manuscript No.
(will be inserted by the editor)

Proximal bundle methods in depth: a unified analysis for inexact oracles

W. de Oliveira · **C. Sagastizábal** · **C. Lemaréchal**

November 29, 2013

Abstract The last few years have seen the advent of a new generation of bundle methods, capable to handle inexact oracles, polluted by "noise". Proving convergence of a bundle method is never simple and coping with inexact oracles substantially increases the technicalities. Besides, several variants exist to deal with noise, each one needing an *ad hoc* proof to show convergence. We state a synthetic convergence theory, in which we highlight the main arguments and specify which assumption is used to establish each intermediate result. Our framework is comprehensive and generalizes in various ways a number of algorithms proposed in the literature. Based on the ingredients of our synthetic theory, we consider various bundle methods adapted to oracles for which high accuracy is possible, yet it is preferable not to make exact calculations often, because they are too time consuming.

1 Introduction and General Aim

We are interested in the problem

$$\min f(u)\,, \quad u \in \mathbb{R}^n\,, \tag{1.1}$$

where $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a (finite-valued) convex function. For each given $u \in \mathbb{R}^n$, an oracle – a noisy one – delivers inexact information, namely

$$\begin{cases} f_u = f(u) - \eta_u \quad \text{and} \\ g_u \in \mathbb{R}^n \ \text{ such that } \ f(\cdot) \geqslant f_u + \langle g_u, \cdot - u\rangle - \eta_u^g \end{cases} \tag{1.2}$$

This research was done during a postdoctoral visit of the first author at Inria.

W. de Oliveira
Instituto Nacional de Matemática Pura e Aplicada - IMPA, Brazil
E-mail: wlo@impa.br

C. Sagastizábal
On leave from INRIA, France. Visiting researcher at IMPA, Brazil
E-mail: sagastiz@impa.br

C. Lemaréchal
Inria, 655 avenue de l'Europe, Montbonnot, 38334 Saint Ismier, France
E-mail: claude.lemarechal@inria.fr

for some unknown scalars η_u and η_u^g with $\eta_u^g \geqslant 0$. Note: for a given u, several values of a function such as f will come into play; they will always be denoted by a letter, such as f_u. To avoid confusion, the slightly pedantic notation $f(\cdot)$ will be used for the function itself.

To solve (1.1) we develop a framework for bundle methods dealing with oracles of the type (1.2) to solve (1.1). The considered setting is versatile and general, in the sense that it covers and extends previous literature, such as the inexact bundle methods [13,26, 18], the incremental bundle method [7], and the partly inexact method [19]. Our theoretical development also allows us to consider new methods, the Controllable Bundle Algorithm 5.4 and the Asymptotically Exact Algorithm in Section 7.1.4. The latter is a proximal variant of the level bundle method for oracles with on-demand accuracy considered in [22],

This paper goes beyond presenting new bundle variants, though. As demonstrated by the works [16,17], convergence of a bundle method is a fairly involved subject (unless simplifying boundedness assumptions are made, as in [20,7]). Our first aim is therefore to state a synthetic convergence theory, highlighting the main arguments and specifying which assumption is used to establish each intermediate result. In a way, we follow the spirit of [5], although this latter work considered a whole class of subgradient methods and was limited to exact oracles. The analysis is developed in a way that reveals how different procedures controlling oracle noise result in algorithms solving (1.1) with different degrees of accuracy.

Before entering into further details, and to give a flavor of different situations that can be handled by our theory, we start in Section 2 with a broad set of examples fitting the oracle (1.2). Section 3 describes the essential features of bundle methods and organizes these features in two separate sets of *parameters* whose specification gives rise to specific bundle methods. The parametric setting is useful to state an abstract algorithmic pattern in Section 4, in a manner that is general enough for our purposes. Section 5 addresses the important step of *noise attenuation* that needs to be put in place when the error oracle in (1.2) cannot be controlled. This section also provides the Controllable Bundle Algorithm 5.4, illustrating a parameter particularization for the abstract algorithmic pattern. The theory in Section 6 refers often to Algorithm 5.4 to make statements more concrete and guide the reader through the various convergence results. The final Section 7 considers many algorithms that are covered by our synthetic theory, including bundle methods for constrained problems.

2 Oracle Examples

Regarding the error signs in (1.2), η_u^g is always non-negative (a negative gradient error can be replaced by a zero value with no harm). As for the functional error, the property $\eta_u \geqslant 0$ may hold, like in (2.3) below, but it is not automatic, because we consider the general case that allows function values to be over or under-estimated. In either case, the vector g_u is an approximate subgradient of $f(\cdot)$: the second line in (1.2) implies

$$f(\cdot) \geqslant f(u) + \langle g_u, \cdot - u\rangle - (\eta_u + \eta_u^g) \tag{2.1}$$

from which, evaluating at u, we deduce

$$\eta_u + \eta_u^g \geqslant 0 \quad \text{for all } \; u \in \mathbb{R}^n . \tag{2.2}$$

With an *exact* oracle, $\eta_u \equiv \eta_u^g \equiv 0$; the output is then $f_u = f(u)$ and a true subgradient g_u. We only require from the oracle output that errors are bounded from above; say for a constant η

$$\eta_u \leqslant \eta \quad \text{and} \quad \eta_u^g \leqslant \eta \quad \text{for all } \; u \in \mathbb{R}^n .$$

In view of (2.2), both η_u and η_u^g are also bounded from below, by $-\eta$. The value of η may be unknown: the boundedness assumption will be made precise when needed.

We now review some examples of oracles. In an important subclass illustrated by Examples 2.1 and 2.2 below, the oracle (1.2) ensures that $\eta_u^g \equiv 0$ and, hence, $\eta_u \geqslant 0$ by (2.2). Following [1], we shall call this subclass of *lower oracles*, because a lower linearization of $f(\cdot)$ is available:

$$f(u) - \eta_u = f_u \leqslant f(u) \quad \text{and} \quad f(\cdot) \geqslant f_u + \langle g_u, \cdot - u \rangle \,. \tag{2.3}$$

Upper oracles, by constrast, may output positive errors and, in particular, $\eta_u^g > 0$.

Example 2.1 (Minimax: Lagrangian) For given functions $h(\cdot)$ and $c(\cdot)$ and X a nonempty set, (1.1) is dual to the primal problem

$$\max_{x \in X} h(x)\,, \quad c(x) = 0 \in \mathbb{R}^n \,. \tag{2.4}$$

Specifically, for a multiplier $u \in \mathbb{R}^n$ the dual function is given by

$$f(u) := \max_{x \in X} L(x,u)\,, \quad \text{with } L(x,u) := h(x) + \langle u, c(x) \rangle \,.$$

In a more general setting, we can consider $f(u) = \sup_{x \in X} L(x,u)$ with $L(x,\cdot)$ convex for each x. Then, for given u_0 and arbitrary $x_0 \in X$, let $f_{u_0} := L(x_0,u_0)$ and take for g_{u_0} some subgradient of $L(x_0,\cdot)$ at u_0. By convexity of $L(x_0,\cdot)$ and definition of $f(\cdot)$,

$$f_{u_0} + \langle g_0, u - u_0 \rangle = L(x_0,u_0) + \langle g_0, u - u_0 \rangle \leqslant L(x_0,u) \leqslant f(u)\,.$$

When computing an approximate maximizer of $L(\cdot,u_0)$, the oracle is of lower type and (2.3) holds with $u = u_0$. This occurs for instance in Lagrangian relaxation or column generation, when the inner maximization $\sup_{x \in X} L(x,u_0)$ is not performed exactly and also in the dynamic bundle method [2], whose Lagrangian relaxes constraints "on the fly". □

Example 2.2 (Minimax: Two-Stage Stochastic Linear Programs) A similar situation arises in stochastic optimization, for problems with decision variables organized in two levels, denoted by u and y for the first and second stage, respectively. Let $\xi \in \Xi$ be the variable representing uncertainty and consider the case where all the involved functions are affine. A two-stage linear program with fixed recourse [25] has the form:

$$\begin{cases} \min_{u,y} \ \langle e,u \rangle + \mathbb{E}[\langle q(\xi), y \rangle] \\ \text{s.t.} \quad T(\xi)u + Wy = d(\xi) \quad \text{for almost every } \xi \in \Xi, \\ \qquad\ \ y \geqslant 0\,, \end{cases}$$

where we use the symbol $\mathbb{E}(\cdot)$ for the expected value. For fixed u and ξ the *recourse* function

$$Q(u;\xi) := \inf_{y \geqslant 0} \langle q(\xi), y \rangle \ \text{ s.t. } Wy = d(\xi) - T(\xi)u$$

gives in (1.1) an objective of the form

$$f(u) := \langle e,u \rangle + \mathbb{E}[Q(u;\xi)]\,.$$

We now explain how to build different oracles for this type of problems.

A Dumb Lower Oracle. For each fixed u and a given realization ξ, the evaluation of the recourse function can be done by solving the dual linear program

$$Q(u;\xi) = \sup_x \langle d(\xi) - T(\xi)u, x\rangle \;\text{ s.t. } W^\top x \leqslant q(\xi)\,.$$

In a manner similar to Example 2.1 suppose that, to speed up calculations, instead of performing the sup-operation above we just take a point $x_{u,\xi}$ satisfying $W^\top x_{u,\xi} \leqslant q(\xi)$ for the considered ξ. Then, because

$$Q(u;\xi) \geqslant \left\langle d(\xi) - T(\xi)u, x_{u,\xi}\right\rangle,$$

the resulting lower oracle fits (2.3) with

$$f_u := \langle e,u\rangle + \mathbb{E}[\left\langle d(\xi) - T(\xi)u, x_{u,\xi}\right\rangle] \quad \text{and} \quad g_u := e - \mathbb{E}[T(\xi)^\top x_{u,\xi}]\,.$$

A Controllable Lower Oracle. Instead of just taking any feasible point, a better estimate for the recourse function can be computed by making some iterations of a linear programming solver based on a primal-dual method. The oracle receives as additional input an error bound $\bar\eta_u \geqslant 0$ and stops the primal-dual solver as soon as it finds a point $x_{u,\xi}$ such that

$$W^\top x_{u,\xi} \leqslant q(\xi) \quad \text{and} \quad \left\langle d(\xi) - T(\xi)u, x_{u,\xi}\right\rangle - Q(u;\xi) \leqslant \bar\eta_u\,.$$

As shown in [22], for this oracle the subgradient error η^g is null and $f_u \in [f(u) - \bar\eta_u, f(u)]$ for any error bound $\bar\eta_u$ chosen by the user.

Asymptotically Exact Oracles. An oracle that is eventually exact everywhere can be built from the controllable lower oracle, simply by letting $\bar\eta_{u_k} \to 0$, thus forcing the error bound to vanish along iterations.

A smarter oracle, called *partly asymptotically exact*, requires eventual exactness only for some input points u_k. This is done by combining the three preceeding lower oracles, as follows. In addition to the input $(u,\bar\eta_u)$ the oracle receives a *target* γ_u and must compute f_u within the on-demand accuracy $\bar\eta_u$ only when the target is reached:

$$f_u \text{ is computed} \begin{cases} \text{as in the dumb lower oracle } (f_u \in [f(u)-\eta, f(u)]) \text{ if } f_u > \gamma_u\,, \\ \text{as in the controllable lower oracle } (f_u \in [f(u)-\bar\eta_u, f(u)]) \text{ if } f_u \leqslant \gamma_u\,. \end{cases}$$

The notation above emphasizes the fact that $\eta \geqslant 0$ may be unknown while $\bar\eta_u \geqslant 0$ is known and controllable. The third oracle (asymptotically exact), comes into play when the user sets the target as a goal of decrease for f at u_k and drives $\bar\eta_{u_k} \to 0$; see [22] for more details.

An Upper Oracle. Instead of considering all the random events (the sampling space may be infinite), for each given u a small finite subset can be drawn from Ξ. The recourse function is computed exactly only at those realizations to find the corresponding minimizers $x_{u,\xi}$ and approximate $f(\cdot)$ and a subgradient. Because of the random sampling, in (2.3) the functional error $\eta_u = f(u) - f_u$ has unknown sign and the subgradient error η^g may be positive. The oracle error bound exists and depends on the probability distribution. □

The following two upper oracles will be considered in Sections 7.2 and 7.3.

Example 2.3 (Chance-Constrained Programs) For a probability level $p \in (0,1]$, in

$$\begin{cases} \min \ \langle h, u \rangle \\ \text{s.t.} \ \ \mathbb{P}(Tu \leqslant b(\xi)) \geqslant p \end{cases}$$

the constraint is convex if ξ has a log-concave probability distribution. In some cases the constraint is also smooth but its evaluation includes computing a gradient which can be very costly, even for relatively low random vector dimensions, say 10. To allow for approximate calculations, in [1] the *improvement function*

$$f(u) = \max\{\langle h, u \rangle - \tau_1, -\ln(p) - \ln[\mathbb{P}(Tu \leqslant b(\xi))] - \tau_2\},$$

depending on a parameter $\tau \in \mathbb{R}^2$, is introduced. The upper oracle in [1] employs numerical integration and quasi-Monte Carlo techniques for which the error is bounded (and can be driven to zero at the expense of heavy computations). □

The following oracles from [24] include in particular eigenvalue optimization [12].

Example 2.4 (Composite Functions) All the functions above involve some maximization operation. In a more general setting, given a convex function $h(\cdot)$ that is positively homogeneous (like the max-function) and a smooth operator $c(\cdot)$, the objective in (1.1) can have the form $f(\cdot) = (h \circ c)(\cdot)$. Suppose $c : \mathbb{R}^n \to \mathbb{R}^m$ and let $Dc(\cdot)$ denote its Jacobian. Given $\hat{u} \in \mathbb{R}^n$, the function $F(\cdot;\hat{u}) : \mathbb{R}^m \to \mathbb{R}$ defined by $F(\cdot;\hat{u}) := h(c(\hat{u}) + Dc(\hat{u})(\cdot - \hat{u}))$ is used in the *composite bundle method* [24] to solve (1.1), which in this setting can be a nonconvex problem. Since computing the Jacobian matrix is expensive, using oracle information for the (convex) function $F(\cdot;\hat{u})$ eases calculations. With respect to the true f-information, nothing can be said about the error sign: the oracle can be of upper type. □

We will retain from these oracles that several situations can occur:

– *Dumb oracle*: not much is known about the (f_u, g_u) output. Such is the case of the dumb lower oracle and of the (totally dumb) upper oracle in Example 2.2.
– *Informative oracle*: in addition to f_u and g_u, the oracle returns some reliable upper bound $\bar{\eta}_u$ for η_u; this could be the case of Example 2.1 if the L-maximization is a constrained problem solved by a primal-dual algorithm. Then the primal-dual output is some x_u giving $f_u = L(x_u, u)$, together with an upper bound $M_u \geqslant f(u)$ obtained by dual arguments. We can set $\bar{\eta}_u := M_u - f_u$.
– *Controllable oracle*: in addition to u, the oracle receives $\bar{\eta}_u$ and must compute $f(u)$ within $\bar{\eta}_u$-accuracy. This situation has itself several subcases:
  · *Achievable* high accuracy. A smaller $\bar{\eta}_u$ implies an acceptable increase in computing time, like with the controllable lower oracle in Example 2.2.
  · *Partly achievable* high accuracy. It is acceptable to make exact calculations at "good" points, as in the partly asymptotically exact oracle in Example 2.2. By this token, $\bar{\eta}_u$ can be managed more aggressively. We shall see in Section 7.3 that the composite oracle in Example 2.4 fits this category.
  · *Exorbitantly* high accuracy. For instance if in Example 2.1 the $L(\cdot,u)$-maximization is a difficult, combinatorial and/or large scale problem, or when in Example 2.2 the sampling space is too large and/or the probability distribution too involved.

3 Parameters Characterizing a Bundle Method

Generally speaking, an algorithm to solve (1.1) constructs two sequences:
– a sequence of special iterates ($\{\hat{u}_k\}$ in our notation below) aimed at minimizing $f(\cdot)$;
– a corresponding sequence of subgradients $\{\hat{g}_k\}$ aimed at approaching $0 \in \mathbb{R}^n$, thus providing a certificate of optimality.

The subsequence $\{\hat{u}_k\}$ is extracted from the algorithm iterates $\{u_k\}$, collecting points that provide sufficient progress towards the goal of solving (1.1); for instance, by reducing the functional value. We describe this construction in the framework of proximal bundle methods, with the least possible references to the relation between the oracle errors and the exact function. This relation comes into play only *a posteriori*, to determine to which extent the algorithm really solves (1.1).

In order to introduce the general bundling mechanism in a manner that is clear for the reader, consider first the pure *cutting-plane* methods.

3.1 Cutting-Plane Methods

Having called the oracle at a number of points u_j, the cutting-plane algorithms [4, 15] accumulate linearizations

$$f_j^L(u) := f_{u_j} + \langle g_{u_j}, u - u_j \rangle \tag{3.1}$$

which satisfy $f_j^L(\cdot) \leqslant f(\cdot) + \eta_{u_j}^g$, easily obtained from (1.2). The next iterate solves the *master-program*, defined by the minimization of the cutting-plane model

$$\check{f}_k(\cdot) := \max \left\{ f_j^L(\cdot) \,:\, j = 1, \ldots, k \right\}.$$

As a result, the following set of *parameters* fully characterizes a cutting-plane method:

$$\texttt{pars} := \begin{cases} \text{the convex } \textit{model } \check{f}_k(\cdot) \text{ and} \\ \text{a measure of progress, used as optimality certificate.} \end{cases}$$

The measure of progress (for instance, the distance between $f(\cdot)$ and the model at the last iterate) is used to stop the algorithm. For future use, note that taking the maximum over $j = 1, \ldots, k$ in the cutting-plane function gives the useful bound:

$$\check{f}_k(\cdot) \leqslant f(\cdot) + \max_{j \leqslant k} \eta_{u_j}^g \quad \text{for all } k. \tag{3.2}$$

A well-known drawback of cutting-plane methods is their instability that makes these algorithms eventually stall and oscillate. We are interested in *stabilized* variants, such as [21, 11, 8, 3], whose sets `pars` include some stabilization devices to ensure *descent* for certain iterates and pinpoint the special subsequence involved in the convergence analysis:

$$\texttt{pars} := \begin{cases} \text{a convex model } f_k^M, \text{ possibly different from the cutting-plane one,} \\ \text{a measure of progress to stop the algorithm,} \\ \text{a } \textit{stability center } \hat{u}_k, \text{ a past iterate attracting the next one,} \\ \text{a proximal stabilization } \frac{1}{2t_k} |\cdot - \hat{u}_k|^2, \\ \text{other parameters and their updating rules, including } t_{k+1},\ f_{k+1}^M(\cdot), \text{ etc.} \end{cases}$$

The stabilization above is of the *proximal* type, more general stabilization terms are considered in [8]. The next iterate u_{k+1} is the (unique) minimum of a stabilized model function:

$$\min_{u\in\mathbb{R}^n} f_k^S(u)\,, \quad \text{for } f_k^S(u) := f_k^M(u) + \frac{1}{2t_k}|u-\hat{u}_k|^2. \tag{3.3}$$

In `pars` the model $f_k^M(\cdot)$ *does not need* to be one based on cutting planes. Traditional bundle methods use models satisfying $f_k^M(\cdot)\leqslant \check{f}_k(\cdot)$ as in (7.1) below; and for eigenvalue optimization, spectral bundle methods [12,23] use non-polyhedral models. The main requirement limiting the choice of $f_k^M(\cdot)$ is pragmatic: problem (3.3) should be easily solvable. However, the model cannot be a totally arbitrary convex function: it should provide a reasonable approximation of $f(\cdot)$. Relevant assumptions on the model will be given later, as the need arises; for now we just mention that for lower oracles (i.e., when (2.3) holds), because $\check{f}_k(\cdot)\leqslant f(\cdot)$, the cutting-plane model (i.e., setting $f_k^M(\cdot) := \check{f}_k(\cdot)$) satisfies the relation

$$f_k^M(\cdot)\leqslant f(\cdot) \quad \text{for all } k. \tag{3.4}$$

When this inequality holds we shall say that a *lower model* is available.

3.2 Ensuring Descent for a Subsequence

Due to its importance for the convergence analysis, we single the descent criterion out from the parameter set and write it as a rule depending on objects specified by another set, denoted by `desc` below. We first define elements in this set for the exact oracles, to motivate their extension to the inexact setting.

3.2.1 Exact Oracles

When the oracle calculations are exact, descent is determined by observing progress towards the goal of minimizing the objective function. Progress can be measured relative to either the model, or some nominal reference value, or the objective function itself. The three corresponding measures are respectively called and denoted *model decrease* δ_k^M, *nominal decrease* δ_k^N, and *effective decrease* δ_k^E.

With exact oracles, the model decrease is

$$\delta_k^M = f(\hat{u}_k) - f_k^M(u_{k+1})\,.$$

For the nominal decrease, there are several possibilities, all yielding a non-negative measure because (3.3) minimizes the stabilized model $f_k^S(\cdot)$ therein:

$$(0\leqslant)\delta_k^N = \begin{cases} f(\hat{u}_k) - f_k^M(u_{k+1})\ , \text{ or} \\ f(\hat{u}_k) - f_k^S(u_{k+1}) = \delta_k^M - \frac{1}{t_k}|u_{k+1}-\hat{u}_k|^2\ , \text{ or} \\ \delta_k^M - \frac{\alpha_k}{t_k}|u_{k+1}-\hat{u}_k|^2, \text{ for some } \alpha_k\in[0,1], \text{ or} \\ f_k^M(\hat{u}_k) - f_k^M(u_{k+1}), \text{ or yet some other variant.} \end{cases} \tag{3.5}$$

The third option above subsumes the first two choices, by setting respectively $\alpha_k = 0$ or 1. Finally, the effective decrease has the expression

$$\delta_k^E = f(\hat{u}_k) - f(u_{k+1})\,.$$

We shall see that driving the model decrease to zero is an important ingredient for the convergence analysis (Proposition 6.1 below). The other two measures are involved in the descent test: a *descent step* is performed if for a parameter $m \in (0,1)$ the relation

$$m\,\delta_k^N \leqslant \delta_k^E \tag{3.6}$$

is satisfied; otherwise the step is declared *null*. This is a sort of Armijo rule, used in linesearches or trust-region algorithms for smooth optimization. Having for example the gradient at the current iterate $\hat{u}_k$, a linear estimate is chosen for δ_k^N and (3.6) reduces to

$$m\,\langle \nabla f(\hat{u}_k), \hat{u}_k - u_{k+1}\rangle \leqslant f(\hat{u}_k) - f(u_{k+1})\,.$$

In our nonsmooth setting, the descent rule depends on specific choices for m, δ_k^N, and δ_k^E, to decide between making

either a *descent step*	or	a *null step*
(3.6) holds		(3.6) does not hold
move the center: $\hat{u}_{k+1} = u_{k+1}$		keep the current center: $\hat{u}_{k+1} = \hat{u}_k$.

3.2.2 Inexact Oracle

When the oracle output has some error neither $f(u_{k+1})$ nor $f(\hat{u}_k)$ are available: only estimates $f_{u_{k+1}}$ or $f_{\hat{u}_k}$ are at hand and some proxies need to be found for the measures above.

Regarding the model decrease, the substitute for $f(\hat{u}_k)$ therein will be a "level" ℓ_k:

$$\delta_k^M := \ell_k - f_k^M(u_{k+1})\,. \tag{3.7}$$

Calculations become more transparent if some flexibility is allowed for the level, so we let:

$$\begin{cases} \ell_k \in [f_{\hat{u}_k}, f(\hat{u}_k)] & \text{if } \eta_u^g \equiv 0\text{, i.e., the oracle is of lower type, and} \\ \ell_k = f_{\hat{u}_k} & \text{otherwise.} \end{cases} \tag{3.8}$$

Therefore, the inequality $\ell_k \geqslant f_{\hat{u}_k}$ always holds. For each bundle variant the level choice is made precise in the parameter set `pars` and Theorem 4.4 below will show that ℓ_k is a natural candidate to estimate the optimal value of (1.1).

In order to define a proxy for the nominal decrease, first note that δ_k^M somehow gives the maximal possible reduction in the objective when the model is of lower type. So, at least when (3.4) holds, δ_k^N should not exceed δ_k^M. Furthermore, for the stability center to be moved often enough in (3.6) it is appealing to take $\delta_k^N < \delta_k^M$. Like in (3.5), we set

$$\delta_k^N = \delta_k^M - \frac{\alpha_k}{t_k}|u_{k+1} - \hat{u}_k|^2 \quad \text{for some } \alpha_k \geqslant 0$$

(more conditions, forcing $\alpha_k \leqslant 1$, will be imposed as needed; cf. (5.4)). At this stage an important consideration arises. For the subsequence of center function values to be monotone (and, hence, to ensure descent as intended), the substitute for the nominal decrease should be non-negative. Yet, Figure 3.1 (left part) shows that the noisy value $f_{\hat{u}_k}$ may lie below some $f_j^L(\hat{u}_k)$, hence below $\check{f}_k(\hat{u}_k)$, which itself is usually below $f(\hat{u}_k)$. If ℓ_k is chosen too low (right part of the picture), the nominal decrease may become negative. In extreme cases, the optimal value of (3.3) may even come above $f_{\hat{u}_k}$, indicating an inadequate $f_k^S(\cdot)$.
In [26] this latter phenomenon was avoided by a stricter descent test, necessitating a severe rule to control the oracle noise. By contrast, [13] initiated a track further developed by

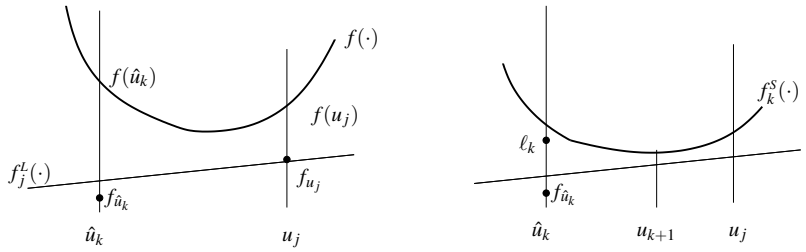

Fig. 3.1 Despite (3.1), $f_j^L(\hat{u}_k)$ may lie above $f_{\hat{u}_k}$

[18], coping with the case $f_{\hat{u}_k} - f_k^S(u_{k+1}) < 0$ via a nice idea: increase t_k, so as to diminish the attraction toward the (suspect) stability center. Lemma 5.1 below gives three equivalent conditions to ensure $\delta_k^N \geqslant 0$ and Corollary 5.2 shows that lower models always satisfy such conditions with lower oracles that are not totally dumb. In other cases, for example with upper oracles, some corrective action called of *noise attenuation* needs to be taken. To introduce complications gradually, we defer dealing with this issue to Section 5.

To make the Armijo rule (3.6) precise, the set of descent parameters should be

$$\texttt{desc} := \{m\,, \delta_k^N - \text{by choosing } \alpha_k \text{ and } \delta_k^M, \text{ and } \delta_k^E\}\,,$$

where to replace the effective decrease the two choices below have been proposed so far.

– **Bold effective decrease**: Take $\delta_k^E = f_{\hat{u}_k} - f_{u_{k+1}}$, the decrease observed for the function, regardless of its noise.

– **Realistic effective decrease**: Take $\delta_k^E = \hat{f}_k - f_{u_{k+1}}$ as in [9,7], for the "threshold" between two descent steps defined by:

$$\hat{f}_k := \max \left\{ \begin{array}{l} f_{\hat{u}_k} \\ \max_{j \leqslant k} f_j^M(\hat{u}_k) \text{ for iterations } j \text{ following the one generating } \hat{u}_k. \end{array} \right\} \tag{3.9}$$

The threshold can be computed by taking $\hat{f}_k := f_k^M(\hat{u}_k)$ after a descent step, and then setting $\hat{f}_k = \max\{f_k^M(\hat{u}_k), \hat{f}_{k-1}\}$ after a null step. The threshold is not a quantity attached to the center (it can increase at null steps); this nasty feature entails some technical difficulties. However, when (3.4) holds, the threshold does take a better account of reality.

Of these two choices, the bold one is globally satisfactory – and anyway is probably the only possible proposal when the oracle accuracy is not controllable and the error bound is unknown. On the other hand, the realistic one is appealing but has a strong requirement: the oracle error must vanish fast enough (see Section 7.1.3 below).

For lower oracles having a *known* error bound (such as the controllable lower oracle in Example 2.2) and lower models, we consider a third type of decrease:

– **Conservative effective decrease**: Take $\delta_k^E = (f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k}) - (f_{u_{k+1}} + \bar{\eta}_{u_{k+1}})$, involving the knowledge of the oracle error upper bounds $\bar{\eta}_{\hat{u}_k}$ and $\bar{\eta}_{u_{k+1}}$. The rationale is to force $f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} \geqslant f_j^M(\hat{u}_k)$ for all $j \geqslant k$, thereby eliminating the difficulty revealed by Figure 3.1; observe also that $f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k}$ remains fixed during null steps.

The corresponding method is the *Controllable Bundle Algorithm* 5.4. Thanks to the additional knowledge provided by the oracle ($\bar{\eta}_u$), this new method eventually solves (1.1) up to the accuracy of the oracle at descent steps.

4 Main Ingredients in the Algorithm

Coping with inexact oracles increases substantially the technicalities; in particular, the notation becomes necessarily fussy, as several f-values come into play at a given u: we have $f(u)$, $f^M(u)$, $f^S(u)$; if the oracle has been called at u, we also have f_u. To clarify notation, we adopt the convention that a superscript $(\cdot)^M$ [resp. $(\cdot)^S$, resp. a hat $\widehat{(\cdot)}$] connotes an item attached to the original model [resp. the objective function in (3.3), resp. the stability center].

4.1 Aggregate Objects and Algorithmic Pattern

Once (3.3) is solved to produce the next iterate, two key objects are the *aggregate linearization* $f^L_{-k}(\cdot)$ and *aggregate subgradient* $\hat{g}_k$ introduced below.

Lemma 4.1 (Aggregate objects) *Knowing that $f^M_k(\cdot)$ is convex, the unique solution of the master-program* (3.3) *is*

$$u_{k+1} = \hat{u}_k - t_k \hat{g}_k\,, \quad \textit{for some } \hat{g}_k \in \partial f^M_k(u_{k+1})\,. \tag{4.1}$$

The affine function

$$f^L_{-k}(u) := f^M_k(u_{k+1}) + \langle \hat{g}_k, u - u_{k+1}\rangle \tag{4.2}$$

is an underestimate of the model: $f^L_{-k}(\cdot) \leqslant f^M_k(\cdot)$.

Proof Since its objective function is finite-valued strongly convex, problem (3.3) has a unique solution characterized by the optimality condition $0 \in \partial f^S_k(u) = \partial f^M_k(u) + \frac{u-\hat{u}_k}{t_k}$, which gives (4.1). The inequality $f^L_{-k}(\cdot) \leqslant f^M_k(\cdot)$ is just the subgradient relation. □

As a consequence of (4.1) the nominal decrease in (3.7) has the equivalent expressions

$$\delta^N_k = \delta^M_k - \alpha_k t_k |\hat{g}_k|^2 = [\ell_k - f^M_k(u_{k+1})] - \alpha_k t_k |\hat{g}_k|^2 \text{ for some } \alpha_k \geqslant 0. \tag{4.3}$$

Without further specification of the sets `pars` and `desc`, an abstract algorithmic pattern (depending on these objects) can now be outlined.

Algorithmic Pattern 4.2(`pars`, `desc`)

Having `pars` and `desc`, a starting point u_1 is chosen. The oracle output (f_{u_1}, g_{u_1}) is available. Set $k = 1$ and initialize $\hat{u}_1 = u_1$.

STEP 1. Having the model and $t_k > 0$ defined by `pars`, solve (3.3) to obtain $\hat{g}_k$, u_{k+1} and $f^L_{-k}(\cdot)$ as in Lemma 4.1. Based on the definitions in `desc`, compute the nominal decrease (4.3) and determine the need of *noise attenuation*.

Loop in Step 1 until noise needs no more attenuation.

STEP 2. Call the oracle at u_{k+1} to obtain the output $(f_{u_{k+1}}, g_{u_{k+1}})$.

STEP 3. With the objects in `desc`, the Armijo-like descent test (3.6) decides to make

Descent step	or	*Null step*
(3.6) holds		(3.6) does not hold
set $\hat{u}_{k+1} = u_{k+1}$		set $\hat{u}_{k+1} = \hat{u}_k$
choose $f^M_{k+1}(\cdot) \in$ `pars`		choose $f^M_{k+1}(\cdot) \in$ `pars` satisfying $f^M_{k+1} \geqslant \max\{f^L_{-k}(\cdot), f^L_{k+1}(\cdot)\}$.

STEP 4. Increase k by 1 and loop to Step 1. □

As our aim is to state general principles for convergence, this is only an algorithmic *pattern*, independent of the bundle variant and of the corresponding specific choices in `pars` and `desc`. To emphasize this fact for the convergence analysis in Section 6 we shall just refer to Algorithmic Pattern 4.2, without mentioning the parameter and decrease sets. Section 7 reviews several variants for such sets, all yielding actually implementable algorithms:

$$\text{Specific Algorithm} = \text{Alg. Pattern } 4.2(\texttt{pars},\texttt{desc}) \text{ for given } \texttt{pars},\texttt{desc}\,.$$

In addition, to help understanding various abstract objects that appear in the analysis, before developing the general convergence theory we give a concrete instance, Algorithm 5.4, suitable for lower oracles with controllable accuracy like the one in Example 2.2.

Step 1 will be endowed with a stopping test based on the approximate optimality inequality (4.11) below. Also, depending on the oracle and on the model, Step 1 may need to incorporate a loop of noise attenuation. This loop ensures eventual non-negativity of δ_k^N, so that the Armijo condition (3.6) ressembles a descent test. We shall see in Section 5 how to deal with this issue for general oracles and models.

Naturally, the model cannot be totally arbitrary: Step 3 (right branch) imposes satisfaction of lower bounds. We will see that upper bounds are also required; they are somewhat linked with the choice of t_k and will arise in Section 6.2, specifically in relation (6.5) therein.

4.2 Measuring Optimality

The Algorithmic Pattern 4.2 can be viewed as a proximal method, in which a descent step updates the current "outer" iterate $\hat{u}_k$ to the next one $\hat{u}_{k+1}$ by performing a sequence of null steps, the "inner" iterations. We now use this interpretation to refer all the bundle information to the center $\hat{u}_k$ and establish fundamental approximate optimality conditions.

4.2.1 Shifting the Bundle Information

When analyzing a bundle method working with exact oracles it is convenient to translate the origin of $\mathbb{R}^n \times \mathbb{R}$ to the stability elements $(\hat{u}_k, f(\hat{u}_k))$, even though these elements change at each outer iteration. For inexact oracles, the level ℓ_k replaces the vertical origin $f(\hat{u}_k)$.

The translation to $(\hat{u}_k, \ell_k)$ is applied in particular to the aggregate linearization (4.2): in terms of the *aggregate linearization gap*

$$\hat{e}_k := \ell_k - f_{-k}^L(\hat{u}_k)\,, \tag{4.4}$$

and using the identity $f_k^M(u_{k+1}) = f_{-k}^L(\hat{u}) - \langle \hat{g}_k, \hat{u}_k - u_{k+1}\rangle$ from (4.2), the function in (4.2) has the equivalent form

$$f_{-k}^L(u) = \ell_k - \hat{e}_k + \langle \hat{g}_k, u - \hat{u}_k\rangle\,. \tag{4.5}$$

Figure 4.1 illustrates this construction for $\ell_k = f_{\hat{u}_k}$; observe in passing the useful relations

$$\hat{e}_k = \ell_k - f_k^M(u_{k+1}) - \langle \hat{g}_k, \hat{u}_k - u_{k+1}\rangle = \ell_k - f_k^M(u_{k+1}) - t_k|\hat{g}_k|^2\,, \tag{4.6}$$

coming from (4.1). At this point we introduce an important convergence parameter:

$$\phi_k := \hat{e}_k + \langle \hat{g}_k, \hat{u}_k\rangle = \ell_k - f_{-k}^L(\hat{u}_k) + \langle \hat{g}_k, \hat{u}_k\rangle\,. \tag{4.7}$$

As shown in Theorem 4.4 below, proving convergence for an algorithm amounts to

$$\text{finding a } K^\infty\text{-subsequence } \{(\phi_k, \hat{g}_k)\} \text{ converging to } (\phi, 0) \text{ with } \phi \leqslant 0\,, \tag{4.8}$$

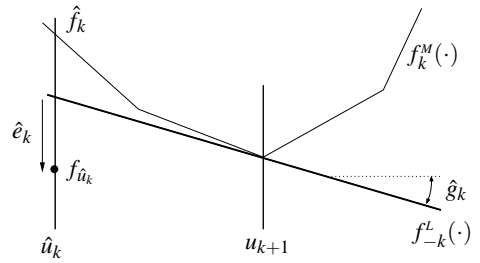

Fig. 4.1 Translation of the origin to the stability elements

for certain infinite iteration sets K^∞ generated by the considered algorithm. For this reason, the set `par` includes a stopping criterion checking if ϕ_k and $\hat{g}_k$ are sufficiently small.

Remark 4.3 (Duality gap interpretation of ϕ_k) For the primal problme (2.4) in Example 2.1, (1.1) comes from Lagrangian relaxation or column generation and ϕ_k has a nice interpretation as a duality gap. Recall that the dual function is given by

$$f(u) := \max_{x\in X} L(x,u)\,, \quad \text{with } L(x,u) := h(x) + \langle u, c(x)\rangle\,.$$

Calling x_j the primal point computed at u_j, the oracle output is

$$f_{u_j} = L(x_j,u_j) = h(x_j) + \big\langle u_j, c(x_j)\big\rangle \quad \text{and} \quad g_{u_j} = c(x_j)\,.$$

As the Lagrangian is affine with respect to u, $f_j^L(\cdot) = L(x_j,\cdot)$. Moreover, when the model is the cutting-plane function, a subgradient $\hat{g}_k$ of the max-function $f_k^M(\cdot)$ is a convex combination of active subgradients. Hence, for $\lambda_j \geqslant 0$ such that $\sum_j \lambda_j = 1$,

$$\hat{g}_k = \sum_j \lambda_j g_{u_j} = \sum_j \lambda_j c(x_j)\,,$$

where for each j such that $\lambda_j > 0$,

$$f_k^M(u_{k+1}) = f_j^L(u_{k+1}) = L(x_j,u_{k+1}) = h(x_j) + \big\langle u_{k+1}, c(x_j)\big\rangle\,.$$

Thus,

$$\begin{aligned} f_k^M(u_{k+1}) &= \textstyle\sum_j \lambda_j h(x_j) + \big\langle u_{k+1}, \sum_j \lambda_j c(x_j)\big\rangle && \text{[by convex combination]}\\ &= \textstyle\sum_j \lambda_j h(x_j) + \langle u_{k+1}, \hat{g}_k\rangle && \text{[by the expression above for } \hat{g}_k\text{]}\\ &= \ell_k - \hat{e}_k - \langle \hat{g}_k, \hat{u}_k - u_{k+1}\rangle\,. && \text{[by (4.6) and (4.1)]} \end{aligned}$$

Using the first identity in (4.7), we have therefore proved that

$$\sum_j \lambda_j h(x_j) = \ell_k - \hat{e}_k - \langle \hat{g}_k, \hat{u}_k\rangle = \ell_k - \phi_k\,. \tag{4.9}$$

Assume for simplicity that (2.4) is a linear program: $h(x) = \langle h,x\rangle$, $c(x) = Ax - a$, and introduce the primal point $\hat{x}_k := \sum_j \lambda_j x_j$. By (4.9), $\langle h,\hat{x}_k\rangle = \ell_k - \phi_k$ and, as ℓ_k from (3.8) is essentially a functional dual value,

$$\phi_k = \ell_k - \langle h,\hat{x}_k\rangle \text{ estimates the duality gap and}$$

$$\hat{g}_k = A(\hat{x}_k - a) \text{ is a constraint value.}$$

These identities justify the fact that both ϕ_k and $\hat{g}_k$ should be driven to 0 (at least with an exact oracle, for which $\ell_k = f(\hat{u}_k)$). □

4.2.2 Convergence: What it Means

Now we address the question whether (4.8) is any good for convergence of our algorithm. A prerequisite is of course that the model should agree with the true function to some extent. This is the reason for the assumption below, a weakened form of (3.4):

$$\text{for some } \eta^M \geqslant 0 \text{ the inequality } f_k^M(\cdot) \leqslant f(\cdot) + \eta^M \text{ holds for all } k\,. \tag{4.10}$$

In other words, the model is allowed to overestimate the true function, but "in a bounded way"; the inequalities (4.11) below show that a smaller η^M yields a more accurate algorithm. When the model is the cutting-plane approximation, the bound (3.2) guarantees (4.10) whenever η_u^g is bounded; in particular such is the case for lower oracles (the cutting-plane model is lower: (3.2) holds with null $\eta_{u_j}^g$, by (2.3)).

Theorem 4.4 (Conditions for approximate optimality) *Suppose the model satisfies* (4.10) *and the Algorithmic Pattern 4.2 generates a K^∞-subsequence $\{(\phi_k, \hat{g}_k)\}$ satisfying* (4.8)*. Then the level defined in* (3.8) *eventually estimates the infimal value of f, namely*

$$\limsup_{k\in K^\infty} \ell_k \leqslant \inf f(\cdot) + \phi + \eta^M \leqslant \inf f(\cdot) + \eta^M\,. \tag{4.11}$$

As a result, let $K' \subset K^\infty$ be such that $\{\hat{u}_k\}_{K'}$ has a limit $\hat{u}$ and consider the corresponding asymptotical oracle error at descent steps

$$\eta_\infty := \liminf_{k\in K'} \eta_{\hat{u}_k}\,. \tag{4.12}$$

Then $\hat{u}$ is an $(\eta_\infty + \eta^M)$-solution to problem (1.1)*.*

Proof Use Lemma 4.1 and (4.10) in (4.5): for all u and all k,

$$f(u) + \eta^M \geqslant f_k^M(u) \geqslant f_{-k}^L(u) = \ell_k - \phi_k + \langle \hat{g}_k, u\rangle\,.$$

Hence $\langle \hat{g}_k, u\rangle - f(u) \leqslant \phi_k - \ell_k + \eta^M$; take the supremum over u to obtain that

$$\sup_u \{\langle \hat{g}_k, u\rangle - f(u)\} =: f^*(\hat{g}_k) \leqslant \phi_k - \ell_k + \eta^M\,.$$

In view of (4.8), by closedness of the conjugate $f^*(\cdot)$,

$$f^*(0) \leqslant \liminf_{k\in K^\infty} f^*(\hat{g}_k) \leqslant \lim_{k\in K^\infty} \phi_k + \liminf_{k\in K^\infty} (-\ell_k) + \eta^M = \phi - \limsup_{k\in K^\infty} \ell_k + \eta^M\,,$$

which is just (4.11) since the value of the conjugate function at zero satisfies $f^*(0) = \inf f(\cdot)$. To see the final statement, given the iterate's subsequence defined over $K' \subset K$, consider $k \in K'$ and use (3.8) in the first line of (1.2), written at $u = \hat{u}_k$:

$$f(\hat{u}_k) - \eta_{\hat{u}_k} = f_{\hat{u}_k} \leqslant \ell_k\,,$$

Passing to the limit yields the desired relation, by lower semicontinuity of $f(\cdot)$:

$$f(\hat{u}) - \eta_\infty \leqslant \limsup_{k\in K'} [f(\hat{u}_k) - \eta_{\hat{u}_k}] \leqslant \limsup_{k\in K'} \ell_k \leqslant \inf f(\cdot) + \eta^M\,. \qquad \square$$

Instead of the asymptotic condition (4.8), seemingly introduced for the first time in [18], convergence for bundle methods has always been established by

$$\text{finding a } K^\infty\text{-subsequence } \{(\hat{e}_k, \hat{g}_k)\} \text{ converging to (0,0).} \tag{4.13}$$

The difference is subtle indeed: both properties are equivalent when $\{\hat{u}_k\}$ is bounded; precisely, condition (4.8) gives an elegant argument in the unbounded case, which was overlooked before, for example in [5, 14].

The inequality in Theorem 4.4 gives some insight on the role played by ℓ_k, in particular on the reason for its definition (3.8). The aim of the Algorithmic Pattern 4.2 is of course to estimate as accurately as possible the optimal value and a solution of (1.1). The latter is done by means of the stability center $\hat{u}_k$ while the former can be accomplished in various ways. A straightforward approximation for the optimal value is $f_{\hat{u}_k}$, but better can be done when both the oracle and the model are of lower type (both (2.3) and (3.4) hold). In this case, the value $\hat{f}_k$ from (3.9) is more accurate than $f_{\hat{u}_k}$ and having in (4.11) that $\ell_k = \hat{f}_k$ ensures by (3.9) that the estimate is the largest available functional value. Having said that, notice that such definition for the level is acceptable only if $\hat{f}_k$ satisfies the relations in (3.8);

$$\hat{f}_k \in [f_{u_k}, f(u_k)] \text{ for lower oracles and } \hat{f}_k = f_{u_k} \text{ otherwise.}$$

The relation $\hat{f}_k \geqslant f_{\hat{u}_k}$ always holds by the definition (3.9). For lower oracles and lower models, (3.4) ensures in addition that $\hat{f}_k \leqslant f(\hat{u}_k)$ and, hence, it is possible to set $\ell_k = \hat{f}_k$.

Theorem 4.4 also clarifies how a "partly asymptotically exact" oracle can yield optimal limit points, without any error in spite of the oracle inexactness. If the oracle is eventually exact only at descent steps, $\eta_\infty = 0$ and $f(\hat{u}_k) - f_{\hat{u}_k} \to 0$. If the model satisfies (3.4) then the error η^M is also null and, by (3.8), the only possible value for ϕ is zero (like in (4.13)).

5 On Noise Management and a Concrete Instance

We now revisit Step 1 in the Algorithmic Pattern 4.2 under the light shed by Theorem 4.4 and make precise the aforementioned *noise attenuation* loop. Also, since the convergence theory in Section 6 is developed for the abstract framework of Algorithmic Pattern 4.2, we provide here a particular instance, giving specific values to the elements in the sets `pars` and `desc`. This is the Controllable Bundle Algorithm 5.4, a new method with the ability of controlling the oracle accuracy that will be used as an example all along Section 6.

5.1 Properties of the Aggregate Gap

Remember from Figure 4.1 that δ_k^M (and, in view of (4.3), δ_k^N) may be negative, in which case (3.6) is not a genuine descent test. We start by noting that the combination of (4.4), (4.2) and (4.1) gives for the model decrease (3.7) the equivalent expression

$$\delta_k^M = \hat{e}_k + t_k |\hat{g}_k|^2 . \tag{5.1}$$

This rewriting suggests that a small δ_k^M will bring the traditional convergence property (4.13) close to hand if $\hat{e}_k$ is "not too negative" (driving δ_k^M to 0 is quite a relevant convergence property, see Proposition 6.1 below).

The need of attenuating noise in Step 1 will be detected by means of the aggregate gap (4.6), whose sign properties and relation with objects in the set `desc` are studied next.

Lemma 5.1 (Aggregate Gap Properties Relevant for Noise Detection) *In the Algorithmic Pattern 4.2, consider the level, best function value, model and nominal decreases, and gap defined, respectively, in* (3.8)*,* (3.9)*,* (3.7)*,* (4.3)*, and* (4.4)*. The following holds.*

(i) $\hat{e}_k \geqslant \ell_k - \hat{f}_k$ *at all iterations.*

(ii) Satisfaction of the inequality

$$\hat{e}_k \geqslant -\beta_k t_k |\hat{g}_k|^2 \text{ for some } \beta_k \in [b, 1-b] \text{ with } b \in (0, \tfrac{1}{2}], \tag{5.2}$$

is equivalent to any of the relations below

$$\delta_k^M \geqslant \max\left\{\hat{e}_k, (1-\beta_k)t_k|\hat{g}_k|^2\right\} \iff \delta_k^N \geqslant \Big(1-(\alpha_k+\beta_k)\Big)t_k|\hat{g}_k|^2 . \tag{5.3}$$

In particular, whenever (5.2) *holds the model decrease is non-negative.*
Furthermore, if we assume in (4.3) *and* (5.2) *that*

$$\alpha_k + \beta_k \leqslant 1 - b , \tag{5.4}$$

whenever (5.2) *holds the nominal decrease is non-negative too.*

(iii) If the model satisfies (4.10) *then* $\hat{e}_k \geqslant -\big(\eta_{\hat{u}_k} + \eta^M\big)$.

Proof By definition (4.4) and the model subgradient inequality in Lemma 4.1,

$$\hat{e}_k = \ell_k - f^L_{-k}(\hat{u}_k) \geqslant \ell_k - f_k^M(\hat{u}_k) . \tag{5.5}$$

The first item follows from adding $\pm\hat{f}_k$ to the right hand side, recalling the definition for $\hat{f}_k$ in (3.9). The second item follows from some simple algebra using (5.1) and (4.3).
For the third item, use the level definition (3.8) and the model assumption (4.10) in (5.5) to write $\hat{e}_k \geqslant f_{\hat{u}_k} - f_k^M(\hat{u}_k) \geqslant f_{\hat{u}_k} - f(\hat{u}_k) - \eta^M$ and (2.3) ends the proof. □

In view of the interval for β_k in (5.2), condition (5.4) implies that $\alpha_k \in [0,1]$, like in (3.5) for the exact oracle setting. Lemma 5.1(ii) justifies the use of condition (5.2) to detect when a corrective action needs to be taken to ensure non-negativity of the nominal decrease in the Armijo test. The condition will be incorporated in the Algorithmic Pattern 4.2 as follows.

STEP 1. Having the model and t_k defined by `pars`, solve (3.3) to obtain $\hat{g}_k$, u_{k+1} and $f^L_{-k}(\cdot)$ as in Lemma 4.1. Based on the definitions in the set `desc`, compute the gap as in (4.4) and determine the need of *noise attenuation*:

Noisy Iteration	or	*Forthcoming Serious or Null Step*
(5.2) does not hold		(5.2) holds
Keep $\begin{cases} \hat{u}_{k+1} = \hat{u}_k \\ f^M_{k+1}(\cdot) = f^M_k(\cdot) \end{cases}$		Go to Step 2
Set $t_{k+1} > t_k$		
Repeat Step 1		

(5.6)

When the test in Step 1 determines that noise at the stability center $\hat{u}_k$ became too large, it is advisable to *correct* $\hat{u}_k$ rapidly, or to detect inexact optimality of $\hat{u}_k$ and stop the algorithm. This twofold goal is achieved by a simple idea: increase the stepsize t_k sharply (and inhibit any decrease until the next descent step, cf. (6.14)). As shown by Corollary 5.3 below, either a new stability center will be obtained or inexact optimality will be achieved. Figure 5.1 illustrates an extreme situation: the minimal value of $f_k^M(\cdot)$ may lie above $f_{\hat{u}}$. This clearly indicates that this latter value is totally deceiving. Unfortunately, the algorithm can only terminate (the solution of (3.3) stops at a minimizer of $f_k^M(\cdot)$ when t_k becomes large).

We now make use of the aggregate gap properties in Lemma 5.1 to analyze when the noise attenuation test can be dismissed or executed a finite number of times.

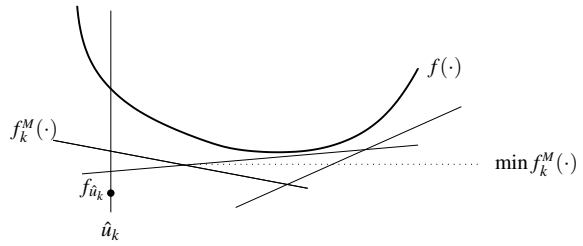

Fig. 5.1 When $\min f_k^M(\cdot) > f_{\hat{u}}$, the algorithm is blocked

Corollary 5.2 (Lower Models and Various Oracles) *Suppose Algorithmic Pattern 4.2 with Step 1 from* (5.6) *uses a lower model:*

$$\textit{in the set } \texttt{pars} \textit{ the model satisfies } (3.4).$$

(i) There is no need of noise attenuation and all iterations satisfy automatically (5.2) *with* β_k *arbitrary,* $b = 0$*, so* $\alpha_k \in [0,1]$ *in* (4.3)*, whenever one of the conditions below hold.*

(ia) In the parameter set the level from (3.8) *is given by*

$$\texttt{pars} \ni \ell_k := \hat{f}_k \textit{ from } (3.9)$$

(recall that such a definition is possible in particular for lower oracles and models: both (2.3) *and* (3.4) *hold.)*

(ib) The oracle is exact:

$$(2.3) \textit{ holds with } \eta_u \equiv 0\,.$$

(ii) The noise attenuation loop is finite if the oracle is lower and asymptotically exact at descent steps:

$$(2.3) \textit{ holds and in } (4.12)\ \eta_\infty = 0 \textit{ for the set } K' := \{k : \textit{satisfying } (3.6)\}. \tag{5.7}$$

Proof Condition (ia) implies (5.2), by items (i) and (ii) in Lemma 5.1. Condition (ib) follows from the first one, noting that when the oracle is exact in (3.8) we have that $\ell_k = f(\hat{u}_k)$, while in (3.9) we have that $\hat{f}_k = f(\hat{u}_k)$, because the model is lower. The second item uses similar arguments, reasoning asymptotically for k satisfying (3.6). □

Regarding (5.7), notice that if the subsequence of descent steps is finite, the condition $\eta_\infty = 0$ therein in fact requires an exact evaluation of descent steps.

For models that are no longer lower and/or for dumb lower and upper oracles, the noise attenuation loop can be infinite. We now show how to use the important Theorem 4.4 to prove that the algorithm remains well defined.

Corollary 5.3 (Upper Oracles and Noisy Steps) *Consider Algorithmic Pattern 4.2 with Step 1 from* (5.6) *and assume in the set* pars *the model satisfies* (4.10)*.*

Either the noise attenuation loop always terminates or after some iteration $\hat{k}$ *the algorithm loops forever in Step 1: the set*

$$K^\infty := \{k \geqslant \hat{k} : \textit{ condition } (5.2) \textit{ does not hold}\}$$

is infinite. Then the last descent iterate $\hat{u} := \hat{u}_{\hat{k}}$ *is an* $(\eta_{\hat{u}_{\hat{k}}} + \eta^M)$*-solution to* (1.1)*.*

Proof Recall that, in the noise attenuation loop, $t_k \uparrow \infty$ and the stability center is maintained fixed to $\hat{u}$. The K^∞-sequence of aggregate gaps $\{\hat{e}_k\}$ is bounded below by Lemma 5.1(iii), and non-positive because (5.2) does not hold. Since the K^∞-iterates satisfy the negation of (5.2) and $\beta_k \geqslant b > 0$ therein,

$$|\hat{g}_k|^2 < -\frac{\hat{e}_k}{\beta_k t_k} \leqslant \frac{\eta_{\hat{u}_k} + \eta^M}{b\, t_k};$$

hence, $\hat{g}_k \to 0$ as t_k is driven to infinity for $k \in K^\infty$. Therefore, the limit of the K^∞-subsequence $\{\phi_k = \hat{e}_k + \langle \hat{g}_k, \hat{u}\rangle\}$ is $\phi = \lim_{K^\infty} \hat{e}_k \leqslant 0$, because (5.2) does not hold. So the convergence condition (4.8) is satisfied and Theorem 4.4 applies, as stated. □

When Step 1 is given by (5.6) and the model choice satisfies (the very reasonable) assumption (4.10), Corollary 5.3 showed validity of the following alternative for the Algorithmic Pattern 4.2, for any type of oracles:

– either Step 1 is repeated a finite number of times until an iterate for which a descent or a null step will be made;
– or the loop in Step 1 is infinite and the last descent iterate is optimal up to the oracle and model precision.

For this reason only the remaining two cases are considered in the convergence analysis in Section 6. Namely, infinitely many descent steps or an infinite tail of consecutive null steps.

5.2 Controllable Bundle Method

We now state a concrete algorithm, suitable for controllable lower oracles like the one in Example 2.2: high accuracy is possible, yet it is preferable not to make exact calculations, due to the computational burden involved.

Oracle assumptions. Oracle of lower type ((2.3) holds because in (1.2) $\eta_u^g \equiv 0$) with accuracy guaranteed to be below the input error bound $\bar{\eta}_{u_k}$, sent to the oracle together with the evaluation point u_k to obtain $f_{u_k} \in [f(u_k) - \bar{\eta}_{u_k}, f(u_k)]$ and an approximate subgradient g_{u_k}.

Parameter and descent sets. The main novelty in the parameter set for this variant is in the specific choice of the level and the accuracy control of the oracle:

$$\texttt{pars} := \begin{cases} \text{the model } f_k^M = \max_{j\in J_k} f_j^L(\cdot), \text{ for an index set } J_k \subset \{-(k-1)\}\cup\{1,\dots,k\}; \\ \text{a stopping test checking if } \phi_k \text{ and } \hat{g}_k \text{ are sufficiently small;} \\ \text{the proximal stabilization } \frac{1}{2t_k}|\cdot - \hat{u}_k|^2 \text{ and an updating rule for } t_k\text{:} \\ \text{- if descent step, } t_{k+1} \geqslant t_k \\ \text{- if null step, } t_{k+1} = \max\{\underline{t}, \sigma t_k\} \text{ for } \sigma \in (0,1] \text{ and } \underline{t} > 0; \\ \text{the current stability center } \hat{u}_k \text{ and its level } \ell_k := \hat{f}_k \text{ from (3.9);} \\ \text{a rule to update the oracle error bound: } \bar{\eta}_{u_{k+1}} = \bar{\eta}_{\hat{u}_k} + f_{\hat{u}_k} - \ell_k. \end{cases} \tag{5.8}$$

Due to the oracle assumptions, the cutting-plane model in `pars` is lower, and in particular satisfies (4.10) with $\eta^M \equiv 0$. By the first condition in Corollary 5.2 there is no need of noise attenuation in Step 1. Also, since by definition (3.9) $\hat{f}_k \geqslant f_{\hat{u}_k}$, the rule in Step 2 makes the sequence $\{\bar{\eta}_{u_k}\}$ of error bounds nonincreasing.

For the Armijo-like descent test we take the set

$$\texttt{desc} = \left\{ \begin{array}{l} m \in (0,1), \alpha_k \in [0,1] \\ \delta_k^N := \ell_k - f_k^M(u_{k+1}) - \alpha_k t_k |\hat{g}_k|^2 \\ \delta_k^E := f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - f_{u_{k+1}} - \bar{\eta}_{u_{k+1}}. \end{array} \right\} \tag{5.9}$$

The combination of these elements gives the following method, fully implementable.

Algorithm 5.4 *Controllable Bundle Method* (=Alg.Pattern 4.2 with `pars,desc` from (5.8), (5.9))
The user chooses the starting point u_1 and $t_1 \geqslant \underline{t}$. The oracle output (f_{u_1}, g_{u_1}) is available. Set $k=1$ and initialize $\hat{u}_1 = u_1$, $J_1 = \{1\}$, and $\ell_1 = f_{u_1}$.

STEP 1. *Solve the quadratic programming problem*

$$\min_{r,u} r + \frac{1}{2t_k}|u - \hat{u}_k|^2 \;\; s.t. \;\; r \geqslant f_j^L(u)\,, j \in J_k$$

to obtain $\hat{g}_k$, u_{k+1} and $f_{-k}^L(\cdot)$ as in Lemma 4.1. Compute ϕ_k as in (4.7)*; if ϕ_k and $|\hat{g}_k|$ are small enough, stop.*

STEP 2. *Update the oracle error bound $\bar{\eta}_{u_{k+1}} = \bar{\eta}_{\hat{u}_k} + f_{\hat{u}_k} - \ell_k$ and call the oracle with input $(u_{k+1}, \bar{\eta}_{u_{k+1}})$ to obtain the output $(f_{u_{k+1}}, g_{u_{k+1}})$.*

STEP 3. *Check the descent test* (3.6)*, or the equivalent relation:*

$$f_{u_{k+1}} + \bar{\eta}_{u_{k+1}} \leqslant f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - m(\ell_k - f_k^M(u_{k+1}) - \alpha_k t_k |\hat{g}_k|^2)$$

and perform one of the steps below:

Descent step	*or*	Null step
The above inequality holds		*The above inequality does not hold*
Set $\hat{u}_{k+1} = u_{k+1}$		*Set $\hat{u}_{k+1} = \hat{u}_k$*
Choose $J_{k+1} \supset \{k+1\}$		*Choose $J_{k+1} \supset \{k+1, -k\}$*
Set $\ell_{k+1} = f_{k+1}^M(u_{k+1})$		*Set $\ell_{k+1} = \max\{\ell_k, f_{k+1}^M(\hat{u}_{k+1})\}$*
Choose $t_{k+1} \geqslant t_k$.		*Choose $t_{k+1} \in \left[\max\left(\underline{t}, \sigma t_k\right), t_k\right]$.*

STEP 4. *Increase k by 1 and loop to Step 1.* □

Note that the oracle accuracy is automatically adjusted, a useful feature if the initial bound $\bar{\eta}_{u_1}$ was taken too large. Algorithm 5.4 requires calculations to be more precise whenever the model approximates the function value at the center better than the oracle itself (i.e., whenever $\ell_k > f_{\hat{u}_k}$). Since the sequence of error bounds is nonincreasing, if the user chooses to start with exact calculations ($\bar{\eta}_{u_1} = 0$), Algorithm 5.4 boils down to the classical proximal bundle method for exact oracles ($\ell_k = \hat{f}_k = f(\hat{u}_k)$ in this case). In Section 7.1.4 we derive a partly asymptotically exact version of Algorithm 5.4, driving $\eta_{\hat{u}_k} \to 0$, by taking in (5.9)

$$\delta_k^E := f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - f_{u_{k+1}}\,.$$

6 Convergence Analysis

Remember that, rather than proposing new algorithms, our purpose is to state a synthetic convergence theory for proximal bundle methods dealing with inexact oracles. We now make the corresponding analysis for the general Algorithmic Pattern 4.2, without stating neither the parameter set, nor the descent rule. References to Algorithm 5.4 will be given throughout the section to make statements less abstract and guide the reader until the final Theorem 6.11,

with a convergence framework suitable for various proximal bundle algorithms, provided that the elements in the sets `pars` and `desc` satisfy some minimal assumptions.

Our analysis is done for the Algorithmic Pattern 4.2 with Step 1 as in (5.6), supposing the stopping test is deactivated. Once that the noise attenuation issue has been settled down by Corollaries 5.2 and 5.3, the K^∞-subsequences generated by the algorithm that will satisfy (4.8) arise from the inner and outer iterations mentioned in Section 4.2. Like when dealing with exact oracles, we shall study conditions to put in place the following arguments:

(a) *Bundling*: null steps issued from the same center suitably improve the model;

(b) *Descent*: steps satisfying (3.6) will force the convergence property (4.8).

Intermediate stages can be established independently of the variant used to implement (3.6) and generate infinite subsequences for which (a) or (b) apply. The goal of this section is to highlight these stages, and to single out the assumptions required by each one of them.

6.1 Common Results

We start with the following technical Féjer-like identity, derived from writing (4.1) in the form $|u_{k+1}-u|^2=|\hat{u}_k-t_k\hat{g}_k-u|^2$ and expanding the square:

$$|u_{k+1}-u|^2-|\hat{u}_k-u|^2=t_k^2|\hat{g}_k|^2+2t_k\langle\hat{g}_k,u-\hat{u}_k\rangle\quad\text{for any } u\in\mathbb{R}^n. \tag{6.1}$$

Corollaries 5.2 and 5.3 already ruled out an infinite loop in Step 1 from (5.6). So the infinite sets K^∞ in (4.8) can only come from either a descent or a null step. Accordingly, separating descent iterates from a possible tail of null steps, we define

$$\begin{aligned}\hat{K}\quad&:=\{k\in\mathbb{N}:(3.6)\text{ is satisfied}\}\\ K^\infty&:=\begin{cases}\{k\in\mathbb{N}:k>\hat{k}\} & \text{if }\hat{K}\text{ is finite, with last element }\hat{k},\\ \hat{K} & \text{if }\hat{K}\text{ has infinite cardinality.}\end{cases}\end{aligned} \tag{6.2}$$

The general convergence property below does not require the model in `pars` to satisfy any condition. However, we do assume the model satisfies (4.10), so that the aforementioned corollaries apply and the algorithm's K^∞-subsequences are well defined.

Proposition 6.1 *Suppose that in the set* `pars` *the Algorithmic Pattern 4.2 has a bounded model and prox-stepsizes bounded away from zero, so that both* (4.10) *and*

$$t_k\geqslant\underline{t}>0,\quad\textit{for all } k \tag{6.3}$$

hold. If for any of the two index sets K^∞ from (6.2) *the model decrease eventually vanishes:*

$$\lim_{k\in K^\infty}\delta_k^M=0,$$

the convergence property (4.8) *holds for such a set.*

Proof As Corollary 5.3 applies, (5.2) holds for $k\in K^\infty$ and (5.3) yields that

$$\lim_{k\in K^\infty}t_k|\hat{g}_k|^2=0\quad\text{and}\quad\begin{cases}\lim_{k\in K^\infty}\hat{g}_k=0\text{ by (6.3)}\\ \lim_{k\in K^\infty}\hat{e}_k=0\text{ by (5.1)}.\end{cases}$$

Therefore, for (4.8) to hold, we only need to show that $\phi=\lim_{K^\infty}\phi_k\leqslant 0$. When the index set is $K^\infty=\{k>\hat{k}\}$ this is direct from passing to the limit in the identity $\phi_k=\hat{e}_k+\langle\hat{g}_k,\hat{u}\rangle$ from (4.7), because $\hat{u}$ remains fixed to the last descent iterate. When the index set is $K^\infty=\hat{K}$,

take two consecutive indices k_1 and k_2 in $\hat{K}$ and apply (6.1) written with $k = k_2$ (so that $\hat{u}_{k_2} = u_{k_1+1}$) to obtain the identity

$$|u_{k_2+1} - u|^2 - |u_{k_1+1} - u|^2 = t_{k_2} z_{k_2}(u)\,, \text{ for } z_k := t_k|\hat{g}_k|^2 + 2\,\langle \hat{g}_k, u - \hat{u}_k\rangle\,.$$

The summation over k gives that $-\infty < -|u_0 - u|^2 \leqslant \sum_{k\in\hat{K}} t_k z_k$. Existence of some $\kappa > 0$ such that $z_k \leqslant -\kappa$ for all $k \in \hat{K}$ would imply $\sum_{k\in\hat{K}} t_k < +\infty$, which is impossible because of (6.3). Thus we have proved the relations

$$0 \leqslant \limsup_{k\in\hat{K}} z_k = \limsup_{k\in\hat{K}} \left(t_k|\hat{g}_k|^2 + 2\,\langle \hat{g}_k, u\rangle - 2\,\langle \hat{g}_k, \hat{u}_k\rangle\right) = -2\liminf_{k\in\hat{K}} \langle \hat{g}_k, \hat{u}_k\rangle\,,$$

because $t_k|\hat{g}_k|^2 \to 0$. Since $\hat{e}_k \to 0$, passing to the limit in (4.7) gives $\phi = \liminf_{\hat{K}} \phi_k \leqslant 0$. □

To relate this result with previous ones in the bundle literature, it is convenient to recall the ε-subdifferential in Convex Analysis. By (4.5) and Lemma 4.1,

$$f_k^M(u) \geqslant f_{-k}^L(\hat{u}_k) + \langle \hat{g}_k, u - \hat{u}_k\rangle = \ell_k - \hat{e}_k + \langle \hat{g}_k, u - \hat{u}_k\rangle\,, \quad \text{for all } u \in \mathbb{R}^n\,.$$

If the model satisfies (4.10), then (3.8) combined with the first line of (1.2) implies that $f(u) + \eta^M \geqslant f(\hat{u}_k) - \eta_{\hat{u}_k} - \hat{e}_k + \langle \hat{g}_k, u - \hat{u}_k\rangle$, i.e.,

$$\text{(4.10) implies that } \hat{g}_k \in \partial_{\varepsilon_k} f(\hat{u}_k) \text{ for } \varepsilon_k := \hat{e}_k + \eta_{\hat{u}_k} + \eta^M \geqslant 0 \tag{6.4}$$

by Lemma 5.1(iii).

It should be noted that the arguments in Proposition 6.1 *do not* extend the standard proof of bundle methods, used for $K^\infty = \hat{K}$ when the oracle is exact. Namely, such a proof is based on the property $\hat{g}_k \in \partial_{\hat{e}_k} f(\hat{u}_k)$ (not valid for inexact oracles), which allows a refinement of (6.1). Then, if the speed of convergence of δ_k^M to 0 can be assessed, better results are obtained: a weaker assumption on the stepsizes is possible, as well as showing convergence of the full sequence $\{\hat{u}_k\}$ when $\operatorname{Argmin} f(\cdot) \neq \emptyset$. Not unexpectedly, a variant of Proposition 6.1 recovers these two results if the oracle noise is suitably controlled, via the asymptotic error at descent steps η_∞ introduced in (5.7).

Theorem 6.2 (Link with the (partly asymptotically) exact case) *Consider the Algorithmic Pattern 4.2 applied with an oracle of lower type that is asymptotically exact at descent steps, as in* (5.7)*. Suppose that in the set* `pars` *the model is lower and the stepsizes series diverges, so that both* (3.4) *and the condition*

$$\sum_{k\in K^\infty} t_k = \infty$$

are satisfied. The following holds.

(i) If $\lim_{k\in K^\infty} \delta_k^M = 0$, *then* $\liminf_{K^\infty} f(u_k) = \inf f(\cdot)$.

Suppose in addition that in `pars` *the prox-stepsize sequence is bounded from above.*

(ii) In the null-step-tail case ($K^\infty = \{k > \hat{k}\}$) *the last descent step* $\hat{u}_k \equiv u_{\hat{k}} =: \hat{u}$ *satisfies*

$$\hat{u} \text{ minimizes } f(\cdot),\ \lim_{k\in K^\infty} u_k = \hat{u}, \text{ and } \lim_{k\in K^\infty} f_{k-1}^M(u_k) = f(\hat{u})\,.$$

Furthermore, suppose both the oracle error and the model decrease series are convergent:

$$\text{in (5.7) } \sum_{k\in\hat{K}} \eta_{\hat{u}_k} < +\infty \text{ and in } \texttt{desc} \sum_{k\in\hat{K}} \delta_k^M < +\infty\,.$$

(iii) In the infinite-descent-step case ($K^\infty = \hat{K}$), for any limit point $\hat{u}$ of the sequence $\{\hat{u}_k\}_{k\in\hat{K}}$

$$\hat{u} \text{ minimizes } f(\cdot), \text{ and the whole sequence } \{\hat{u}_k\}_{k\in\hat{K}} \text{ converges to } \hat{u}.$$

Proof The assumptions on both the oracle and the model fit the setting in Corollary 5.2(ii), thus ensuring well definition of the sets K^∞. In (i), convergence of $\{\delta_k^M\}$ and (5.3) yield that $t_k|\hat{g}_k|^2 \to 0$ for the considered subsequence. Since the model is lower, (3.4) implies that $\eta^M = 0$ in (4.10) and, by (6.4), the inclusion $\hat{g}_k \in \partial_{\varepsilon_k} f(\hat{u}_k)$ holds with $\varepsilon_k := \hat{e}_k + \eta_{\hat{u}_k}$. Adding $\eta_{\hat{u}_k}$ to the left hand side inequality in (5.3) gives that $\varepsilon_k = \hat{e}_k + \eta_{\hat{u}_k} \leqslant \delta_k^M + \eta_{\hat{u}_k}$ and our assumption on $\eta_{\hat{u}_k}$ implies that $\varepsilon_k \to 0$. Then (i) is [5, Prop. 1.2], where $\hat{g}$ is denoted γ. To see (ii), note that by (4.1), $|u_{k+1} - \hat{u}|^2 = t_k^2|\hat{g}_k|^2 \to 0$ because the stepsizes are bounded above by assumption. As the model decrease vanishes, (3.7) gives that $\lim_k \ell_k = \lim_k f_k^M(u_{k+1})$. Together with the level definition (3.8) and the oracle assumption (5.7), which forces $\eta_{\hat{u}} = 0$,

$$f(\hat{u}) = f_{\hat{u}} \leqslant \lim_k f_k^M(u_{k+1}) \leqslant f(\hat{u})\,.$$

By lower semicontinuity, $f(\hat{u}) \leqslant \liminf_k f(u_{k+1})$ and (ii) follows. To prove (iii), observe first that $\hat{u}$ minimizes $f(\cdot)$ by (i). Then use that $\hat{g}_k \in \partial_{\varepsilon_k} f(\hat{u}_k)$ and write from (6.1)

$$\begin{aligned} |u_{k+1} - \hat{u}|^2 - |\hat{u}_k - \hat{u}|^2 &= t_k^2|\hat{g}_k|^2 + 2t_k\,\langle \hat{g}_k, \hat{u} - \hat{u}_k\rangle \\ &\leqslant t_k^2|\hat{g}_k|^2 + 2t_k[f(\hat{u}) - f(\hat{u}_k) + \varepsilon_k] \\ &\leqslant t_k^2|\hat{g}_k|^2 + 2t_k\varepsilon_k\,. \end{aligned}$$

The definition of ε_k and (5.1) yield that $t_k^2|\hat{g}_k|^2 + 2t_k\varepsilon_k = t_k(\delta_k^M - \hat{e}_k + 2\varepsilon_k) \leqslant 2t_k(\delta_k^M + \eta_{\hat{u}_k})$. For successive indices k_1 and k_2 in $\hat{K}$ summing the inequalities

$$|u_{k_2+1} - \hat{u}|^2 - |u_{k_1+1} - \hat{u}|^2 \leqslant 2t_k(\delta_k^M + \eta_{\hat{u}_k}),$$

together with the assumptions on $\{\eta_{\hat{u}_k}\}$ and $\{\delta_k^M\}$, implies that the rightmost side term forms a convergent series, so (iii) follows from [5, Prop. 1.3]. □

In view of Proposition 6.1, obtaining small δ_k^M will be our main concern in Section 6.3 below. Before we state conditions ensuring this property when $\hat{K}$ in (6.2) is finite.

6.2 Null-Step Tail

As the stability center remains fixed throughout the present subsection, we use the notation $\hat{u} := \hat{u}_k$. We assume a weakened form of (4.10), holding only at $\hat{u}$:

$$\text{for some } \hat{\eta}^M \geqslant 0 \text{ the inequality } f_k^M(\hat{u}) \leqslant f(\hat{u}) + \hat{\eta}^M \text{ holds for all } k\,. \tag{6.5}$$

Recall that for the concrete Algorithm 5.4, the stronger condition (4.10) always holds with $(\hat{\eta}^M =)\eta^M = 0$, because the oracle is lower and the algorithm uses a cutting-plane based model, which is also lower in this case.

The null-step situation, in (6.2) $K^\infty = \{k > \hat{k}\}$, just relies upon the memory effect implied by $f_{k+1}^M(\cdot) \geqslant f_{-k}^L(\cdot)$, triggered by the right branch in Step 3 of the Algorithmic Pattern 4.2. Accordingly, we consider an execution in which Step 3 systematically makes a null step, regardless of any descent test. We claim that in this case

$$\limsup_{k>\hat{k}} \left[f_{u_k} - f_{k-1}^M(u_k)\right] \leqslant 0\,; \tag{6.6}$$

a very important inequality indeed. In fact, two sources of errors make (1.1) difficult to solve: one comes from the oracle and one comes from the model. The asymptotic property (6.6) states that the model inexactness eventually vanishes. Then proving convergence when the sequence of stability centers is finite becomes easy.

Another important observation on the role of (6.6) refers to the nominal and effective decreases in (3.6). For simplicity, take $\alpha_k = 0$ in (4.3), so that $\delta_k^N = \delta_k^M$ from (3.7). Then the effective and nominal decreases satisfy the relation

$$\delta_k^E = \ell_k - f_{u_{k+1}} = \delta_k^M + f_k^M(u_{k+1}) - f_{u_{k+1}} = \delta_k^N - \left[f_{u_{k+1}} - f_k^M(u_{k+1})\right].$$

When the bracket becomes small, the effective and nominal decreases get close together. This is a little known point in bundle methods: a good (effective) decrease $\ell_k - f_{u_{k+1}}$ entails a more accurate model's approximation $f_{u_{k+1}} - f_k^M(u_{k+1})$.

To establish (6.6) we state first a technical result linking successive optimal values of the master-program (3.3), based on arguments similar to the exact oracle case.

Lemma 6.3 *Consider the Algorithmic Pattern 4.2 and suppose that in the set* `pars` *the model satisfies* (6.5) *and the prox-stepsize is not increased at null steps:*

$$t_k \leqslant t_{k-1} \quad \text{if at iteration } k-1 \text{ the Armijo test } (3.6) \text{ is not satisfied.}$$

For u_k and u_{k+1} obtained by a null step issued from the center $\hat{u}$ the following holds.

(i) $f_{k-1}^S(u_k) + \frac{1}{2t_{k-1}}|u_{k+1} - u_k|^2 \leqslant f_k^S(u_{k+1})$,

(ii) $f_{k-1}^S(u_k) + \frac{1}{2t_{k-1}}|\hat{u} - u_k|^2 \leqslant f(\hat{u}) + \hat{\eta}^M$,

(iii) $f_k^M(u_{k+1}) - f_{k-1}^M(u_k) \leqslant f_k^S(u_{k+1}) - f_{k-1}^S(u_k) + o_k$ *,where we have set*

$$o_k := \frac{\langle u_{k+1} - u_k, \hat{u} - u_k\rangle}{t_k} = \frac{t_{k-1}}{t_k}\,\langle \hat{g}_{k-1}, u_{k+1} - u_k\rangle\,. \tag{6.7}$$

Proof For (i) and (ii) we refer to [18, Lemma 3.3]. To see (iii), by the definition in (3.3),

$$f_k^S(u_{k+1}) - f_k^M(u_{k+1}) = \frac{1}{2t_k}|u_{k+1} - u_k + u_k - \hat{u}|^2\,.$$

Develop the square and use $t_k \leqslant t_{k-1}$ in the right hand side to see that

$$\frac{1}{2t_k}|u_{k+1} - u_k|^2 - o_k + \frac{1}{2t_k}|u_k - \hat{u}|^2 \geqslant -o_k + \frac{1}{2t_{k-1}}|u_k - \hat{u}|^2\,.$$

The result follows, because $f_k^S(u_{k+1}) - f_k^M(u_{k+1}) \geqslant -o_k + f_{k-1}^S(u_k) - f_{k-1}^M(u_k)$. □

With an exact oracle, (6.6) becomes $f(u_k) - f_{k-1}^M(u_k) \to 0$, a known result, see for instance [5, Prop. 4.3]. Typically, $f_{k-1}^M(u_{k-1}) = f(u_{k-1})$; so we can write this as

$$[f(u_k) - f(u_{k-1})] + [f_{k-1}^M(u_{k-1}) - f_{k-1}^M(u_k)] \to 0\,,$$

easily proved with the Lipschitz property of $f(\cdot)$ and $f^M(\cdot)$ (Lemma 6.3 turns out to imply $u_k - u_{k-1} \to 0$, see (6.10) below). In the inexact case, observed values of $f(\cdot)$ may behave erratically, as well as the successive models $f^M(\cdot)$. Since Lemma 6.3(i) implies a better behavior of the stabilized function $f^S(\cdot)$, item (iii) relates the model to the stabilized model.

Theorem 6.4 (Null steps) *Consider the Algorithmic Pattern 4.2 applied with an oracle having locally bounded inaccuracy:*

$$\forall R \geqslant 0,\ \exists\, \eta(R) \geqslant 0 \ \text{ such that } \ |u| \leqslant R \Longrightarrow \eta_u + \eta_u^g \leqslant \eta(R)\,. \tag{6.8}$$

Suppose that in the set `pars` *the model satisfies* (6.5) *and the prox-stepsize is updated so that, whenever at iteration $k-1$ the Armijo test* (3.6) *is not satisfied,*

$$\text{there exist positive } \underline{t} \text{ and } \sigma \in (0,1] \text{ such that } t_k \in \left[\max\left(\underline{t}, \sigma t_{k-1}\right), t_{k-1}\right]\,.$$

Then the asymptotic property (6.6) *holds.*

Proof We first establish the preliminary results (6.9) and (6.10) below. The prox-stepsize update satisfies the condition in Lemma 6.3. By item (i) therein, the sequence $\{f_{k-1}^S(u_k)\}$ is nondecreasing, hence bounded from below; say $f_{k-1}^S(u_k) \geqslant -M$ for all k. By Lemma 6.3(ii),

$$\frac{1}{2t_{k-1}}|\hat{u}-u_k|^2 \leqslant f(\hat{u}) + \hat{\eta}^M - f_{k-1}^S(u_k) \leqslant f(\hat{u}) + \hat{\eta}^M + M\,.$$

Using once more that prox-stepsizes do not increase at null steps, we obtain that the sequence $\{u_k\}$ is bounded. By (2.1) and (2.2), $g_{u_k} \in \partial_{\eta_{u_k}+\eta_{u_k}^g} f(u_k)$, and the oracle assumption (6.8) implies that $\{g_{u_k}\}$ is bounded ([14, Prop. XI.4.1.2]). Our assumption on the prox-stepsize implies in particular that (6.3) holds and, hence, $\hat{g}_k = (\hat{u}-u_{k+1})/t_k$ is also bounded:

$$\text{the sequences } \{u_k\},\ \{\hat{g}_k\} \text{ and } \{g_{u_k}\} \text{ are bounded}\,. \tag{6.9}$$

By Lemma 6.3(ii), the monotone sequence $\{f_{k-1}^S(u_k)\}$ is bounded from above and has a limit. Together with Lemma 6.3(i) and using once again the monotonicity of prox-stepsizes,

$$f_k^S(u_{k+1}) - f_{k-1}^S(u_k) \to 0 \quad \text{and} \quad u_{k+1}-u_k \to 0\,. \tag{6.10}$$

We now use these preliminary results to show (6.6). The right branch in Step 3 of the algorithmic pattern forces $f_k^M(\cdot) \geqslant f_k^L(\cdot)$ so, by (3.1),

$$f_{u_k} + \left\langle g_{u_k}, u-u_k\right\rangle = f_k^L(u) \leqslant f_k^M(u)\,.$$

In particular, when $u = u_{k+1}$

$$\begin{aligned} f_{u_k} - f_{k-1}^M(u_k) &= f_k^L(u_{k+1}) + \left\langle g_{u_k}, u_k-u_{k+1}\right\rangle - f_{k-1}^M(u_k) \\ &\leqslant f_k^M(u_{k+1}) + \left\langle g_{u_k}, u_k-u_{k+1}\right\rangle - f_{k-1}^M(u_k) \\ &\leqslant [f_k^S(u_{k+1}) - f_{k-1}^S(u_k)] + [\left\langle g_{u_k}, u_k-u_{k+1}\right\rangle] + o_k\,, \end{aligned}$$

by Lemma 6.3(iii).The results follows: by (6.9) and (6.10), the first two brackets tend to 0; and similarly for the last term, recalling our assumptions for the prox-stepsize and (6.7). □

Remark 6.5 (On boundedness) Assumption (6.8) could be refined as follows: the oracle is bounded for any infinite sequence of null steps. We shall see make use of this refinement for some concrete instances in Section 7 related to Examples 2.3 and 2.4.

Remark 6.6 (On the role of the lower bound $\underline{t}$) In the proof above the assumption (6.3), keeping the stepsizes bounded away from zero, was only used to establish boundedness of the sequence $\{\hat{g}_k\}$. This assumption can be dropped if the model is bounded everywhere, i.e., if (4.10) holds instead of (6.5). The reasoning goes as follows: in this case (6.4) gives that $\hat{g}_k \in \partial_{\varepsilon_k} f(\hat{u})$ with $\varepsilon_k = \hat{e}_k + \eta_{\hat{u}} + \eta^M$. By local boundedness of the ε_k-subdifferential and by (6.8), we only need to show that $\{\hat{e}_k\}$ is bounded. The latter results from boundedness of $\{f_k^S(u_{k+1})\}$: plug (4.1) into the expression (4.6) of $\hat{e}_k$ to obtain

$$\hat{e}_k = \ell_k - f_k^M(u_{k+1}) - t_k|\hat{g}_k|^2 \leqslant \ell_k - f_k^S(u_{k+1}) \leqslant \ell_k + M \leqslant f(\hat{u}) + M\,,$$

where the last inequality follows from (3.8) recalling that $\hat{u}_k = \hat{u}$.
Notwithstanding, for Proposition 6.1 to apply (6.3) needs to hold, unless the oracle is of lower type and partly asymptotically exact, thus fitting the assumptions in Theorem 6.2. □

6.3 The Role of the Descent Test and General Convergence Result

For the bundling argument, we now make use of the property (6.6) to analyze when the model decrease eventually vanishes. Recall that this argument enters the game when in (6.2) we have $K^\infty = \{k > \hat{k}\}$ – the Armijo test (3.6) does not hold. Since such a test depends on the effective decrease, below we give a sufficient condition for $\delta_k^M \to 0$ involving this decrease.

Proposition 6.7 (Effective Decrease and Bundling Mechanism) *In the setting of Theorem 6.4, suppose that for the level in the set* `pars` *and the effective decrease in the set* `desc`

$$\limsup_{k>\hat{k}} \left[\ell_k - f_{u_{k+1}} - \delta_k^E\right] \leqslant 0\,; \tag{6.11}$$

then $\lim_{\hat{k}<k\to\infty} \delta_k^M = 0$.

Proof By Corollary 5.3 and Lemma 5.1 for the null step tail (5.2) holds and the model and nominal decreases satisfy the inequalities in (5.3) for all $k > \hat{k}$. Subtract the identity $f_k^M(u_{k+1}) = \ell_k - \delta_k^M$ from both sides of the negation of (3.6) and use (4.3) to obtain

$$-\delta_k^E - f_{u_{k+1}} + f_{u_{k+1}} - f_k^M(u_{k+1}) > -\ell_k + \delta_k^M - m\,\delta_k^N\,.$$

Since in (4.3) the parameter $\alpha_k \geqslant 0$, $\delta_k^N \leqslant \delta_k^M$, and reordering terms we obtain that

$$(1-m)\delta_k^M < z_k + f_{u_{k+1}} - f_k^M(u_{k+1}) \text{ for } z_k := (\ell_k - f_{u_{k+1}}) - \delta_k^E\,.$$

By Theorem 6.4, the property (6.6) holds, together with (6.11) we obtain in the limit that

$$(1-m)\limsup \delta_k^M \leqslant \limsup\left[z_k + f_{u_{k+1}} - f_k^M(u_{k+1})\right] \leqslant \limsup z_k \leqslant 0\,.$$

The result follows, recalling that $m \in (0,1)$ and, by (5.3), $\delta_k^M \geqslant 0$. □

Remark 6.8 (Interpretation for Algorithm 5.4) The effective decrease can be defined in several ways (bold, realistic, conservative). With the help of condition (6.11) we now analyze the impact of those different definitions for the concrete algorithm given in Section 5.2.
For the *conservative* choice given in Algorithm 5.4, (6.11) is satisfied because $\ell_k = \hat{f}_k$ and

$$\begin{aligned}\ell_k - f_{u_{k+1}} - \delta_k^E &= \ell_k - f_{u_{k+1}} - (f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - f_{u_{k+1}} - \bar{\eta}_{u_{k+1}}) \\ &= \ell_k - f_{\hat{u}_k} - \bar{\eta}_{\hat{u}_k} + \bar{\eta}_{u_{k+1}} \\ &= 0\,,\end{aligned}$$

where the last equality follows from the updating rule $\bar{\eta}_{u_{k+1}} = \bar{\eta}_{\hat{u}_k} + f_{\hat{u}_k} - \ell_k$ for the error bounds. If we were to take the same level $\ell_k = \hat{f}_k$, but use instead the *realistic* decrease $\delta_k^E = f_{\hat{u}_k} - f_{u_{k+1}}$, condition (6.11) would not hold. Indeed, reasoning like above,

$$\begin{aligned}\ell_k - f_{u_{k+1}} - \delta_k^E &= \hat{f}_k - f_{u_{k+1}} - (f_{\hat{u}_k} - f_{u_{k+1}})\\ &= \hat{f}_k - f_{\hat{u}_k}\\ &\geqslant 0\,.\end{aligned}$$

The above inequality can be strict due to definition (3.9). In order to ensure (6.11) for this setting, the oracle should be asymptotically exact on descent steps, as in (5.7). Finally, with the *bold* choices from [18], $\ell_k = f_{\hat{u}_k}$ and $\delta_k^E = f_{\hat{u}_k} - f_{u_{k+1}}$, condition (6.11) holds but not necessarily (5.2) – this is straightforward from Lemma 5.1 and (6.11). For (5.2) to hold in this setting, the controllable bundle algorithm would need to incorporate the noise attenuation loop, replacing Algorithm 5.4's Step 1 by the one in (5.6). □

When the bundling argument applies, as above, satisfaction of (6.11) is easy to accomplish (taking for example $\delta_k^E = \ell_k - f_{u_{k+1}}$). By contrast, when the algorithm generates infinitely many descent iterates ($K^\infty = \hat{K}$ in (6.2)), the working horse is the Armijo test (3.6). For the model decrease to vanish the effective decrease needs to vanish too; this is (6.12) below, a property that cannot be imposed *a priori*, but needs to be shown case by case.

Proposition 6.9 (Effective Decrease and Descent Mechanism) *Consider the Algorithmic Pattern 4.2 and suppose that for the sets* `pars` *and* `desc` *the parameters* α_k, β_k *satisfy* (5.4). *If for infinitely many iterations the Armijo test* (3.6) *is satisfied and*

$$\lim_{k\in\hat{K}} \delta_k^E = 0 \quad \text{for } \hat{K} \text{ from (6.2)}, \tag{6.12}$$

then $\lim_{k\in\hat{K}} \delta_k^M = 0$.

Proof By Corollary 5.3, condition (5.2) holds and by Lemma 5.1(ii) the model and nominal decreases satisfy the inequalities in (5.3) for all $k \in \hat{K}$. In view of the assumption (5.4), both δ_k^M and $\delta_k^N \geqslant 0$ and satisfaction of (3.6) gives that

$$m(1-\alpha_k-\beta_k)t_k|\hat{g}_k|^2 \leqslant m\,\delta_k^N \leqslant \delta_k^E\,, \quad \text{for } k \in \hat{K}\,.$$

The result follows, because $\alpha_k + \beta_k < 1$ by (5.4) and $\delta_k^M = \delta_k^N + \alpha_k t_k |\hat{g}_k|^2$ by (4.3). □

Since with an exact oracle taking $\delta_k^E = f(\hat{u}_k) - f(u_{k+1})$ is natural, satisfaction of (6.12) is direct from having $f(\cdot)$ – hence $f(\hat{u}_k)$ – bounded from below:

$$\text{in (1.1)} \quad \inf f(\cdot) > -\infty\,. \tag{6.13}$$

We now show that this condition also implies (6.12) for our concrete algorithm.

Remark 6.10 (Interpretation for Algorithm 5.4 (cont.)) Recall that in the setting of Algorithm 5.4 the parameter $\beta_k \geqslant 0$ can be arbitrary, by Corollary 5.2(ia). Since the oracle output satisfies (1.2), the effective decrease is equal to $\delta_k^E = f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - f_{u_{k+1}} - \bar{\eta}_{u_{k+1}}$. Satisfaction of (6.12) follows from the property (6.13):

$$\sum_{k\in\hat{K}} \delta_k^E = f_{\hat{u}_1} + \bar{\eta}_{\hat{u}_1} - \lim_{k\in\hat{K}}(f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k}) = f_{\hat{u}_1} + \bar{\eta}_{\hat{u}_1} - \lim_{k\in\hat{K}}(f(\hat{u}_k) - \eta_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k}) < +\infty\,,$$

because $\eta_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} \leqslant 2\bar{\eta}_{\hat{u}_1}$. □

We are now in a position to prove convergence of a generic proximal bundle method. For the result to hold, oracle errors play no major role. By contrast, the prox-stepsize management needs to be done in a manner that combines harmoniously all the various requirements in the different statements. The following rule, depending on parameters $\underline{t} > 0$ and $\sigma \in (0,1]$, addresses this issue. The update prevents decreasing the stepsize at null steps if noise was detected, by introducing a binary variable $\mathtt{na}_k$, set to 1 if after the iteration generating the current $\hat{u}_k$ there was a loop attenuating noise in Step 1, and set to 0 otherwise:

$$\begin{cases} t_k \geqslant t_{k-1} + \underline{t} & \text{if } k-1 \text{ is a noisy step} \\ t_k \geqslant \underline{t} & \text{if } k-1 \text{ is a descent step} \\ t_k \in \left[\max\left(\underline{t}, (1-\mathtt{na}_{k-1})\sigma t_{k-1} + \mathtt{na}_{k-1} t_{k-1}\right), t_{k-1}\right] & \text{if } k-1 \text{ is a null step}. \end{cases} \tag{6.14}$$

Theorem 6.11 (Convergence) *Consider the Algorithmic Pattern 4.2 with Step 1 from* (5.6) *applied with an oracle* (1.2) *with locally bounded inaccuracy, as in* (6.8)*.*
Suppose that in the set `pars`

$$\begin{array}{l} \textit{the model satisfies } (4.10), \\ \textit{the level is given as in } (3.8), \textit{ and} \\ \textit{the prox-stepsize update satisfies the rule } (6.14). \end{array} \tag{6.15}$$

If in the set `desc`

$$\begin{array}{l} \textit{the effective decrease from } (4.3) \textit{ satisfies } (6.11) \textit{ and } (6.12) \textit{ and} \\ \textit{the parameters } \alpha_k \textit{ and } \beta_k \textit{ satisfy } (5.4) \textit{ if } (5.2) \textit{ is not automatic} \\ \qquad \textit{or } \alpha_k \in [0,1] \textit{ otherwise}, \end{array} \tag{6.16}$$

then the algorithm always generates some K^∞-subsequence that is "optimal", in the sense that (4.8) *is satisfied and Theorem 4.4 applies.*

Proof If Step 1 needs to test (5.2) and there is an infinite loop of noise attenuation, since the update in (6.14) drives t_k to infinity in this case, Corollary 5.3 gives the result. Otherwise, the loop in Step 1 is always finite and the algorithm generates either a last descent step at iteration $\hat{k}$ followed by a null-step tail, or $\hat{K}$ in (6.2) is infinite. In the first case (6.14) satisfies the prox-stepsize conditions in Theorem 6.4. As (3.4) implies satisfaction of (6.5) with $\hat{\eta} = \eta^M$ the theorem applies. By Proposition 6.7, the model decrease vanishes and the assertion follows from Proposition 6.1, written with $K^\infty = \{k > \hat{k}\}$. Finally, if infinitely many descent steps are generated, the assumption that $\lim_{k\in\hat{K}} \delta_k^M = 0$ follows from Proposition 6.9 and the result follows once again from Proposition 6.1, this time written with $K^\infty = \hat{K}$. □

Corollary 6.12 (Interpretation for Algorithm 5.4 (end)) *If problem* (1.1) *satisfies* (6.13)*, any limit point $\hat{u}$ of the Controllable Bundle Algorithm 5.4 is η_∞-optimal with $\eta_\infty \leqslant \bar{\eta}_{\hat{u}_1}$.*

Proof In this variant the sets `pars` and `desc` are given in (5.8) and (5.9), respectively. In particular, the sequence of error bounds is nonincreasing and there is no need of noise attenuation, by Corollary 5.2(ia). The prox-stepsize update in (5.8) fits the rule (6.14) and, as specified in Section 5.2, the oracle inaccuracy is controllable with a nonincreasing sequence of error bounds, so (6.8) is satisfied with $\eta(R) = \bar{\eta}_{\hat{u}_1}$. Finally, as explained in Remarks 6.8 and 6.10, both (6.11) and (6.12) are satisfied for the choices for δ_k^N and δ_k^E in (5.9). Theorem 6.11 applies with $\eta^M = 0$ because both the oracle and the model are lower. □

7 Instances

This section reviews a number of bundle methods entering our framework. For each variant, we need to check if the oracle is bounded (in the sense of (6.8) or of the Remark 6.5) and that both (6.15) and (6.16) are satisfied by the sets `pars` and `desc` of the specific method.

Regarding conditions (6.15), we assume the prox-stepsize update is given by (6.14); we only need to check that the level and the model satisfy (3.8) and (4.10).

As for (6.16), we first determine if the algorithm needs to attenuate noise. If such is the case, parameters α_k and β_k will be given by (5.4); otherwise $\alpha_k \in [0,1]$. Therefore, we shall only check that the effective decrease in `desc` satisfies conditions (6.11) and (6.12), and to show this last property, we assume that (1.1) satisfies (6.13).

7.1 A Collection of Bundle Methods for Lower Oracles and Models

We now review several bundle methods working with models satisfying

$$f_k^M(\cdot) \leqslant \check{f}_k(\cdot) \quad \text{for all } k, \tag{7.1}$$

which is the case of the cutting-plane model endowed with *bundle compression*. To keep the master-program size controlled, this mechanism replaces past linearizations by the aggregate one, $f_{-k}^L(\cdot)$, without impairing convergence. Since we also consider lower oracles, as in (2.3), (3.4) holds and in (4.10) the error is null $\eta^M = 0$.

For all the methods in this section $\ell_k := \hat{f}_k$. By Corollary 5.2, Step 1 never needs to attenuate noise. The convergence result for these methods is given by Theorem 6.2.

7.1.1 Exact Oracles

Like the *classical* bundle methods, the *spectral* algorithms [12,23] use an exact oracle and, hence, $\ell_k = f(\hat{u}_k)$. Both the polyhedral cutting-plane model (classical bundle) and the non-polyhedral model in [12] (spectral bundle) satisfy (7.1). Taking $\delta_k^E := f(\hat{u}_k) - f(u_{k+1})$ gives (6.11), because the left hand side therein is null. The final property (6.12) is direct from (6.13). The descent test (3.6)

$$f(u_{k+1}) \leqslant f(\hat{u}_k) - m\delta_k^N \quad \text{for} \quad \delta_k^N = f(\hat{u}_k) - f_k^M(u_{k+1}) - \alpha_k t_k |\hat{g}_k|^2$$

usually takes $\alpha_k \equiv 0$ in (4.3), but any value in $[0,1]$ could be used instead.

7.1.2 Partially Inexact Oracles

The *partially inexact* proximal bundle method was introduced in [10] and revisited in [19], for a level variant see [22]. To ensure that the function information is exact at descent steps the oracle should be a particular case of the partly asymptotically exact given in Example 2.2, with $\bar{\eta}_u = 0$ whenever $f_u \leqslant \gamma_u$. Then $\ell_k := f(\hat{u}_k)$ and the bold choice $\delta_k^E := f(\hat{u}_k) - f_{u_{k+1}}$ yields both a null left hand side in (6.11) and satisfaction of (6.12), by (6.13).

7.1.3 Incremental Bundle Method

The *incremental bundle* method [7] was developed for lower oracles whose errors at descent iterates vanish fast enough, as in (5.7). The realistic choice $\delta_k^E = \hat{f}_k - f_{u_{k+1}}$ yields in (6.11) a null left hand side. Condition (6.12) is satisfied because $\hat{f}_k \leqslant f_{\hat{u}_k} + \eta_{\hat{u}_k}$ and, hence,

$$(0 \leqslant) \sum_{k\in\hat{K}} \delta_k^E \leqslant \sum_{k\in\hat{K}} (f_{\hat{u}_k} + \eta_{\hat{u}_k} - f_{\hat{u}_{k+1}}) = \sum_{k\in\hat{K}} (f_{\hat{u}_k} - f_{\hat{u}_{k+1}}) + \sum_{k\in\hat{K}} \eta_{\hat{u}_k} < +\infty.$$

When $\hat{K}$ is finite, the last descent step $\hat{u}$ is η_∞-solution with $\eta_\infty = \eta_{\hat{u}}$ not necessarily zero. When $\hat{K}$ is infinite the algorithm determines asymptotically an optimal solution to problem (1.1), because in Theorem 4.4, $\eta_\infty = 0$. Thanks to (4.7), the *unboundedness detection* loop in [7] is not required for our convergence results to hold (see the errata in [6]).

7.1.4 Asymptotically Exact Bundle Method

This new method is the proximal version of the level bundle method for oracles with on-demand accuracy considered in [22]. The variant is suitable for lower oracles that are eventually exact at descent iterates, like the partly asymptotically exact oracle in Example 2.2: $\eta_u^g \equiv 0$, $\lim_k \eta_{\hat{u}_k} = 0$ in (1.2), and an error bound $\bar{\eta}_{\hat{u}_k}$ is known.

The conservative choice $\delta_k^E = f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - f_{u_{k+1}}$ yields satisfaction of (6.11):

$$\ell_k - f_{u_{k+1}} - \delta_k^E = \hat{f}_k - (f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k}) \leqslant \hat{f}_k - f(\hat{u}_k)$$

because the oracle is lower. As for (6.12), by (6.13) combined with (1.2)

$$(0 \leqslant) \sum_{k\in\hat{K}} (f_{\hat{u}_k} - f_{u_{k+1}}) = \sum_{k\in\hat{K}} (f_{\hat{u}_k} - f_{\hat{u}_{k+1}}) = f_{\hat{u}_1} - \lim_{k\in\hat{K}} f_{\hat{u}_k} = f_{\hat{u}_1} - \lim_{k\in\hat{K}} (f(\hat{u}_k) - \eta_{\hat{u}_k}) < +\infty. \tag{7.2}$$

So $(f_{\hat{u}_k} - f_{u_{k+1}}) \to 0$ and (6.12) holds because $\bar{\eta}_{\hat{u}_k} \to 0$ by the oracle assumption.

The descent test (3.6) is $f_{u_{k+1}} \leqslant f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - m\delta_k^N$ for $\delta_k^N = \hat{f}_k - f_k^M(u_{k+1}) - \alpha_k t_k |\hat{g}_k|^2$.

We now show that, even though exact evaluations are not needed, convergence for this method is the same than for Section 7.1.2. For this, it suffices to show satisfaction of (5.7), which in turn only needs to prove that $\eta_{\hat{u}} = 0$ if $K^\infty = \{k > \hat{k}\}$ (because $\lim_{k\in K^\infty} \eta_{u_k} = 0$ when $K^\infty = \hat{K}$, by assumption). When there is a last descent step $\hat{u}$, the definitions of ℓ_k and δ_k^E imply that $f_{u_{k+1}} > f_{\hat{u}_k} + \bar{\eta}_{\hat{u}_k} - m\,\delta_k^N$ if (3.6) does not hold. Then

$$\begin{aligned}
f_{u_{k+1}} - f_k^M(u_{k+1}) &> f_{\hat{u}} + \bar{\eta}_{\hat{u}} - m\,\delta_k^N - f_k^M(u_{k+1}) && \text{[by “not” (3.6)]}\\
&= f_{\hat{u}} + \bar{\eta}_{\hat{u}} - m\,(\delta_k^M - \alpha_k t_k |\hat{g}_k|^2) - f_k^M(u_{k+1}) && \text{[by (4.3)]}\\
&= f_{\hat{u}} + \bar{\eta}_{\hat{u}} - \ell_k - m\,(\delta_k^M - \alpha_k t_k |\hat{g}_k|^2) + \delta_k^M && \text{[by (3.7)]}\\
&= \big(f_{\hat{u}} + \bar{\eta}_{\hat{u}} - \hat{f}_k\big) + (1-m)\,\delta_k^M + m\,\alpha_k t_k |\hat{g}_k|^2. && \text{[because } \ell_k = \hat{f}_k\text{]}
\end{aligned}$$

In view of (6.6), $\limsup_k \big(f_{\hat{u}} + \bar{\eta}_{\hat{u}} - \hat{f}_k\big) + (1-m)\,\delta_k^M + m\,\alpha_k t_k |\hat{g}_k|^2 \leqslant 0$. However, the second and third terms above are non-negative ($\delta_k^M \geqslant 0$ by Proposition 6.9). Similarly for the first term because, by the first line in (1.2) together with (3.9) and (7.1) evaluated at $\hat{u}$,

$$f_{\hat{u}} + \bar{\eta}_{\hat{u}} \geqslant f(\hat{u}) \geqslant \hat{f}_k\,. \tag{7.3}$$

Therefore $f_{u_{k+1}} - f_k^M(u_{k+1}) \to 0$, hence $\hat{f}_k \to f_{\hat{u}} + \bar{\eta}_{\hat{u}}$. The result follows because in the limit the right hand side term in (7.3) yields that $f_{\hat{u}} + \bar{\eta}_{\hat{u}} = f(\hat{u})$ and, by the first line in (1.2),

$$f(\hat{u}) \leqslant f_{\hat{u}} + \eta_{\hat{u}} \leqslant f_{\hat{u}} + \bar{\eta}_{\hat{u}} = f(\hat{u}),$$

so $\eta_{\hat{u}} = \bar{\eta}_{\hat{u}} = 0$, as stated.

Before passing to constrained problems, we mention that the inexact bundle methods [13,18], suitable for dumb oracles (possibly not of lower type), also fits our convergence framework (with a slight generalization for α_k and β_k in (4.3) and (5.2), respectively).

7.2 Constrained Problems

In the following generalization of Example 2.3, the convex problem

$$\begin{cases} \min\ h(u) \\ \text{s.t.}\ \ c(u) \leqslant 0 \\ \qquad u \in U, \end{cases}$$

has an "easy" objective function $h(\cdot)$. a scalar constraint $c(\cdot)$ hard to evaluate and a set U assumed to be a simple polyedron. Suppose a Slater condition holds for this problem. We now consider two methods for solving this type of problems.

7.2.1 Using Improvement Functions

Letting $\bar{h}$ denote the optimal value, solving the problem above is equivalent to minimizing

$$f(u) := \max\{h(u) - \bar{h}, c(u)\} \quad \text{over the set } U.$$

Being polyhedral, U can be introduced directly in the master-program (3.3), so we are essentially in the setting (1.1). When the optimal value is unknown, the value of the objective at the current center can be used as a replacement, possibly adding a penalization when the center is unfeasible, to prevent zigzagging. For example, we can take the approximation

$$f(u) \approx \max\{h(u) - h(\hat{u}_k) - \rho_k \max(c_{\hat{u}_k}, 0), c_u - \sigma_k \max(c_{\hat{u}_k}, 0)\},$$

depending on parameters ρ_k, σ_k and where we made explicit that the oracle for $h(\cdot)$ and $c(\cdot)$ are respectively exact and inexact.

The algorithms analyzed in [1] suppose there is an upper oracle for $c(\cdot)$ satisfying (4.10) and use the approximate improvement function above to put a bundle methodology in place. To revisit these methods, we consider instead that there is an inexact oracle for the exact improvement function $f(\cdot)$. Its output is the right hand side term above:

$$f_{u_{k+1}} := \max\{h(u_{k+1}) - h(\hat{u}_k) - \rho_k \max(c_{\hat{u}_k}, 0), c_{u_{k+1}} - \sigma_k \max(c_{\hat{u}_k}, 0)\}.$$

As for the approximate subgradient, it is given by either the exact subgradient $g^h(u_{k+1})$ or the inexact one $g^c_{u_{k+1}}$, depending on which term realizes the maximum. Since the c-oracle is of upper type, so is the f-oracle and the method needs a noise attenuation loop in Step 1.

The cutting-plane models for the objective and constraint functions give the model:

$$f_k^M(u) = \max\{\check{h}_k(u) - h(\hat{u}_k) - \rho_k \max(c_{\hat{u}_k}, 0), \check{c}_k(u) - \sigma_k \max(c_{\hat{u}_k}, 0)\}.$$

As the oracle is exact for $h(\cdot)$, satisfaction of (4.10) for this model is inherited from the fact that the condition holds for $\check{c}_k(\cdot)$, assuming the set U is compact. Also, in view of Remark 6.5, the f-oracle satisfies (6.8).
By (3.8), the level is

$$\ell_k = f_{\hat{u}_k} = (1-\sigma_k)\max(c_{\hat{u}_k}, 0)\,.$$

Regarding the parameters in (5.4), they relate as follows

$$2\alpha_k = \alpha_k^{[1]} \quad \text{and} \quad 2\beta_k = 1-\beta_k^{[1]}\,.$$

The (bold) effective decrease is

$$\delta_k^E := \ell_k - f_{u_{k+1}}\,,$$

which gives automatic satisfaction of (6.11), required for the null-step tail.

Showing the condition (6.12) is more involved, since it requires considering feasibility of the descent sequence $\{\hat{u}_k\}$. To ease the presentation, we consider one particular case in which the penalty ρ_k is driven to infinity:

$$\rho_{k+1} = \rho_k + 1 \text{ at each iteration } k \text{ satisfying (3.6) for which } c_{u_{k+1}} > 0$$

(the setting in [1] is more general, allowing different penalty parameters choices as well as an inexact h-oracle). To show (6.12), i.e., that the effective decrease vanishes, first note that because $f_{u_{k+1}}$ is a max-function, as soon as a feasible descent step is found, all the subsequent ones remains feasible. Suppose first the descent sequence remains infeasible: $c_{\hat{u}_k} > 0$ for all $k \in \hat{K}$ and $K^\infty = \hat{K}$. The penalty update implies that in the max-operation defining f_u the second term eventually prevails. Hence, for $k \in \hat{K}$ sufficiently large

$$\delta_k^E = \ell_k - f_{u_{k+1}} = (1-\sigma_k)c_{\hat{u}_k} - (c_{\hat{u}_{k+1}} - \sigma_k c_{\hat{u}_k}) = c_{\hat{u}_k} - c_{\hat{u}_{k+1}}\,.$$

Summing over $k \in \hat{K}$ shows that the effective decrease series is convergent, so (6.12) holds (regardless of the value of σ_k). The remaining case refers to descent steps eventually becoming feasible, so that

$$\delta_k^E = \ell_k - f_{u_{k+1}} = 0 - \max\{h(\hat{u}_{k+1}) - h(\hat{u}_k), c_{\hat{u}_{k+1}}\} \leqslant -h(\hat{u}_{k+1}) + h(\hat{u}_k)$$

for large $k \in \hat{K}$. Assuming once more (6.13) and making the sum gives (6.12).

By Theorem 6.11, the limit points of the sequence either solve (1.1) within the accuracy bound $\eta_\infty + \eta^M$ or the problem is unfeasible, up to the same accuracy (if calculations are asymptotically exact, the Slater assumption rules out the last option).

7.2.2 Using Exact Penalties

The Slater condition ensures that set of Lagrange multipliers is nonempty and bounded. Therefore, for any $\rho > \max\lambda$, solutions to the constrained problems can be found by minimizing the exact penalty function

$$f(u) := h(u) + \rho \max\{c(u), 0\},$$

over the set U, which puts us once more in the setting (1.1). For such a penalty parameter ρ, we may take $f_k^M(u) := h_k^M(u) + \rho \max\{c_k^M(u), 0\}$ as a model for $f(\cdot)$, and each iterate of the method can be obtained by minimizing, over $u \in U$, the function

$$f_k^S(u) = f_k^M(u) + \frac{1}{2t_k}|u - \hat{u}_k|^2 = h_k^M(u) + \rho \max\{c_k^M(u), 0\} + \frac{1}{2t_k}|u - \hat{u}_k|^2.$$

The approach bottleneck is estimating the penalty: making successive approximations of this parameter amounts to having a more accurate inexact oracle for the function $f(\cdot)$. As before, the f-oracle assumptions follow from the h- and c-oracles, and similarly for the models. A possible penalty update is $\rho_k = \max\{\rho_{k-1}, \lambda_k + 1\}$ for a Lagrange multiplier $\lambda_k \geqslant 0$ of

$$\min_{u \in U} h_k^M(u) + \frac{1}{2t_k}|u - \hat{u}_k|^2 \;\text{ s.t. }\; c_k^M(u) \leqslant 0, \tag{7.4}$$

a Successive Quadratic Programming-like problem that gives the next iterate u_{k+1}. This is a feasible problem (by the Slater assumption) whose solution also solves

$$\min_{u \in U} h_k^M(u) + \rho_k \max\{c_k^M(u), 0\} + \frac{1}{2t_k}|u - \hat{u}_k|^2.$$

for sufficiently large ρ_k.

For the approach to behave as an exact penalty method, the penalties needs to stabilize eventually. This requires the Lagrange multiplier sequence to be bounded, a property that that we prove under appropriate assumptions.

Lemma 7.1 *Suppose U is a bounded set and there exists $u^0 \in U$ such that $c(u^0) < 0$. If the prox-stepsizes satisfy* (6.3) *and the c-model satisfies*

$$c_k^M(\cdot) \leqslant \eta^M \quad \textit{for some bound } 0 \leqslant \eta^M < -c(u^0),$$

the sequence of Lagrange multipliers $\{\lambda_k\}$ of (7.4) *is bounded.*

Proof The optimality condition for (7.4) provides $p_k^h \in \partial h_k^M(u_{k+1})$, $p_k^c \in \partial c_k^M(u_{k+1})$, $p_k^u \in N_U(u_{k+1})$, and $\lambda_k \geqslant 0$ satisfying $p_k^h + \frac{u_{k+1}-\hat{u}}{t_k} + \lambda_k p_k^c + p_k^u = 0$. Without loss of generality, suppose $\lambda_k > 0$, so that $p_k = p_k^c + p_k^u/\lambda_k$ and $\lambda_k p_k = -(p_k^h + \frac{u_{k+1}-\hat{u}}{t_k})$ yield the identity $\lambda_k|p_k|^2 = -\left\langle p_k, (p_k^h + \frac{u_{k+1}-\hat{u}}{t_k})\right\rangle$. Assume for the moment $p_k \neq 0$; by Cauchy-Schwarz,

$$\lambda_k = -\frac{1}{|p_k|^2}\left(\left\langle p_k, p_k^h\right\rangle + \left\langle p_k, \frac{u_{k+1}-\hat{u}_k}{t_k}\right\rangle\right) \leqslant \frac{|p_k|}{|p_k|^2}\left(|p_k^h| + \frac{|u_{k+1}-\hat{u}_k|}{t_k}\right).$$

As U is bounded and $h_k^M(\cdot)$ is convex,n $p_k^h \in \partial h_k^M(u_{k+1})$ is bounded as well. Together with (6.3) we see that there exists a constant $M > 0$ such that $\lambda_k \leqslant M/|p_k|$ for all k.
It remains to show that the sequence $\{|p_k|\}$ is bounded away from zero. The definitions of p_k^c and p_k^u imply that $c_k^M(u_{k+1}) + \langle p_k^c, u^0 - u_{k+1}\rangle \leqslant c_k^M(u^0)$ and $\frac{1}{\lambda_k}\langle p_k^u, u^0 - u_{k+1}\rangle \leqslant 0$. By

adding these two inequalities and remembering that $c_k^M(u_{k+1}) = 0$ because $\lambda_k > 0$, we get $\langle p_k, u^0 - u_{k+1}\rangle \leqslant c_k^M(u^0)$. Therefore,

$$-|p_k||u^0 - u_{k+1}| \leqslant \langle p_k, u^0 - u_{k+1}\rangle \leqslant c_k^M(u^0) \leqslant c(u^0) + \eta^M < 0\,,$$

where the last inequality follows from the assumption on u^0 and η^M. Since U is a bounded set, we conclude that $\liminf_k |p_k| > 0$, and hence $\{\lambda_k\}$ is a bounded sequence. □

Once the penalty parameter stabilizes at a value ρ, Theorem 6.11 applies: the limit points of the sequence $\{\hat{u}_k\}$ solve the constrained problem within an accuracy bound depending also on the value $\limsup_{k\in\hat{K}}\{0, \rho - \rho_{\hat{k}}\}$, the asymptotic error made when estimating the penalty at descent steps.

7.3 Composite Functions

The composite bundle method [24] uses the approximation $F(\cdot;\hat{u})$ in Example 2.4 (with $\hat{u} = \hat{u}_k$) as an economic intermediate model for the function $f(\cdot) = (h\circ c)(\cdot)$. The reason is that evaluating the f-subgradients is expensive, because they need computing the c-Jacobian:

$$\partial f(u) = Dc(u)^\top \partial h(C) \quad \text{for} \quad C = c(u)\,.$$

To interpret this method in our setting, consider as oracle output

$$\begin{aligned} f_{u_{k+1}} &:= F(u_{k+1};\hat{u}_k) = h(C_{k+1}) \quad \text{for} \quad C_{k+1} := c(\hat{u}_k) + Dc(\hat{u}_k)(u_{k+1} - \hat{u}_k) \\ g_{u_{k+1}} &:= Dc(\hat{u}_k)^\top G_{k+1} \quad \text{for} \quad G_{k+1} \in \partial h(C_{k+1})\,. \end{aligned}$$

Note that the oracle is exact at the center $\hat{u}_k$. Also, by smoothness of the operator c, this oracle satisfies (6.8) for each fixed stability center $\hat{u}_k$, in the sense explained in Remark 6.5.

By convexity, the cutting-plane model $\check{h}(\cdot)$ stays below the function $h(\cdot)$ and by positive homogeneity, the model

$$f_k^M(\cdot) := \check{h}(c(\hat{u}_k) + Dc(\hat{u}_k)(\cdot - \hat{u}_k))$$

stays below $F(\cdot;\hat{u}_k)$, but not necessarily below $f(\cdot)$. An interesting feature of the approach is that, even though the model is not of lower type ((3.4) may not hold), the method *does not need* to attenuate noise, thanks to the special model structure. More precisely, first note that

$$f_k^M(\hat{u}_k) = (\check{h}\circ c)(\hat{u}_k)) \leqslant (h\circ c)(\hat{u}_k)) = f(\hat{u}_k)\,.$$

Then, because $\hat{g}_k \in \partial f_k^M(u_{k+1})$ by Lemma 4.1, from (4.5) to see that

$$f(\hat{u}_k) = f_k^M(\hat{u}_k) \geqslant f_k^M(u_{k+1}) + \langle \hat{g}_k, \hat{u}_k - u_{k+1}\rangle = f_{-k}^L(\hat{u}_k)\,.$$

So taking as level $\ell_k = f(\hat{u}_k)$ gives that the aggregate error in (4.4) is zero and by item (ii) in Lemma 5.1 there is no need of noise attenuation. In [24] a null step is declared whenever there is no descent for the approximating function, corresponding to the bold choice

$$\delta_k^E := F(\hat{u}_k;\hat{u}_k) - F(u_{k+1};\hat{u}_k) = f(\hat{u}_k) - h(C_{k+1}) = \ell_k - f_k^M(u_{k+1})\,,$$

which trivially ensures (6.11). Regarding (6.12), the composite bundle method makes an additional test before declaring a descent step, called of *backtracking*. For $k \in \hat{K}$ rewrite the effective decrease as follows:

$$f(\hat{u}_k) - f(\hat{u}_{k+1}) = [f(\hat{u}_k) - h(C_{k+1})] + [h(C_{k+1}) - f(\hat{u}_{k+1})] = \delta_k^E + [h(C_{k+1}) - f(\hat{u}_{k+1})]\,.$$

The usual argument invoking (6.13) would give (6.12) if the second bracket eventually vanished. The backtracking test declares a descent step if, in addition to (3.6), the condition

$$\langle G_{k+1}, C_{k+1} - c(u_{k+1})\rangle \geqslant -m_2\delta_k^N \text{ for } G_{k+1} \in \partial h(c(u_{k+1}))$$

holds. Otherwise a backtracking step is done: the prox-stepsize is decreased, maintaining the model and the center. As the number of backtracking steps is finite ([24]), eventually the algorithm generates sets K^∞ as in (6.2). The backtracking condition is easy to test, because the additional oracle call does not involve the expensive Jacobian at u_{k+1}. Such a computation is done only if the backtracking test is passed, to define the next cheap oracle, i.e. $F(\cdot;\hat{u}_{k+1})$. Positively homogeneous functions are support functions and as such $h(\cdot)$ satisfies:

$$h(c(u_{k+1})) = \langle G_{k+1}, c(u_{k+1})\rangle \quad \text{and} \quad h(C_{k+1}) \geqslant \langle G_{k+1}, C_{k+1}\rangle\,.$$

As a result, checking the need of backtracking is equivalent to testing if

$$[h(C_{k+1}) - f(\hat{u}_{k+1})] \geqslant -m_2\delta_k^N\,.$$

Together with (3.6) this means that when $K^\infty = \hat{K}$ we have that

$$f(\hat{u}_k) - f(\hat{u}_{k+1}) = \delta_k^E + [h(C_{k+1}) - f(\hat{u}_{k+1})] \geqslant (m_1 - m_2)\delta_k^N\,.$$

Since $m_1 - m_2 > 0$, the sum and (6.13) imply that the series of nominal decreases converges. Therefore the effective decrease series converges too and (6.12) follows.

By Theorem 6.11, the limit points of the sequence satisfy (4.11), and in view of the level definition, they solve (1.1) exactly.

References

1. W. van Ackooij and C. Sagastizábal. Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. Optimization Online Report 3711, 2012.
2. A. Belloni and C. Sagastizábal. Dynamic bundle methods. *Mathematical Programming*, 120(2):289–311, 2009.
3. M.T. BenAmor, J. Desrosiers, and A. Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009.
4. E. Cheney and A. Goldstein. Newton's method for convex programming and Tchebycheff approximations. *Numerische Mathematik*, 1:253–268, 1959.
5. R. Correa and C. Lemaréchal. Convergence of some algorithms for convex minimization. *Mathematical Programming*, 62(2):261–275, 1993.
6. G. Émiel and C. Sagastizábal. Incremental-like bundle methods with application to energy planning (corrigendum). Optimization Online report 2147, 2009.
7. G. Émiel and C. Sagastizábal. Incremental-like bundle methods with application to energy planning. *Computational Opt. and Appl.*, 46(2):305–332, 2010.
8. A. Frangioni. Generalized bundle methods. *SIAM Journal on Optimization*, 13(1):117–156, 2003.
9. A. Frangioni, 2008. Private communication.
10. M. Gaudioso, G. Giallombardo, and G Miglionico. An incremental method for solving convex finite min-max problems. *Mathematics of Operations Research*, 31(1), 2006.
11. J.-L. Goffin, A. Haurie, and J.-Ph. Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 38(2):284–302, 1992.
12. C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.
13. M. Hintermüller. A proximal bundle method based on approximate subgradients. *Computational Optimization and Applications*, 20(3):245–266, 2001. 10.1023/A:1011259017643.
14. J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer Verlag, Heidelberg, 1993. Two volumes.

15. J. E. Kelley. The cutting plane method for solving convex programs. *J. Soc. Indust. Appl. Math.*, 8:703–712, 1960.
16. K.C. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983.
17. K.C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*. Lecture Notes in Mathematics 1133. Springer Verlag, 1985.
18. K.C. Kiwiel. A proximal bundle method with approximate subgradient linearizations. *SIAM Journal on Optimization*, 16(4):1007–1023, 2006.
19. K.C. Kiwiel. Bundle methods for convex minimization with partially inexact oracles. *Comp. Opt. Appl.*, 2013. Accepted for publication.
20. C. Lemaréchal. Nonsmooth optimization and descent methods. Research Report 78-4, IIASA, 1978.
21. R.E. Marsten, W.W. Hogan, and J.W. Blankenship. The boxstep method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.
22. W. Oliveira and C. Sagastizábal. Level bundle methods for oracles with on-demand accuracy. *Optimization Methods and Software*, 2013. Accepted for publication.
23. F. Oustry. A second-order bundle method to minimize the maximum eigenvalue function. *Mathematical Programming*, 89(1):1–33, 2000.
24. C. Sagastizábal. Composite Proximal Bundle Method. *Mathematical Programming*, 140(1):189–233, 2013.
25. A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. MPS-SIAM Series on Optimization. SIAM - Society for Industrial and Applied Mathematics and Mathematical Programming Society, Philadelphia, 2009.
26. M.V. Solodov. On approximations with finite precision in bundle methods for nonsmooth optimization. *J. Opt. Theory and Appl.*, 119(1):151–165, 2003.