and reaching a feasible solution with objective function value -104154 in 2079.17 minutes (see Table 6.1), is definitely out-performed by the bundle method settings of Table 6.2.

The analysis above shows a strong dependence of the CPU time on the initial point, for all methods.

6.4. Different variants of Alg.2.1SEV. In order to determine the overall impact on the convergence, we varied the precision ε^g of the oracle. This test also allows us to check the potential of varying this precision along the iterations. We took the best method, Alg.2.1SEV, with the same parameter setting, for different oracle precisions. Table 6.3 reports the corresponding results, that all gave feasible solutions, except for the unreported case $\varepsilon^g = 1e^{-2}$.

TABLE 6.3
Effect of setting ε^g in the oracle. (nne^x stands for nn10^x), Alg.2.1SEV with $K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0\kappa = 0.5$.

Starting Point	ε^g in (5.6)	objective Value	Nb. Iterations	CPU time (mins)
Feasible	$1e^{-3}$	-104163	216	87.13
Feasible	$1e^{-4}$	-104162	222	722.59
Feasible	$1e^{-5}$	-104161	261	54390.5
Infeasible	$1e^{-3}$	-104163	108	41.36
Infeasible	$1e^{-4}$	-104162	128	338.37

From this table, we see that for higher oracle accuracy, the CPU time reaches inacceptable large values.

Continuing with our analysis of the best variant, Alg.2.1SEV with fixed settings, in Table 6.4 we consider different values of β_k in (2.11), ranging between the severe noise test (close to -1) to the permissive one (close to 1), and likewise for α_k in (2.12), which varies from a severe serious step test (close to 0) to a permissive one (close to 2).

TABLE 6.4
Effect of noise test (nne^x stands for for nn10^x), Alg.2.1SEV with $K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0\kappa = 0.5$, TOL $= 0.5$, *and $\varepsilon^g = 5e^{-4}$ in (5.6)*

Starting Point	α	β	objective Value	Nb. Iterations	CPU time (mins)
Infeasible	0	$-1+\varepsilon^m$	-104160	88	21.33
Infeasible	0	0	-104157	87	21.0
Infeasible	0	$1-\varepsilon^m$	-104159	111	30.30
Infeasible	1	$-1+\varepsilon^m$	-104158	60	12.8
Infeasible	1	$-\varepsilon^m$	-104158	70	15.57
Infeasible	$2-2\varepsilon^m$	$-1+\varepsilon^m$	-104077	24	5.35

Higher α values lead to more serious steps, until the extreme of declaring all iterates serious ($\alpha = 2-2\varepsilon^m$). In our tests, a severe noise test has resulted in a more stable management of the prox-parameter. Indeed, we have observed that if noise gets detected early on, the value of μ_k remains unchanged a significant number of iterations. By contrast, the permissive choice for β_k leads to some chaotic changes throughout the iterative process.

Finally, we also examined the impact of penalty parameter ρ_k, comparing taking $\rho_k = 0$ (as in the previous runs in this subsection) with an update as for Alg.[21]. For the sake of brevity, we omit the table with results. Instead, we observe that with a nonnull ρ_k a feasible stability center is found in half the number of iterations. Unfortunately this quest for feasibility strongly deteriorates the objective function value, and eventually the algorithm takes

longer to converge to very similar solutions. The two alternative stopping criteria and the alternative serious step condition have been tested, but no significant difference was observed.

We conclude that, at least for our runs, the best variant is Alg.2.1SEV with the settings in Table 6.4.

6.5. Solution Quality. To assess the obtained solutions, we simulate the reservoirs evolution and check if the stipulated probability level is satisfied numerically. Figure 6.1 shows the reservoir levels for 100 simulated inflow scenarios, using the expected-value strategy obtained from solving problem (6.1) and Alg.2.1SEV. The figure clearly shows the importance of integrating uncertainty in order to obtain robust turbining strategies.

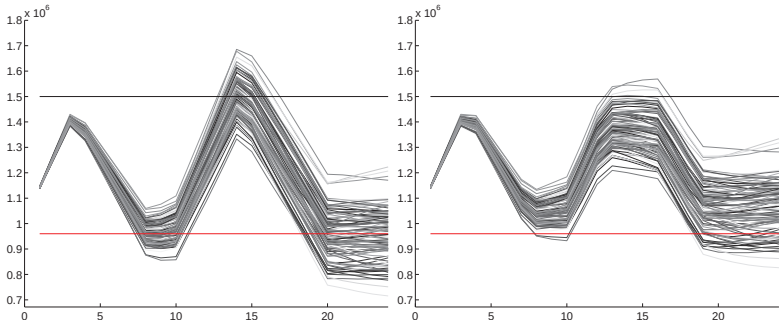


FIG. 6.1. *Evolution of reservoir "Saut Mortier", for the expected-value strategy (left) and for Alg.2.1*SEV *strategy (right).*

Except for the deterministic solution, which violates constraints for almost all scenarios, the various methods in our benchmark provide very similar solutions. When slight differences arise, they account for increased robustness (with respect to the deterministic solution) and for increased optimality (with respect to various solution methods).

Conclusions. For convex nonsmooth constrained problems we have presented a new bundle algorithm capable of dealing with inexact upper oracles. These oracles provide linearizations that may not lie below the true convex function. Such a setting naturally arises when dealing with certain classes of Joint Chance Constrained Programming. Indeed, in many cases of interest, an appropriate transformation of the Joint Chance Constraint is a convex function for which obtaining a gradient or function value involves (quasi-)Monte Carlo sampling and multivariate numerical integration. A similar situation arises in stochastic programs with second order stochastic dominance, Conditional Value-at-Risk, or robust constraints. As long as the resulting optimization problems remain convex with a finite number of constraints, the bundle method presented in this paper can be applied, taking advantage of its ability of handling inexact oracles to accelerate the calculations.

A possible extension is to develop a *randomized* bundle method, that instead of deterministic information would work with probabilistic oracles (computing inexact values for the chance constraint and its gradient, as in (5.6), but using an algorithm piloting the probability q). By mimicking the partly asymptotically exact setting, the resulting method should find exact solutions with probability one. This is an interesting subject for future research.

The good qualities of the proposed methodology were illustrated on a real-life numerical example coming from unit-commitment: the hydro-reservoir problem. When integrating uncertainty on inflows this problem can be formulated as a Joint Chance Constrained Program. The bottleneck for an efficient numerical resolution lies in the cost of the oracle calculations for the Joint Chance Constraint. For inflows modelled as a causal time series with Gaussian innovations, and using an efficient code like [13, 14], the oracle evaluation typically takes up to 98% of the total CPU running time. The methods considered in this work do not need the (potentially time consuming) calculation of a Slater point. With respect to the well-known supporting hyperplane method, the bundle methods presented in this paper were shown to accelerate the CPU time by a factor of 20 on the typical instance considered in this paper. This shows a great potential for this methodology. With a varying precision for the oracle ("asymptotically exact" variant, not tested in this work), an additional factor of 3 seems to be reachable.

Acknowledgments. We thank the reviewers and the associate editor for providing insightful comments and remarks that helped us improving this paper. Research of the second author was partially supported by Grants CNPq 303840/2011-0, AFOSR FA9550-08-1-0370, NSF DMS 0707205, as well as by PRONEX-Optimization and FAPERJ.

REFERENCES

[1] P. APKARIAN, D. NOLL, AND A. RONDEPIERRE, *Mixed H_2/H_∞ control via nonsmooth optimization*, SIAM J. Optim. and Control, 47 (2008), pp. 1516–1546.

[2] J.F. BONNANS, J.C. GILBERT, C. LEMARÉCHAL, AND C. SAGASTIZÁBAL, *Numerical Optimization: Theoretical and Practical Aspects*, Springer-Verlag, 2nd ed., 2006.

[3] A. CHARNES AND W. COOPER, *Chance-constrained programming*, Management Science, 6 (1959-1960), pp. 73–79.

[4] R. CORREA AND C. LEMARÉCHAL, *Convergence of some algorithms for convex minimization*, Mathematical Programming, 62 (1993), pp. 261–275.

[5] W. DE OLIVEIRA AND C. SAGASTIZÁBAL, *Level bundle methods for oracles with on demand accuracy*, To appear in Optimization Methods and Software, (2014), pp. 1–31.

[6] W. DE OLIVEIRA, C. SAGASTIZÁBAL, AND C. LEMARÉCHAL, *Bundle methods in depth: a unified analysis for inexact oracles*, Available at http://www.optimization-online.org/DB_HTML/2013/02/3792.html, (2013).

[7] W. DE OLIVEIRA, C.A. SAGASTIZÁBAL, AND S. SCHEIMBERG, *Inexact bundle methods for two-stage stochastic programming*, SIAM Journal on Optimization, 21 (2011), pp. 517–544.

[8] I. DURANYILDIZ, B. ÖNÖZ, AND M. BAYAZIT, *A chance-constrained LP model for short term reservoir operation optimization*, Turkish Journal of Engineering, 23 (1999), pp. 181–186.

[9] N.C.P. EDIRISINGHE, E.I. PATTERSON, AND N. SAADOULI, *Capacity planning model for a multipurpose water reservoir with target-priority operation*, Annals of Operations Research, 100 (2000), pp. 273–303.

[10] G. EMIEL AND C. SAGASTIZÁBAL, *Incremental like bundle methods with applications to energy planning*, Computational Optimization and Applications, 46 (2009), pp. 305–332.

[11] C.I. FÁBIÁN, *Bundle-type methods for inexact data*, in Proceedings of the XXIV Hungarian Operations Researc Conference (Veszprém, 1999), vol. 8 (special issue, T. Csendes and T. Rapcsk, eds.), 2000, pp. 35–55.

[12] M. GAUDIOSO, G. GIALLOMBARDO, AND G. MIGLIONICO, *An incremental method for solving convex finite min-max problems*, Math. of Oper. Res., 31 (2006).

[13] A. GENZ, *Numerical computation of multivariate normal probabilities*, J. Comp. Graph Stat., 1 (1992), pp. 141–149.

[14] A. GENZ AND F. BRETZ, *Computation of multivariate normal and t probabilities.*, no. 195 in Lecture Notes in Statistics, Springer, Dordrecht, 2009.

[15] A. GOUDA AND T. SZÁNTAI, *On numerical calculation of probabilities according to dirichlet distribution*, Annals of Operations Research, 177 (2010), pp. 185–200.

[16] R. HENRION AND A. MÖLLER, *A gradient formula for linear chance constraints under Gaussian distribution*, Mathematics of Operations Research, 37 (2012), pp. 475–488.

[17] R. HENRION AND W. RÖMISCH, *Lipschitz and differentiability properties of quasi-concave and singular normal distribution functions*, Annals of Operations Research, 177 (2010), pp. 115–125.

[18] M. HINTERMÜLLER, *A proximal bundle method based on approximate subgradients*, Computational Optimization and Applications, 20 (2001), pp. 245–266. 10.1023/A:1011259017643.

[19] J.B. HIRIART-URRUTY AND C. LEMARÉCHAL, *Convex Analysis and Minimization Algorithms II*, no. 306 in Grundlehren der mathematischen Wissenschaften, Springer-Verlag, 2nd ed., 1996.

[20] K.C. KIWIEL, *A proximal bundle method with approximate subgradient linearizations*, SIAM Journal on Optimization, 16 (2006), pp. 1007–1023.

[21] ———, *A method of centers with approximate subgradient linearizations for nonsmooth convex optimization*, SIAM Journal on Optimization, 18 (2008), pp. 1467–1489.

[22] ———, *Bundle methods for convex minimization with partially inexact oracles*, To appear in Comp. Opt. Appl, (2012).

[23] K.C. KIWIEL AND C. LEMARÉCHAL, *An inexact bundle variant suited to column generation*, Math. Program., 118 (2009), pp. 177–206.

[24] H.A. LOIACIGA, *On the use of chance constraints in reservoir design and operation modeling*, Water Resources Research, 24 (1988), pp. 1969–1975.

[25] D. P. LOUCKS, J. R. STEDINGER, AND D. A. HAITH, *Water Resource Systems Planning and Analysis*, Prentice-Halls, Inc., 1981.

[26] J. MAYER, *On the Numerical solution of jointly chance constrained problems. Chapter 12 in [39]*, Springer, 1st ed., 2000.

[27] D.R. MORGAN, J.W. EHEART, AND A.J. VALOCCHI, *Aquifer remediation design under uncertainty using a new chance constraint programming technique*, Water Resources Research, 29 (1993), pp. 551–561.

[28] A. PRÉKOPA, *Stochastic Programming*, Kluwer, Dordrecht, 1995.

[29] A. PRÉKOPA, *Probabilistic programming. In [32] (Chapter 5)*, Elsevier, Amsterdam, 2003.

[30] A. PRÉKOPA AND T. SZÁNTAI, *Flood control reservoir system design using stochastic programming*, Math. Programming Study, 9 (1978), pp. 138–151.

[31] ———, *A new multivariate gamma distribution and its fitting to empirical streamflow data*, Water Resources Research, 14 (1978), pp. 19–24.

[32] A. RUSZCZYŃSKI AND A. SHAPIRO, *Stochastic Programming*, vol. 10 of Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, 2003.

[33] C. SAGASTIZÁBAL, *Divide to conquer: Decomposition methods for energy optimization*, Mathematical Programming, 134 (2012), pp. 187–222.

[34] C. SAGASTIZÁBAL AND M. SOLODOV, *An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter*, SIAM Journal on Optimization, 16 (2005), pp. 146–169.

[35] R.H. SHUMWAY AND D.S. STOFFER, *Time Series Analysis and Its Applications*, Springer, 1st ed., 2000.

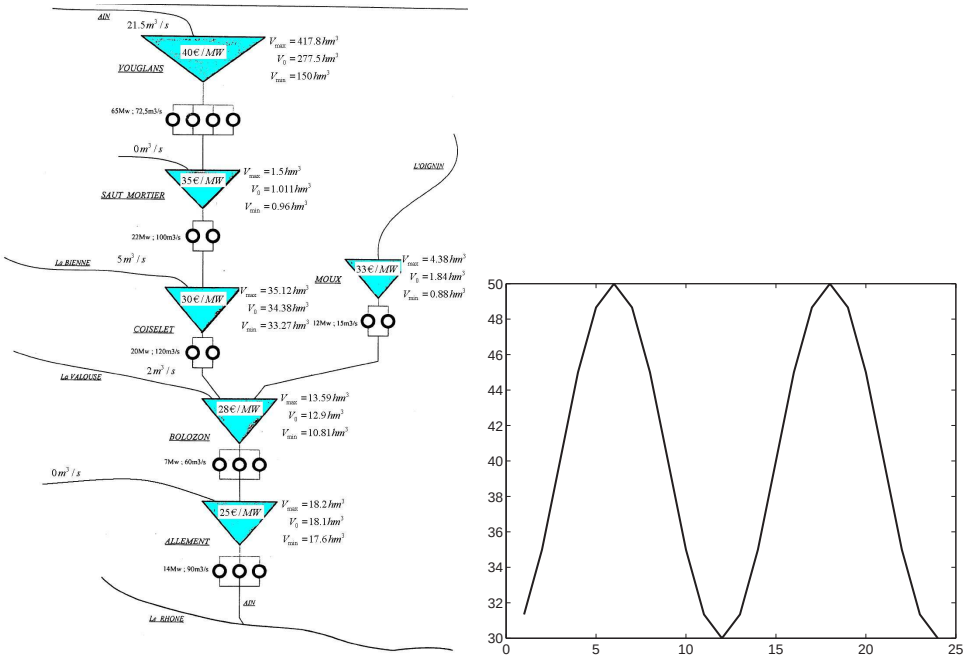
[36] M.V. SOLODOV, *On approximations with finite precision in bundle methods for nonsmooth optimization*, Journal of Optimization Theory and Applications, 119 (2003), pp. 151–165.

[37] T. SZÁNTAI, *Numerical evaluation of probabilities concerning multi-dimensional probability distributions*, PhD thesis, Hungarian

Academy of Sciences, 1985.
[38] ———, *A computer code for solution of probabilistic-constrained stochastic programming problems.*, In (Y. Ermoliev and R.J.-B. Wets eds.): *Numerical Techniques for Stochastic Optimization*, (1988), pp. 229–235.
[39] S. URYAS'EV (ED), *Probabilistic Constrained Optimization: Methodology and Applications*, Kluwer Academic Publishers, 2000.
[40] W. VAN ACKOOIJ AND R. HENRION, *Gradient formulae for nonlinear probabilistic constraints with gaussian and gaussian-like distributions*, Draft submitted, WIAS Preprint No. 1799, (2013), pp. 1–18.
[41] W. VAN ACKOOIJ, R. HENRION, A. MÖLLER, AND R. ZORGATI, *On probabilistic constraints induced by rectangular sets and multivariate normal distributions*, Mathematical Methods of Operations Research, 71 (2010), pp. 535–549.
[42] ———, *Joint chance constrained programming for hydro reservoir management*, to Appear in Optimization and Engineering, (2011).
[43] A.F. VEINOTT, *The supporting hyperplane method for unimodal programming*, Operations Research, 15 (1967), pp. 147–152.
[44] S.W. WALLACE AND S.-E. FLETEN, *Stochastic programming models in energy (chapter 10 in [32])*, in Stochastic Programming, A. Ruszczynski and A. Shapiro, eds., vol. 10 of Handbooks in Operations Research and Management Science, Elsevier, 2003, pp. 637–677.
[45] R. ZORGATI AND W. VAN ACKOOIJ, *Optimizing financial and physical assets with chance-constrained programming in the electrical industry*, Optimization and Engineering, 12 (2011), pp. 237–255.
[46] R. ZORGATI, W. VAN ACKOOIJ, AND R. APPARIGLIATO, *Supply shortage hedging: estimating the electrical power margin for optimizing financial and physical assets with chance-constrained programming*, IEEE Transactions on Power Systems, 24 (2009), pp. 533–540.

Appendix A. Brief Description of Cascaded Reservoir management.

Initial description of the system. Consider a hydro valley as in Fig. (A.1(a)).



(a) The Hydro Valley (b) The Price Signal

FIG. A.1. *Numerical Instance Data*

The system topology, that is the relation between the six power reservoirs in the valley, is expressed by the reservoir connection matrix A, having as non-zero elements $A_{12} = A_{23} = A_{35} = A_{45} = A_{56} = 1$. Likewise, the vector $\sigma_{\mathscr{T}} = (1,1,1,1,2,2,3,3,4,4,5,5,5,6,6,6)$ associates each one of the 16 turbines in the valley to a specific reservoir. Uphill and downhill connections are identified in the sets $\mathscr{A}(n) = \{m \in \{1,\dots,6\} : A_{m,n} = 1\}$ and $\mathscr{F}(n) = \{m \in \{1,\dots,6\} : A_{n,m} = 1\}$ for $n \in \{1,\dots,6\}$. For simplicity, we do not consider flow delays (also known as "water travel" times), which can easily be incorporated in the model. The time horizon $\mathfrak{T}$ is discretized into 24 time steps, each one with length $\Delta t = 2$ hours. For $i = 1,\dots,16$ the control of the i-th turbine is done by variables $x^i(t)$ for each $t \in \mathfrak{T}$, in units m^3/h, and satisfying $0 \le x^i(t) \le \overline{x}^i$, $\forall t \in \mathfrak{T}, i = 1,...,16$.

Random Inflows: modelling and data. Reservoir inflows (in m^3/h) are modelled by a stochastic process. Accordingly, $A^n(t)$ stands for the stochastic process for reservoir $n \in \{1,\ldots,6\}$. Some of the reservoirs have deterministic inflows (typically zero). Indeed, while uphill reservoirs have random inflows due to the melting of snow in the high mountains, rain can be neglected for downhill reservoirs, which can be modelled in a deterministic framework. The set $\mathscr{N}^r \subseteq \{1,\ldots,6\}$ gathers all reservoirs with random inflows. For $n \in \mathscr{N}^r$, the stochastic inflow process is modelled as a sum of a deterministic trend s_t^n and a causal process ([35]) generated by Gaussian innovations. To this end let $\zeta^n(t)$ be a Gaussian white noise process: $(\zeta^{k_1}(t),\ldots,\zeta^{k_l})$ is a Gaussian random vector of zero average and covariance matrix $\Sigma(t)$ ($\{k_1,\ldots,k_l\} = \mathscr{N}^r$). The ζ vector is assumed to have independent t-components. Since $A^n(t)$ is a causal process, for some coefficient vector ψ^n it holds that

$$A^n(t) = s_t^n + \sum_{j=0}^{t-1} \psi_j^n \zeta^n(t-j), \text{ for all } n \in \mathscr{N}^r, t \in \mathfrak{T}$$

In this instance, both reservoirs 1 and 2 have random inflows, with a standard deviation equal to 20% of the nominal values. This implies that the standard deviations are 4.24 m^3/s for reservoir 1 and 0.3 m^3/s for reservoir 2. Inflows to reservoir 2 are modelled as an autoregressive process of order three, AR(3), with coefficients $(0.9, 0.7, -0.7)$. For reservoir 1, inflows follow an AR(1) process with coefficient 0.9.

To express the hydro valley relations in the presence of uncertainty, we need to write down the covariance matrix of the stochastic process. This needs the introduction of some notation. Specifically, we let ψ be a 2×24 matrix such that $\psi_{1,j} = 0.9^j$ and $\psi_{2,j} = 0.9\psi_{2,j-1} + 0.7\psi_{2,j-2} - 0.7\psi_{2,j-3}$, $j = 1,\ldots,24$, where in the last formula any negative indexed terms are assumed 0. We also denote by Σ_ζ the block diagonal matrix with 24 copies of the submatrix $\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$, left and right multiplied by a matrix with diagonal entries 4.24 and 0.3. The expression for matrix Σ can be obtained by letting C^ψ be the 48×48 block diagonal matrix, where the first block is obtained from the first row of ψ by accumulating elements, i.e., element (i,j) of the block is $\sum_{k=0}^{i-j} \psi_k$. As a result, if R has nonnull entries $(i, mod(i-1,24)2 + 1 + \lfloor \frac{i-1}{24} \rfloor)$ equal to 1 for $i = 1,\ldots,48$, the covariance matrix is $\Sigma = (7200)^2 (R^{-1}C^\psi R)\Sigma_\zeta (R^{-1}C^\psi R)^\mathsf{T}$.

Flow constraints and Volume bounds. In the hydro valley, there are flow constraints induced by turbining water in each reservoir. Letting $\xi \in \mathbb{R}^{48}$ denote the Gaussian multivariate vector of inflows,

$$V^n(t) = V^n(t-1) + \sum_{m \in \mathscr{A}(n)} \sum_{i \in \sigma_{\mathscr{T}}^{-1}[m]} x^i(t)\Delta t - \sum_{i \in \sigma_{\mathscr{T}}^{-1}[n]} x^i(t)\Delta t s_t^n \Delta t + \xi_{24(n-1)+t}, n = 1,\ldots,6\,.$$

The above equation is deterministic, except for the reservoirs $n \in \mathscr{N}^r$. To deal with the underlying randomness while respecting each reservoir bounds, we introduce the constraints

$$\begin{aligned} \mathbb{P}\Big[V_{min}^n(t) \le V^n(t) \le V_{max}^n(t) \text{ for all } t \in \mathfrak{T}, n \in \mathscr{N}^r\Big] &\ge p \\ V_{min}^n(t) \le V^n(t) \le V_{max}^n(t) \text{ for all } t \in \mathfrak{T}, n &\in \mathscr{N} \setminus \mathscr{N}^r. \end{aligned}$$

The last joint chance constraint states that, for any level of uncertain inflows, reservoirs 1 and 2 must keep their volumes between the prescribed bounds, with probability higher than p, taken equal to 0.8 in our application.

Objective function and problem structure. In this stylized numerical example, water is priced by a constant factor W^n; a more realistic description, with volume dependent water values, can be found in [42]. Since such modelling is represented by a set of affine constraints, our setting also covers the more general case. Generation is valued by using (given) time dependent price signal $\lambda(t), t \in \mathfrak{T}$ (in €/MWh) like the one shown in Fig. (A.1(b)). Finally, letting ρ_i denote the ith turbine efficiency in MWh/m^3, the objective function in (1.2) is given by

$$\sum_{n=1}^{6} (W^n(V^n(0) - \mathbb{E}(V^n(T))) - \sum_{t \in \mathfrak{T}} \lambda(t)\Delta t \sum_{i=1}^{16} \rho_i(t) x^i(t)\,.$$

Appendix B. Table with full results, for different parameter settings and algorithms.

TABLE B.1
Comparison of Algorithms (nne^x stands for nn10^x), assuming a Slater Point available.

method	Obj. Value	Nb. Iter.	CPU time (mins)	parameters
Alg.[29]	-103197	17	247.7	$tol = 1e^{-2}$
Alg.[29]	-104070	45	940.3	$tol = 1e^{-3}$
Alg.[29]	-104154	94	2079.17	$tol = 1e^{-4}$
Alg.2.1SEV	-104162	294	1028.43	$K = 1e^5, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.7$
Alg.2.1SEV	-104160	300	1060.35	$K = 2e^5, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.7$
Alg.2.1SEV	-104162	286	991.33	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 1.05, \kappa = 0.95$
Alg.2.1SEV	-104162	243	846.20	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.7$
Alg.2.1SEV	-104160	215	723.55	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.25$
Alg.2.1SEV	-104162	255	887.07	$K = 5e^4, \mu_0 = 1e^{-5}, \mu_s = 2, \kappa = 0.1$
Alg.2.1SEV	-104162	257	907.20	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.1$
Alg.2.1SEV	-104162	239	827.26	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 4, \kappa = 0.5$
Alg.2.1SEV	-104162	265	932.19	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.7$
Alg.2.1NUL	-104109	417	1686.12	$K = 1e^5, \mu_0 = 1e^{-6}, \mu_s = 1.05, \kappa = 0.95$
Alg.2.1NUL	-104096	402	1539.16	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 1.05, \kappa = 0.95$
Alg.2.1NUL	-104102	395	1639.40	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2, \kappa = 0.7$
Alg.2.1NUL	-104089	277	1106.43	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.25$
Alg.2.1NUL	-104091	323	1257.09	$K = 5e^4, \mu_0 = 1e^{-6}, \mu_s = 4.0, \kappa = 0.5$
Alg.2.1NUL	-104078	366	1480.21	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 1.05, \kappa = 0.95$
Alg.2.1NUL	-104077	395	1591.13	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.7$
Alg.2.1NUL	-104078	278	1063.49	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.5$
Alg.2.1NUL	-104076	234	887.38	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.25$
Alg.2.1NUL	-104078	253	981.13	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.35$
Alg.2.1NUL	-104079	144	549.19	$K = 1e^4, \mu_0 = 1e^{-5}, \mu_s = 2.0, \kappa = 0.1$
Alg.2.1NUL	-104072	208	792.17	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 2.0, \kappa = 0.1$
Alg.2.1NUL	-104078	218	828.58	$K = 1e^4, \mu_0 = 1e^{-7}, \mu_s = 2.0, \kappa = 0.1$
Alg.2.1NUL	-104078	278	1126.51	$K = 1e^4, \mu_0 = 1e^{-6}, \mu_s = 4.0, \kappa = 0.5$
Alg.[23]	-104026	525	1933.29	$K = 1, \mu_0 = 1e^{-5}, \mu_s = 1.05, \kappa = 0.7$
Alg.[23]	-104026	481	1691.07	$K = 1, \mu_0 = 1e^{-6}, \mu_s = 1.05, \kappa = 0.7$
Alg.[23]	-104027	447	1749.57	$K = 1, \mu_0 = 1e^{-7}, \mu_s = 1.05, \kappa = 0.7$
Alg.[23]	-104027	454	1852.24	$K = 1, \mu_0 = 1e^{-8}, \mu_s = 1.05, \kappa = 0.7$
Alg.[23]	-104025	339	1311.24	$K = 1, \mu_0 = 1e^{-8}, \mu_s = 1.05, \kappa = 0.1$
Alg.[23]	-104024	297	1170.13	$K = 1, \mu_0 = 5e^{-9}, \mu_s = 1.05, \kappa = 0.1$
Alg.[23]	-104027	418	1602.25	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.05, \kappa = 0.7$
Alg.[23]	-104022	293	1089.25	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.05, \kappa = 0.1$
Alg.[23]	-104026	273	1030.21	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.05, \kappa = 0.05$
Alg.[23]	-104024	254	893.40	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.01, \kappa = 0.05$
Alg.[23]	-104024	254	798.32	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.01, \kappa = 0.01$
Alg.[23]	-104024	241	761.37	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 1.001, \kappa = 0.01$
Alg.[23]	-104026	303	1150.23	$K = 1, \mu_0 = 1e^{-9}, \mu_s = 4.0, \kappa = 0.1$
Alg.[23]	-104025	391	1456.13	$K = 1, \mu_0 = 1e^{-10}, \mu_s = 1.05, \kappa = 0.7$
Alg.[23]	-104026	329	1193.51	$K = 1, \mu_0 = 1e^{-10}, \mu_s = 1.05, \kappa = 0.1$
Alg.[23]	-104028	595	2340.03	$K = 1, \mu_0 = 1e^{-10}, \mu_s = 2.0, \kappa = 0.7$
Alg.[23]	-104020	360	1348.14	$K = 1, \mu_0 = 1e^{-10}, \mu_s = 2.0, \kappa = 0.1$
Alg.[23]	-104026	325	1250.17	$K = 1, \mu_0 = 1e^{-10}, \mu_s = 4.0, \kappa = 0.1$

The author has requested enhancement of the downloaded file. All in-text references underlined in blue are linked to publications on ResearchGate.